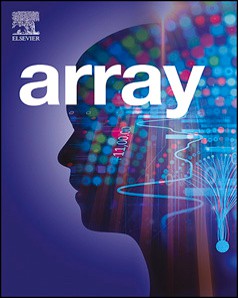
[Array 17 (2023) 100268](https://doi.org/10.1016/j.array.2022.100268)



Contents lists available at [ScienceDirect](https://www.elsevier.com/locate/array)

Array

journal homepage: [www.elsevier.com/locate/array](http://www.elsevier.com/locate/array)

Evaluation of next-generation high-order compressible fluid dynamic solver on cloud computing for complex industrial flows

R. Al Jahdali [a](#_bookmark0),[∗](#_bookmark3), S. Kortas [c](#_bookmark2), M. Shaikh [c](#_bookmark2), L. Dalcin [a](#_bookmark0), M. Parsani [a](#_bookmark0),[b](#_bookmark1)

a *King Abdullah University of Science and Technology (KAUST), Computer Electrical and Mathematical Science and Engineering Division (CEMSE), Extreme Computing Research Center (ECRC), 23955-6900 Thuwal, Saudi Arabia*

b *King Abdullah University of Science and Technology (KAUST), Physical Sciences and Engineering Division (PSE), 23955-6900 Thuwal, Saudi Arabia*

c *KAUST Supercomputing Core Laboratory, 23955-6900 Thuwal, Saudi Arabia*

A R T I C L E I N F O A B S T R A C T

*Keywords:*

Cloud computing

Amazon Web Services Elastic Compute Cloud Fluid mechanics

Compressible Navier–Stokes equations Fully-discrete entropy stable algorithms

Industrially relevant computational fluid dynamics simulations frequently require vast computational resources that are only available to governments, wealthy corporations, and wealthy institutions. Thus, in many contexts and realities, high-performance computing grids and cloud resources on demand should be evaluated as viable alternatives to conventional computing clusters. In this work, we present the analysis of the time-to-solution and cost of an entropy stable collocated discontinuous Galerkin (SSDC) compressible computational fluid dynamics framework on Ibex, the on-premises cluster at KAUST, and the Amazon Web Services Elastic Compute Cloud for complex compressible flows. SSDC is a prototype of the next generation computational fluid dynamics frameworks developed following the road map established by the NASA CFD vision 2030. We simulate complex flow problems using high-order accurate fully-discrete entropy stable algorithms. In terms of time-to-solution, the Amazon Elastic Compute Cloud delivers the best performance, with the Graviton2 processors based on the Arm architecture being the fastest. However, the results also indicate that the Ibex nodes based on the AMD Rome architecture deliver good performance, close to those observed for the Amazon Elastic Compute Cloud. Furthermore, we observed that computations performed on the Ibex on-premises cluster are currently less expensive than those performed in the cloud. Our findings could be used to develop guidelines for selecting high-performance computing cloud resources to simulate realistic fluid flow problems.

# Introduction

The process of predicting fluid flow (such as gases and liquids), mass transfer, chemical reactions, and other associated phenomena with a computer during the design or production process is known as computational fluid dynamics (CFD). State-of-the-art CFD is critical in modeling many physical phenomena, including biomedicine [[1](#_bookmark48)], plasma science [[2](#_bookmark49)], climate [[3](#_bookmark50),[4](#_bookmark51)], weather [[5](#_bookmark52),[6](#_bookmark53)], and aerodynamics [[7](#_bookmark54),[8](#_bookmark55)], and for elucidating and analyzing the underlying mechanisms of these complex phenomena. In the aeronautics and aerospace fields, aggressive use of CFD has resulted in significant reductions in wind tunnel time as well as a reduction in the number of experimental rig tests. Thus, through the use of CFD, industries, governments and national laboratories have been able to produce products faster and at reduced costs [[8](#_bookmark55)], saving hundreds of millions of dollars. In addition to reducing testing requirements, CFD has the additional potential of providing superior understanding and insight into the critical physical

phenomena limiting component performance, thereby opening up new frontiers in a variety of fields.

Among the critical goals, NASA’s vision roadmap [[9](#_bookmark56)] indicates the computational procedure and algorithms to perform realistic aerody- namic simulations by 2030 with the next generation of computing architectures. The path to the final aerodynamic simulation goal in- cludes a scheduled demonstration of various critical technologies and production scalable entropy-stable solvers are one of them [[9](#_bookmark56)]. A prototype of these new solvers, named SSDC, where ‘‘Entropy’’ is represented in the acronym by its thermodynamic symbol, ‘‘S’’, has been developed and deployed for industrial collaborations [[10](#_bookmark57)]. SSDC combines a state-of-the-art, provably stable, adaptive order solver, with software engineering that exploits hybrid shared-distributed memory capabilities.

High-performance computing (HPC) has had a significant influence on CFD. The demand for HPC systems increases with the increasing

∗ Corresponding author.

*E-mail addresses:* [Rasha.Aljahdali@kaust.edu.sa](mailto:Rasha.Aljahdali@kaust.edu.sa) (R. Al Jahdali), [samuel.kortas@kaust.edu.sa](mailto:samuel.kortas@kaust.edu.sa) (S. Kortas), [mohsin.shaikh@kaust.edu.sa](mailto:mohsin.shaikh@kaust.edu.sa) (M. Shaikh), [dalcinl@gmail.com](mailto:dalcinl@gmail.com) (L. Dalcin), [matteo.parsani@kaust.edu.sa](mailto:matteo.parsani@kaust.edu.sa) (M. Parsani).

<https://doi.org/10.1016/j.array.2022.100268>

Received 25 October 2022; Received in revised form 1 December 2022; Accepted 5 December 2022

Available online 9 December 2022

2590-0056/© 2022 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license ([http://creativecommons.org/licenses/by-](http://creativecommons.org/licenses/by-nc-nd/4.0/) [nc-nd/4.0/](http://creativecommons.org/licenses/by-nc-nd/4.0/)).

complexity of the simulation, which is a primary element in fluid flow simulations. Owning high-performance computer facilities helps to process simulations faster and allows one to perform large-scale sim- ulations. Supercomputer centers and other HPC computing capabilities are being used to perform massive simulations in academia, national laboratories, and industry. Nevertheless, the cost of the computing cluster may be exorbitant to purchase and maintain, and therefore, it may not be able to keep up with today’s computational demands. Hence, on-demand grids and cloud resources should be viewed as alternatives to traditional computing clusters in various situations and circumstances. A major advantage of cloud resources is the opportunity to use cutting-edge hardware without the requirement for financial investment or IT maintenance costs. In addition, instead of an annual license, the software can be purchased on a pay-per-use basis. Be- cause the industry is interested in using on-demand computing services for engineering simulations, benchmarking the performance of next- generation of compressible CFD solver prototypes, such as SSDC, on cloud resources is essential.

Currently, the industry is experimenting with a variety of ways for incorporating cloud computing into CFD workloads. Thus, it is pivotal to evaluate the performance of parallel systems to determine whether they are capable of running complex industrial CFD problems with up- coming solvers. On one side, important studies in this direction for well established classic solvers based on finite volume, finite differences, and finite elements (up to second order accuracy) have been reported, for instance, by Peña-Monferrer et al. [[11](#_bookmark58)], Ashton et al. [[12](#_bookmark59)] and Turner et al. [[13](#_bookmark60)] for complex industry problems. In particular, in early 2020, the results of the performance of the open-source OpenFOAM software on the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) service for racing vehicles employing a hybrid RANS-LES model were presented by Ashton and co-authors [[12](#_bookmark59)]. The authors demonstrated that AWS might provide an HPC environment that would enable wider usage of high-fidelity CFD methods by permitting higher core counts and reducing turnaround time. In early 2021, in Peña-Monferrer et al.

[[11](#_bookmark58)] presented a hybrid cloud solution for efficient simulation and analysis of drop dispersions by breaking down and study the CFD data analysis pipeline into small microservice-like processes. Turner and co-authors [[13](#_bookmark60)] gave an exhaustive analysis of how well GPU and CPU architectures work and how much they cost for a complete aircraft RANS simulation using the CFD code zCFD. On the other side,

studies on the performance of adaptive non-linearly stable (high-order)

Numerous commercial solvers are currently deployed and utilized on the cloud to simulate fluid flow problems based on the Reynolds- averaged Navier–Stokes (RANS) approach. However, there is no doubt that large eddy simulation (LES) has the potential to deliver results that are more accurate and reliable and will be one of the main approaches to solving challenging fluid flows problems. This study reports the first step towards high-performance CFD simulations on the AWS cloud

cluster for industrial workloads based on LES approaches with an *ℎ𝑝*-

adaptive compressible solver. The main contributions of this work are

as follows. First, we deploy and test a novel prototype of the next generation of compressible CFD solver ‘‘SSDC’’ on the HPC cluster on the cloud. SSDC is a prototype of the framework defined in the NASA CFD 2030 vision, and it has novel capabilities in terms of adaptivity in space and time. Second, we manage and operate the AWS cloud cluster by optimizing the PETSc library and SSDC framework for each of the architectures used in the study. The installation details of the op- [timized](https://github.com/ecrc/petsc_installation_aws_cloud_2022) PETSc library can be found in [https://github.com/ecrc/petsc\_](https://github.com/ecrc/petsc_installation_aws_cloud_2022) [installation\_aws\_cloud\_2022](https://github.com/ecrc/petsc_installation_aws_cloud_2022). The former step is done in conjunction with the Slurm scheduler to simplify the management of jobs. Since we are simulating the LES model, a large number of AWS EC2 instances is needed. Third, we compare the on-premises cluster and the most recent architectures on the AWS cloud in terms of performance and cost.

The paper is organized as follows. In Section [2](#_bookmark4), we present the key ideas and elements of the spatial and temporal algorithms implemented in SSDC, whereas in Section [3](#_bookmark12), we give a brief overview of the soft- ware implementation and infrastructure. Section [4](#_bookmark13) briefly describes the computing environments of the on-premise Ibex cluster and Amazon EC2 cloud computing environment. Section [5](#_bookmark15) presents the performance results of the test case studies to analyze the performance of the SSDC solver on Ibex and Amazon EC2 clusters. Finally, the conclusion and future work are drawn in Section [6](#_bookmark27).

# The compressible Navier–Stokes equations

In this section, we give an overview of the discretization of the compressible Navier–Stokes equations. A detailed presentation of the key elements of the spatial discretization is presented for the advection– diffusion equation in multiple dimensions in [Appendix](#_bookmark37) [A](#_bookmark37).

In the framework of the method of line approach, we first present the spatial discretization and subsequently describe the temporal inte- gration approach. The compressible Navier–Stokes equations in Carte- sian coordinates read

∑

compressible algorithms on the HPC cloud computing is lacking. Those

*𝜕****𝒒***

+ ∑3

*𝜕***T** *𝐼* 3

*𝑥𝑚* =

*𝜕***T** *𝑉*

*𝑥𝑚 ,* ∀

(*𝑥 , 𝑥 , 𝑥*

) ∈ *𝛺, 𝑡* ≥ 0*,*

spatial and temporal discretizations, in the wake of the final NASA 2030 CFD vision report [[9](#_bookmark56)], received an impetus from leading U.S.

*𝜕𝑡*

(

*𝑚*=1

*𝜕𝑥𝑚*

)

*𝑚*=1(

*𝜕𝑥𝑚*

1 2 3

) ( )

(1)

***𝒒*** *𝑥*1*, 𝑥*2*, 𝑥*3*, 𝑡*

= ***g***(*𝐵*)

*𝑥*1*, 𝑥*2*, 𝑥*3*, 𝑡 ,* ∀

*𝑥*1*, 𝑥*2*, 𝑥*3

∈ *𝛤 , 𝑡* ≥ 0*,*

national laboratories, (*e.g.*, NASA and Sandia National Laboratories),

***𝒒*** (*𝑥 , 𝑥 , 𝑥 ,* 0) = ***g***(0) (*𝑥 , 𝑥 , 𝑥* ) *,* ∀ (*𝑥 , 𝑥 , 𝑥* ) ∈ *𝛺,*

top-tier academic institutions in Europe, U.S., and the Middle East. In

1 2 3

1 2 3

1 2 3

particular, at the end of 2021, the first fully discrete entropy stable

solver of any order on unstructured for compressible CFD, named

where the vectors ***𝒒***, **T** *𝐼*

*𝑚*

*𝑥*

and **T** *𝑉*

*𝑚*

*𝑥*

denote the conserved variables, the

SSDC, was presented in [[10](#_bookmark57)]. The SSDC’s scalable algorithms have been proven to run fast and effectively on hardware ranging from laptops to supercomputers with more than 180,000 processors, such as the Shaheen XC40 supercomputer. However, no studies on the performance of these new solvers on the HPC cloud computing is available in literature.

Here, we provide a comprehensive cost analysis of the SSDC solver for simulating flow problems on AWS EC2. The purpose is to assess the viability of the HPC cluster on the Amazon cloud for solving complex

CFD industry flow problems and identify a set of AWS EC2 instances

***g***(*𝐵*), and the initial condition, ***g***(0), are assumed to be in *𝐿*2(*𝛺*), with inviscid fluxes, and the viscous fluxes, respectively. The boundary data, the further assumption that ***g***(*𝐵*) will be set to coincide with linear,

well-posed boundary conditions, prescribed in such a way that either entropy conservation or entropy stability is achieved. System ([1](#_bookmark5)) is closed with the assumption of the ideal gas model. The conserved variable vector can be written as

***𝒒*** = [*𝜌, 𝜌*T1*, 𝜌*T2*, 𝜌*T3*, 𝜌*£] T*,*

where *𝜌* denotes the density, **U** = [T1*,* T2*,* T3] T is the velocity vector,

£

and is the specific total energy. The inviscid fluxes are given as

*𝑚 𝑚* 1

*𝑚* 2

*𝑚,*2

that deliver the shortest time and the lowest possible price. We use four test cases [[10](#_bookmark57),[14](#_bookmark61),[15](#_bookmark62)]: the turbulent flow over two spheres in tandem at

**T** *𝐼*

*𝑥𝑚*

*𝑚*

= [*𝜌*T *, 𝜌*T T

+ *𝛿𝑚,*1

h*, 𝜌*T T

+ *𝛿* h*,*

a Reynolds number and a Mach number of *𝑅𝑒* = 3*.*9×103

and *𝑀𝑎* = 0*.*1,

*𝜌*T T3

+ *𝛿*

*𝑚,*3

h*, 𝜌*T ] T*,*

respectively; the flow past a delta wing with the experimental sting

*𝑚*

where h is the pressure,  is the specific total enthalpy and *𝛿*

is the

at *𝑅𝑒* = 106 and *𝑀𝑎* = 0*.*07; the NASA juncture flow experiment at

*𝑖,𝑗*

Kronecker delta. The required constituent relations are

*𝑅𝑒* = 2*.*4 × 106 and *𝑀𝑎* = 0*.*189; and the flow past the Imperial front wing at *𝑅𝑒* = 2*.*2*.* × 105 and *𝑀𝑎* = 0*.*036.

 = *𝑐*h f +

1 **U** T**U** *,* h = *𝜌𝑅*f *, 𝑅* = *𝑅𝑢 ,*

2 *𝑀𝑤*

where f is the temperature, *𝑅* is the universal gas constant, *𝑀* is the molecular weight of the gas, and *𝑐*h is the specific heat capacity at

*𝑢*

*𝑤*

constant pressure. Finally, the specific thermodynamic entropy is given

The continuous entropy stability analysis is mimicked by approx- imating the derivatives of the inviscid fluxes as follows [[27](#_bookmark72)], *i.e.*,

as

*𝑅*

( f )

( *𝜌* )

*𝑐𝑝*

*𝜕* ***f****𝐼*

*𝑥𝑚* ≈ 2𝖣*𝐼,𝜅* ◦𝖥

*𝑥*

(***𝒒*** *,* ***𝒒***

) **𝟏***𝜅 ,* (7)

*𝑠* = log

f

*𝛾* − 1

*𝜅 𝜅*

– *𝑅* log

*𝜌*

*, 𝛾* =

*𝑐*

– *𝑅 ,*

*𝜕𝑥𝑚*

*𝑥𝑚 𝑚*

∞ ∞ *𝑝*

where 𝖣*𝐼,𝜅* is a differentiation matrix for the *𝑥*

direction, ◦ denotes the

where f∞ and *𝜌*∞ are the reference temperature and density. The

*𝑥𝑚* ( ) *𝑚*

viscous fluxes **T** *𝑉*

*𝑚*

*𝑥*

]

are given by

Hadamard product, 𝖥*𝑥𝑚*

**𝟏***𝜅* is a vector of ones.

***𝒒****𝜅 ,* ***𝒒****𝜅*

is a two-point flux function matrix, and

[ 3 f

*𝜏* T − *𝜅*

The differentiation matrix 𝖣*𝐼,𝜅* is constructed as

**T** *𝑉*

*𝑥* =

*𝑚*

0*, 𝜏*1*,𝑚, 𝜏*2*,𝑚, 𝜏*3*,𝑚,*

∑

*𝑖*=1

*𝑖,𝑚 𝑖*

*𝜕*

*𝜕𝑥𝑚*

T*,* (2)

𝖣*𝐼,𝜅* ≡ 1 −1 ∑3 (

[J *𝜕𝜉𝑙* ]

*𝑥𝑚*

[

J *𝜕𝜉𝑙* ] )

while the viscous stresses are defined as

𝖩

𝖣

*𝑥𝑚*

2 *𝜅*

*𝑙*=1

*𝜉𝑙*

+

*𝜕𝑥𝑚 𝜅*

*𝜕𝑥𝑚*

𝖣*𝜉𝑙*

*𝜅*

*,* (8)

*𝜏* = *𝜇*

( *𝜕*T

*𝑖* +

*𝜕*T

*𝑗* − *𝛿* 2

∑3 *𝜕*T )

*,* (3)

*𝑛*

where 𝖩*𝜅* denotes the determinant of the discrete Jacobian, 𝖣*𝜉𝑙*

*𝜕 𝜕𝜉*

[ ]

is an

*𝑖,𝑗*

*𝜕𝑥𝑗*

*𝜕𝑥𝑖*

*𝑖,𝑗* 3

*𝑛*=1

*𝜕𝑥𝑛*

SBP operator approximating

*𝜕𝜉*

, while

J *𝑙*

*𝜕𝑥𝑚*

indicates the discrete

*𝜅*

where *𝜇*(f ) is the dynamic viscosity and *𝜅*(f ) is the thermal conduc-

metric terms which must satisfy a discrete version of the geometric

*𝑉 ,𝜅*

conservation law (GCL) constraints [[19](#_bookmark64),[25](#_bookmark70),[28](#_bookmark73)]. In addition, 𝖣*𝑥*1 is the

tivity.

*𝑚*

differentiation matrix, constructed as

vex extension, that when integrated over the physical domain, *𝛺*, The compressible Navier–Stokes equations given in ([1](#_bookmark5)) have a con-

𝖣*𝑉*1 *,𝜅* ≡ 𝖩−1 ∑ 𝖣

[J *𝜕𝜉𝑙* ]

*.* (9)

depends only on the boundary data and negative semi-definite dissi-

*𝑥𝑚*

*𝜅*

*𝑙*=1

3

*𝜉𝑙*

*𝜕𝑥𝑚 𝜅*

pation terms. This convex extension depends on an entropy function,

S

, that is constructed from the thermodynamic entropy as

S

Furthermore, the derivative of the entropy variables, ***w***, is approxi- mated as

= −*𝜌𝑠,*

and provides a mechanism for proving stability in the *𝐿*2

norm. The

***𝜽****𝑎*

= 𝖣*𝑉*2 *,𝜅* ***𝒘***

+ **𝐒𝐀𝐓***𝑉*2 ≈ *𝜕****w*** *.* (10)

*𝜕𝑥𝑎*

entropy variables **M** are an alternative variable set related to the conservative variables via a one-to-one mapping. They are defined

*𝑥𝑎*

*𝜅*

T

The differentiation matrix 𝖣*𝑉*2*,𝜅* in ([10](#_bookmark6)) is defined as

*𝑗*

*𝑥*

∑3 [ ]

in terms of the entropy function S by the relation **M** = *𝜕*S∕*𝜕*****

𝖣*𝑉*2 *,𝜅*

≡ 𝖩−1

J *𝜕𝜉𝑎*

𝖣 *.* (11)

and they are extensively used in the entropy stability proofs of the algorithms used herein; see, for instance, [[16](#_bookmark63)–[19](#_bookmark64)] and the references

*𝑥𝑗*

*𝜅*

*𝑎*=1

*𝜕𝑥𝑗*

[

*𝜉𝑎*

*𝜅*

] [ ]

therein. In addition, they simultaneously symmetrize the inviscid and the viscous flux Jacobians in all three spatial directions. Following

The metric terms

J *𝜕𝜉𝑙*

*𝜕𝑥𝑚*

and

*𝜅*

J *𝜕𝜉𝑎*

*𝜕𝑥𝑗*

can be computed in different

*𝜅*

the analysis described in [[16](#_bookmark63),[20](#_bookmark65)], we multiply Eqs. ([1](#_bookmark5)) by the (local) entropy variables **M** and arrive at the integral form of the (scalar) entropy equation

ways. In the SSDC solver, these metrics terms are computed based upon

conforming interfaces are given in [[25](#_bookmark70)]; for *𝑝*- and *ℎ𝑝*-nonconforming the optimization procedure of Crean et al. [[28](#_bookmark73)]. Algorithmic details for

interfaces, the interested reader is referred to [[19](#_bookmark64),[24](#_bookmark69)].

d d*𝑡*

*𝑆*d*𝛺* = d *𝜂* ≤

*𝛺* d*𝑡*

∫

∑3

*𝑚*=1 *𝛤*

∫

(**M** T**T** *𝑉*

*𝑥*−

)

*𝑥 𝑛𝑥* d*𝛤* − *𝐷𝑇 ,* (4)

T

The entropy function of the semidiscretization ([6](#_bookmark9)) mimics closely Eq. ([4](#_bookmark7)). In fact, following closely the entropy stability analysis, the SBP operators and their equivalent telescoping forms yield [[16](#_bookmark63),[19](#_bookmark64),[22](#_bookmark67),[24](#_bookmark69)–

where *𝑛𝑥* is the *𝑚*th component of the outward facing unit normal to *𝛤*

and

[26](#_bookmark71)]

*𝑑* **𝟏***⊤̂*𝖯 **𝐒** = *𝑑 𝜂* = **𝐁𝐓** − **𝐃𝐓** + **Υ***,* (12)

∑3 ( *𝜕***M** )*⊤*

∫

*𝜕***M**

*𝑑𝑡*

*𝑑𝑡*

*𝐷𝑇* =

*𝑚,𝑗*=1 *𝛺*

*𝜕𝑥𝑚*

𝖢*𝑚,𝑗 𝜕𝑥* d*𝛺.* (5)

which is the semi-discrete analog of ([4](#_bookmark7)). Here **𝐁𝐓** is the discrete

boundary term (i.e., the discrete version of the first integral term on

More details about the continuous entropy analysis can be found [[21](#_bookmark66)– [23](#_bookmark68)].

*𝑗*

* 1. *Spatial discretization of the compressible Navier–Stokes equations*

The details of the algorithm for conforming and nonconforming interfaces can be found in [[10](#_bookmark57),[16](#_bookmark63),[19](#_bookmark64),[24](#_bookmark69)–[26](#_bookmark71)]. Herein, we summarized the main steps of the algorithm for conforming interfaces.

main *𝛺* is divided into *𝐾* nonoverlapping elements. Then, following the To approximate the compressible Navier–Stokes equations, the do-

Eqs. ([A.1](#_bookmark40)), on the *𝜅*th element, the generic entropy stable discretization procedure outlined for the spatial discretization of convection–diffusion

of ([1](#_bookmark5)) reads

the right-hand side of ([4](#_bookmark7))), **𝐃𝐓** is the discrete dissipation term (i.e., the

discrete version of the second term on the right-hand side of ([4](#_bookmark7))) and

completeness, we note that the matrix *̂*𝖯 may be thought of as the **Υ** enforces interface coupling and boundary conditions [[16](#_bookmark63),[20](#_bookmark65)]. For mass matrix in the context of the discontinuous Galerkin finite element

method.

In our framework, the boundary conditions necessary to close sys- tem ([1](#_bookmark5)) preserve the entropy stability of the interior operators described above. Precisely, solid inviscid and viscous wall boundary conditions are imposed as described in [[20](#_bookmark65),[29](#_bookmark74)] whereas, for the far-field, we use the approach described in [[30](#_bookmark75)]. The implementation of boundary conditions for a general framework (not necessarily entropy stable) can be found in [[31](#_bookmark76)].

d***𝒒****𝜅* d*𝑡*

+ ∑3

*𝑚*=1

2𝖣*𝐼,𝜅* ◦𝖥*𝑥*

(***𝒒*** *,* ***𝒒***

) **𝟏***𝜅* =

∑3

*𝑚,𝑗*=1

*𝑉*1 *,𝜅* [

*𝑥𝑚*

𝖣

𝖢*𝑚,𝑗* ]

***𝜽****𝑗*

(6)

* 1. *Temporal discretization of the compressible Navier–Stokes equations*

*𝑚*

+ **𝐒𝐀𝐓***𝐼* + **𝐒𝐀𝐓***𝑉* + **𝐝𝐢𝐬𝐬***𝐼* + **𝐝𝐢𝐬𝐬***𝑉*1 *,*

*𝑥𝑚*

*𝜅 𝜅*

where ***𝒒****𝜅* is the numerical solution vector at the mesh nodes, while **𝐝𝐢𝐬𝐬***𝐼* and **𝐝𝐢𝐬𝐬***𝑉* denote the interface dissipation contributions for the inviscid

and viscous parts of the equations, respectively.

A general (explicit or implicit) *𝑠*-stage Runge–Kutta (RK) scheme

is composed of a matrix *𝐴* of dimensions *𝑠* × *𝑠* and two vectors *𝑏* and can be concisely encapsulated using its Butcher tableau [[32](#_bookmark77)], which

*𝑐* of length *𝑠*. The basic idea of the relaxation procedure is to enforce

conservation, dissipation, or other solution properties with respect to

a convex functional by scaling the weights *𝑏𝑖* of the RK method by a real-value parameter *𝛾̃*. Hence, the time step from *𝑢𝑛* ≈ *𝑢*(*𝑡𝑛*) is given by

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Model | # CPUs | Memory (GiB) | Network bandwidth (Gbps) |
| Intel Xeon Platinum 8000 | 40 | 350 | Up to 100 |
| Intel Xeon Platinum 8276M | 40 | 350 | Up to 100 |
| (13) | AMD EPYC 7642 | 128 | 375 | Up to 100 |

∑*𝑠*

**Table 1**

Details of the Ibex’s architectures.

*𝑦𝑖* = *𝑢𝑛* + *𝛥𝑡 𝑎𝑖𝑗 𝑓* (*𝑡𝑛* + *𝑐𝑗 𝛥𝑡, 𝑦𝑗* )*,*

*𝑗*=1

∑*𝑠*

*𝑢𝑛*+1 = *𝑢𝑛* + *𝛾̃𝑛𝛥𝑡 𝑏𝑖𝑓* (*𝑡𝑛* + *𝑐𝑖𝛥𝑡, 𝑦𝑖*)*,*

*𝛾̃*

*𝑖*=1

**Table 2**

Amazon EC2 instances feature details.

where the stage values of the RK method are denoted by *𝑦𝑖*. The

parameter *𝛾̃𝑛* is computed using the global relaxation procedure, *i.e.*, *𝛾̃𝑛* is a root of a global nonlinear algebraic equation for *𝜂* [[33](#_bookmark78)]. We recover the step of the classic RK method if *𝛾̃𝑛* = 1.

In the SSDC solver, global relaxation RK schemes are used to in- tegrate in time systems of ODEs ([6](#_bookmark9)) such that Eq. ([12](#_bookmark8)) is fulfilled. For performance and robustness reasons, we select the secant method as the root finding algorithm for computing the relaxation parameter

*𝛾̃𝑛* [[14](#_bookmark61),[34](#_bookmark79)].

# Software implementation

The SSDC solver used in this work is being developed in the Ad- vanced Algorithms and Numerical Simulations Laboratory (AANSLab), which is part of the Extreme Computing Research Center (ECRC) at King Abdullah University of Science and Technology (KAUST). The SSDC framework is built on top of the highly-scalable Portable and Ex- tensible Toolkit for Scientific computing (PETSc) [[35](#_bookmark80)], its mesh topol- ogy abstraction (DMPlex) [[36](#_bookmark81)], and its scalable differential–algebraic

equation solver components [[37](#_bookmark82)]. The spatial discretization features *ℎ𝑝*-

adaptive capabilities on unstructured quadrilateral/hexahedral meshes.

Support for nonconforming meshes relies on the p4est software library [[38](#_bookmark83),[39](#_bookmark84)] and its bridge to PETSc’s DMPlex [[40](#_bookmark85)]. Leveraging the capa- bilities of the PETSc library allows support for different mesh formats including fluent, Exodus II, CGNS and GMSH. Triangle/tetrahedral meshes are converted on the fly into quadrilateral/hexahedral ele- ments; uniform and non-uniform mesh refinements algorithms are also available.

# HPC cluster’s architectures

Amazon EC2 offers a variety of instance types that are suitable for a wide range of applications. It is possible to mix and match CPU, memory, storage, and networking resources by using different instance types. Depending on the workload, we can choose from a variety of instance sizes available for each type of instance. AWS Cloud provides highly flexible computing platforms that are suited for performing HPC applications in terms of resource availability and configurability. CFD usually necessitates a large amount of computer resources, and its software architecture makes it suited for parallel processing; this requires the use of an HPC or cloud computing infrastructure. In order to deal with parallel computation needs, an adequate resource management system is also necessary. AWS ParallelCluster is one of the possible solutions to the aforementioned requirements. Here, we will briefly describe the compute environments of Amazon EC2 cloud and on-premise resource Ibex cluster.

* 1. *Ibex cluster*

Ibex is a heterogeneous cluster composed of nodes with various CPU architectures and an assortment of GPUs hosted at KAUST. Ibex is made up of more than 400 nodes that are constantly monitored by the systems team. This heterogeneous cluster has a mix of CPU architectures, including Intel Xeon Platinum 8000 series (Skylake-SP) 1st generation, Intel Xeon Platinum 8276M (Cascade Lake-SP) 1st gen- eration, AMD EPYC 7642 (AMD Rome), and various GPUs nodes with P100s, V100s, GTX1080Ti, and RTX2080Ti, and with variable amounts

|  |  |  |  |
| --- | --- | --- | --- |
| Model | # vCPUs | Memory (GiB) | Network bandwidth (Gbps) |
| c5d.4xlarge | 16 | 32 | Up to 10 |
| c5d.9xlarge | 36 | 72 | 10 |
| c5a.4xlarge | 16 | 32 | Up to 10 |
| c5a.8xlarge | 32 | 64 | 10 |
| c6 g.4xlarge | 16 | 32 | Up to 10 |
| c6 g.8xlarge | 32 | 64 | 12 |

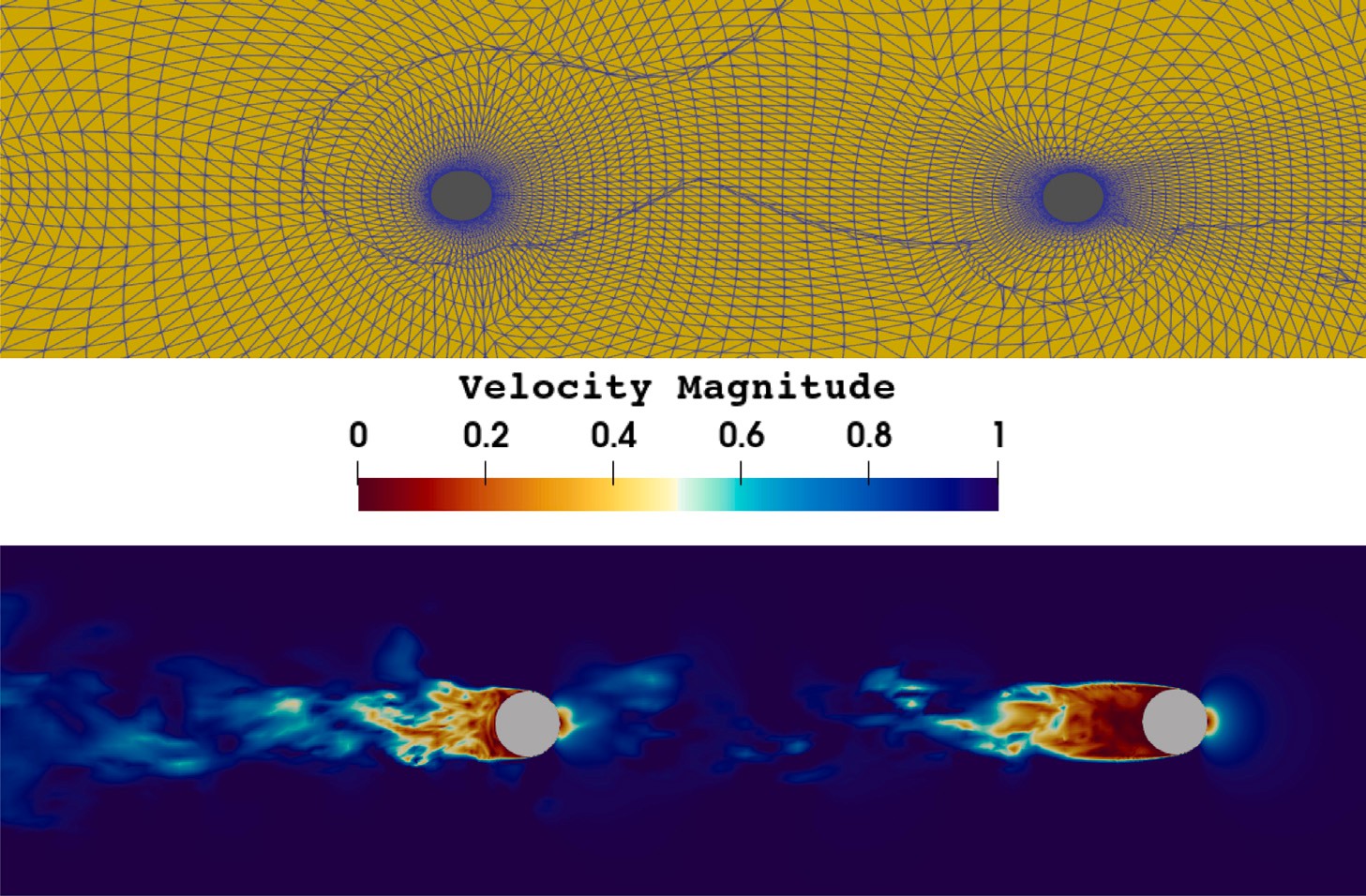
of RAM ranging from 360 GB to 700 GB. The network connectivity on Ibex depends on InfiniBand HDR Director Switch. It has the following features and benefits: Mellanox HDR Infiniband is capable of 200 Gbps, it uses in compute and storage nodes with HDR-100 at 100 Gbps speed. [Table](#_bookmark10) [1](#_bookmark10) summarizes the main features of the Ibex’s architectures used in this work. We highlight that Ibex has a daily occupancy of about 75%, with Intel architectures in high demand.

* 1. *Amazon ParallelCluster*

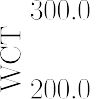
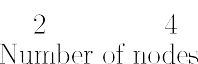
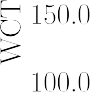
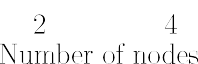
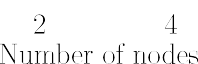
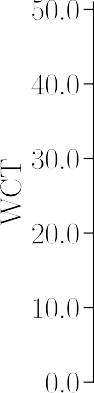
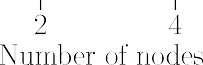
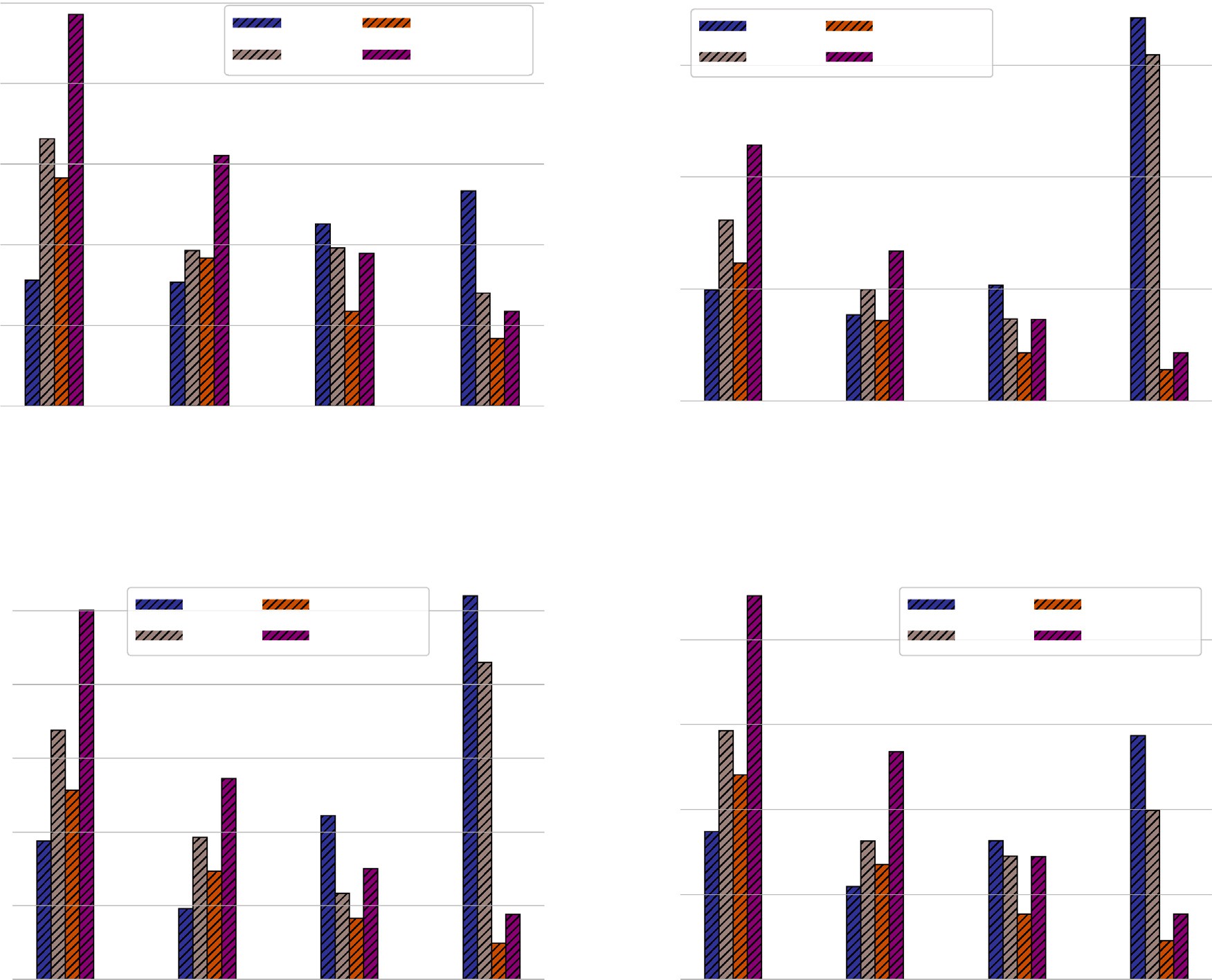
AWS ParallelCluster is an open source cluster management tool that is supported by AWS to deploy and manage HPC clusters in the AWS cloud. It enables the operation of HPC clusters in the AWS Cloud Environment and provides a wide range of configuration options. AWS ParallelCluster offers a variety of batch schedulers for manag- ing nodes, resources, and parallel workloads. Users can use them to run large-scale scientific and engineering tasks. AWS cloud has many data centers throughout the world. In this work, all the experiments were conducted on Amazon’s cloud data center in US East region in Northern Virginia. Our assessments cover the possible instance types which are similar to those available on KAUST’s Ibex cluster described in Section [4.1](#_bookmark14) with in addition one Arm architecture as described at the end of this section. Compute Optimized instances are suited for compute intensive applications that benefit from powerful CPUs. In this work, we have built AWS ParallelCluster and we included the following compute-optimized instances: C5d, C5a, and C6 g instances. This family’s instances are ideal for HPC. C5d will run on the Intel Xeon Scalable Processor 2nd generation or the Intel Xeon Platinum 8000 series (Skylake-SP) 1st generation with a sustained turbo frequency of up to 3.4 GHz and a single core turbo frequency of up to 3.5 GHz using Intel Turbo Boost Technology at launch. C5a instances provide the best x86 price/performance for a wide range of compute-intensive tasks, using 2nd generation 3.3 GHz AMD EPYC 7002 series processors built on a 7 nm technology node for greater efficiency. C5a instances can deliver up to 20 Gbps of network capacity as well as 9.5 Gbps of dedicated bandwidth to Amazon. EC2 C6 g instances are powered by AWS Graviton2 processors based on the Arm architecture. [Table](#_bookmark11) [2](#_bookmark11) shows the instances feature details.

# Performance of the SSDC solver on Ibex and AWS EC2 clusters

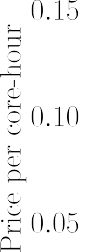
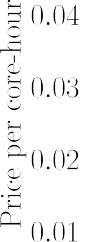
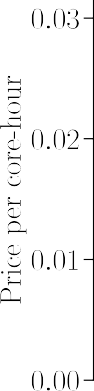
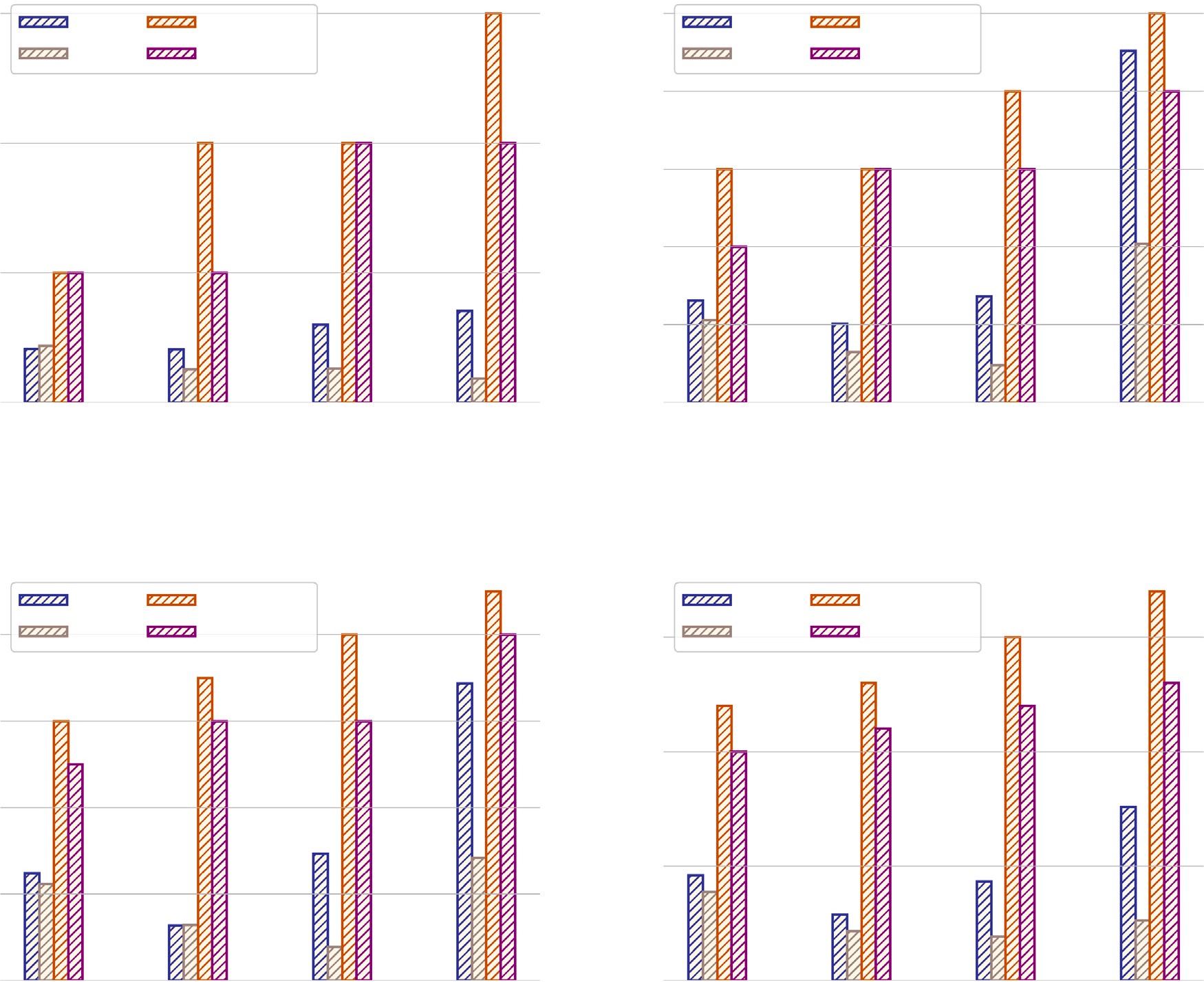
The purpose of this study is to analyze the performance of the SSDC solver on Ibex and Amazon EC2 clusters for complex flow problems and provide the current cost per core hour. The default number of CPU cores for an EC2 instance depends on its type. We only allowed the instance to use the available physical CPU cores. This means we disabled hyperthreading. In the following sections, we report the arithmetic average of the wall-clock time (WCT) in seconds of three independent runs and the associated estimated cost in United States Dollars (USD).



**/ig. 1.** Upper panel: Illustration of the mesh. Lower panel: Contour plots of the velocity magnitude for the flow past two identical spheres in tandem at time *𝑡* = 100.



**/ig. 2.** Flow past two spheres in tandem: Wall-clock time in seconds for each simulation against the number of nodes for Intel CPU architectures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 3.** Flow past two spheres in tandem: Cost performance in USD of Intel CPU architectures.

* 1. *Flow past two spheres in tandem*

D, held fixed in a rectangular domain located at a separation distance of We simulate the flow past two equally sized spheres with a diameter 10D [[41](#_bookmark86)]. The Reynolds and Mach numbers are set to *𝑅𝑒𝐷* = ∞D∕*𝜈* = 104 and *𝑀𝑎*∞ = 0*.*1, respectively. The Prandtl number is set to *𝑃 𝑟* = 0*.*7.

T

The quite upstream flow conditions of the first sphere are used to define the similarity parameters, e.g., ∞ is the free-stream velocity.

T

Regardless of the geometry’s simplicity, capturing the flow in this regime is relatively difficult. The relevance of such flows around sev- eral bodies, specifically around two spheres, is considered significant in many practical applications, as it allows a better understanding of the effect of the wake behind a leading bluff body on the flow around a trailing one, for instance. A non-exhaustive list of important applications ranging from industrial fluidized beds to bio-reactors, to the combustion of aerosols, could be liquid–gas two-phase flows [[42](#_bookmark87)], suppression of icing on the solid surface [[43](#_bookmark88)], and oil droplets [[44](#_bookmark89)]. The complexity of the flow fields is shown in the lower panel of [Fig.](#_bookmark16) [1](#_bookmark16)

by plotting the velocity magnitude at time *𝑡* = 100.

We perform the numerical simulations using one of the grids TandemSpheresHexMesh2Pm provided by Steve Karman of Point- wise for the HiOCFD5 [[41](#_bookmark86)]. An illustration of the grid structure is shown in the upper panel of [Fig.](#_bookmark16) [1](#_bookmark16). In this study, we use this mesh

in combination with a solution polynomial order of *𝑝* = 10 (leading to

degrees of freedom (DOFs) of ≈ 2*.*616 × 107. a formally eleventh-order accurate scheme), yielding a total number of

[Figs.](#_bookmark17) [2](#_bookmark17) and [4](#_bookmark19) show the WCT in seconds for the Ibex cluster and the

cluster, we simulated the flow problems using 40 (dark blue — Ibex I) EC2 instances using various CPU architectures. On the on-premises Ibex and 20 (gray — Ibex II) physical cores. Moreover, each processor is

responsible for one MPI thread. The wall-clock time for an on-premises cluster based on Intel’s Cascade Lake architecture grows proportionally

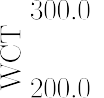
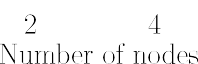
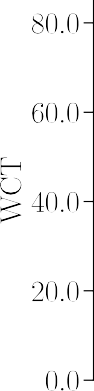
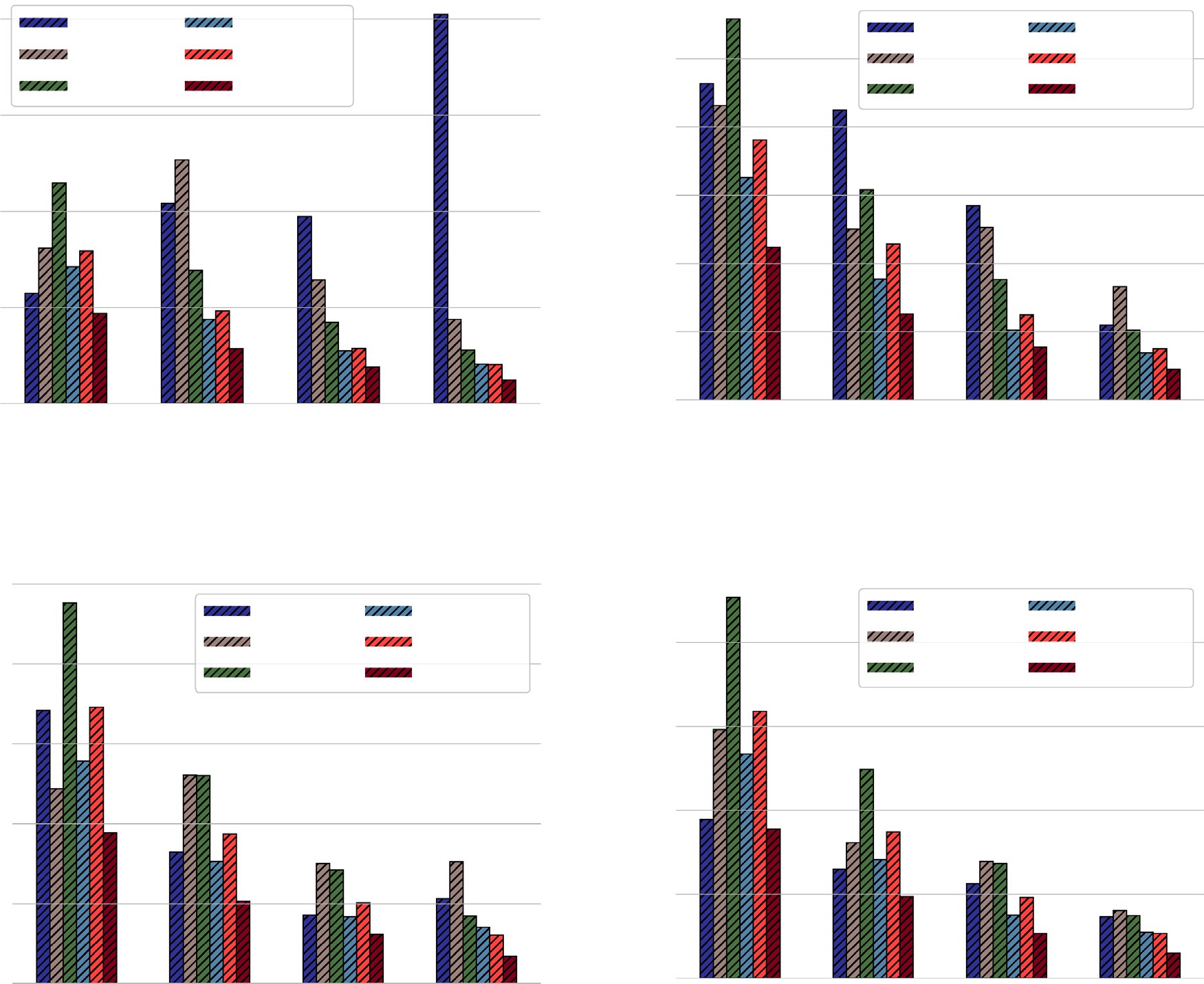
with the node count, as shown in [Fig.](#_bookmark17) [2](#_bookmark17). This pattern holds in the 40

and 20 physical core setups. On-premise cluster performance degrades

for several reasons, including the problem’s size and the fact that the

nodes are non-exclusive, with a daily occupancy of the Ibex cluster of approximately 75%. Thus, the more computing nodes, (1) the greater the likelihood that computing resources will be shared with other users, and (2) the more partitioned the job, the more likely it is that network bandwidth will be shared. We highlight that this behavior can also be observed in the case of exclusive access to the nodes. In fact, for a ‘‘sufficiently’’ large number of CPU cores count and hence, sufficiently smaller local problems, communication between partitions cannot be hidden behind computations. Thus, the solver’s performance degrades and departs from the ideal behavior.

For the EC2 c5d instances and all orders of accuracy, the simulations are speeded up as the number of CPUs increases. Moving from one to



**/ig. 4.** Flow past two spheres in tandem: Wall-clock time in seconds for each simulation against the number of nodes for AMD & Arm CPU architectures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

eight computing nodes, in particular, results in a speed-up factor of about 5.6 for both cases (8 being the perfect scaling factor). Overall, the c5d.9xlarge EC2 instance delivers the shortest time-to-solution. Notably, although the on-premises cluster’s wall-clock time increases as the number of compute nodes increases, up to two Ibex nodes running

40 and 20 MPI threads are still faster than EC2 instances. The executions

on the on-premises cluster with 20 MPI threads are marginally faster

or comparable to the executions on the c5d.4xlarge instance with four

computing nodes. Therefore, it seems that utilizing half of a node’s physically available cores has a favorable effect on wall-clock time. In fact, only half of the cores utilize the shared intra-node network, and more crucially, the workload per core permits better communications to be hidden behind computations.

As observed from [Fig.](#_bookmark17) [2](#_bookmark17), the results are favorable for simulations conducted on 8 EC2 nodes. In particular, the c5d.9xlarge EC2 instance delivers the results in the least amount of time. However, in terms of cost, [Fig.](#_bookmark18) [3](#_bookmark18) leads to a different conclusion: For any number of nodes and order of accuracy tested, it is cheaper to run on the Ibex cluster.

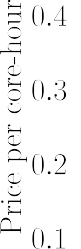
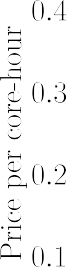
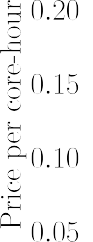
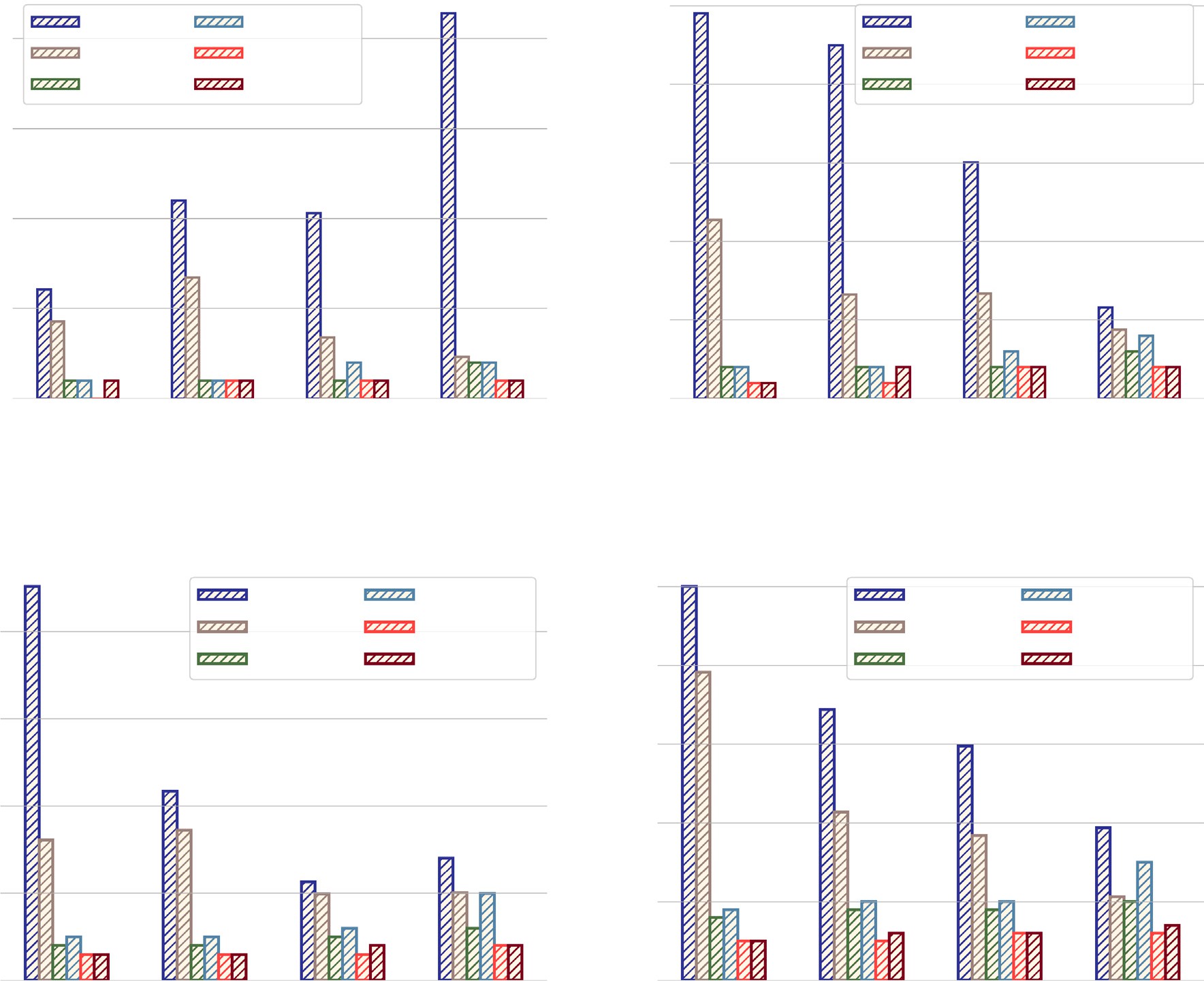
Specifically, the simulations run with 20 physical cores cost at least

two times less than those performed on AWS EC2 instances. Except for

the third-order accurate solver on 8 nodes, running in an on-premises

of available physical cores (*i.e.*, 40). cluster is always advantageous, even when using the maximum number

For the on-premises cluster with the AMD Rome architecture, the performance results are shown in [Fig.](#_bookmark19) [4](#_bookmark19). We observe that for the second- order accurate solver, the wall-clock time oscillates and does increase drastically when 8 nodes, and 40 cores per node are used. This is again the combined effect of the problem size and the network connectivity shared with other users — a common scenario for a parallel cluster. On the contrary, for the third-, fourth-, and fifth-order accurate solvers, we observe that by increasing the number of nodes, the wall-clock time decreases substantially. Increasing the number of nodes of EC2 AMD EPYC (*i.e.*, c5a instances) also leads to a reduction in time- to-solution. While the WCT decreases by around 35%-to-40% when moving from the c5a.4xlarge to the c5a.8xlarge. Thus, more on-chip cores are beneficial for this problem size. [Fig.](#_bookmark19) [4](#_bookmark19) also exhibits the performance of the Arm-based AWS Graviton2 processors (*i.e.*, c6 g instances). As we can observe, the c6 g instances perform well and deliver the least wall-clock time. Additionally, the time to solution is roughly halved when the nodes are doubled—indicating that a nearly perfect scaling is achieved. Furthermore, using four Arm nodes, the time-to-solution is nearly identical to that required by eight AMD EPYC nodes. Nevertheless, as shown in [Fig.](#_bookmark19) [4](#_bookmark19), the WCT on the Arm- based architecture c6 g.8xlarge and the Ibex cluster with eight nodes using 40 MPI processes per node are almost identical. Thus, the AMD



**/ig. 5.** Flow past two spheres in tandem: Cost performance in USD of AMD & Arm CPU architectures.

Rome nodes of the on-premises cluster can deliver similar performance compared to the c6 g.8xlarge instance.

In [Figs.](#_bookmark20) [5](#_bookmark20), we show the cost of simulating the test case with AMD and Arm architectures. Compared to the AWS ParallelCluster, the Ibex cluster offers significantly more affordable computations. While between all AWS EC2 instances, the Arm nodes are the least expensive for c6 g.4xlarge and c6 g.8xlarge. Finally, it is worth noting that the cost and performance of the EC2 c6 g instances are similar to those of the AWS EC2 c5d instances for all orders of accuracy.

* 1. *Flow around delta wing*

We study the flow around a 65° swept delta wing. We use the geometry reported by Hummel and Redeker [[45](#_bookmark90)] for the Second In-

radius leading edge configuration, *𝑟𝐿𝐸* ∕*𝑐̄* = 0*.*0015, where *𝑐̄* = 0*.*653 m. ternational Vortex Flow Experiment. In this work, we use the medium The delta wing has a mean aerodynamic chord of 𝓁 = 0*.*667 m, a root chord length of *𝑐𝑟* = 1*.*47𝓁, and a wing span of *𝑏* = 1*.*37𝓁. Furthermore,

coordinate system is positioned at the delta wing’s apex with the *𝑥*1 the central region is flat, and it has no twist or camber. A Cartesian coordinate pointing downstream, the *𝑥*2 coordinate pointing in a span- wise direction, and the *𝑥*3 coordinate perpendicular to the flat plate. We consider the sting as part of the setup up to the position *𝑥*1∕*𝑐𝑟* = 1*.*758.

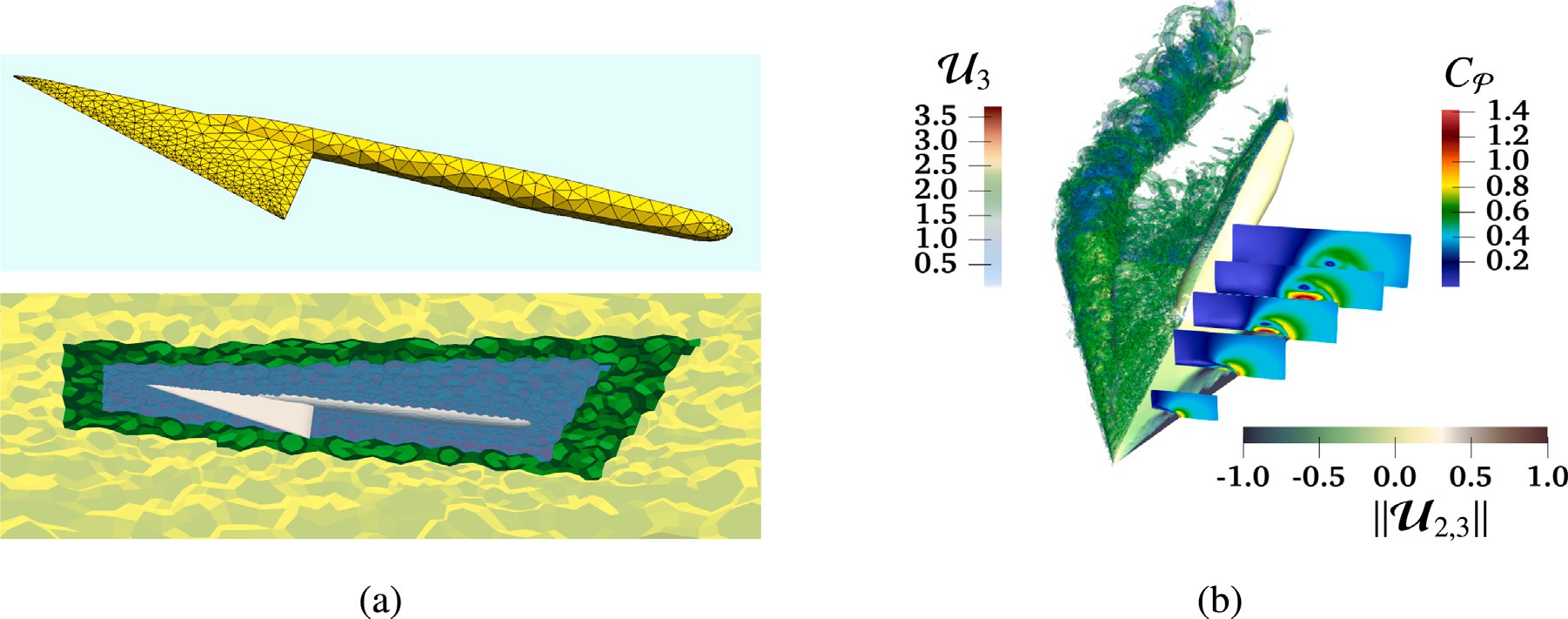
The grids consist of ≈ 9*.*209 × 104 hexahedral cells. As shown in

a different solution polynomial degree, *𝑝*. Given the degree of the [6](#_bookmark21)(a), the grid is divided into three blocks, where each block is assigned

is ≈ 1*.*435 × 107. The simulations are carried out for an angle of attack solution and the number of cells in each block, the number of DOFs of AoA = 13°, a Mach number *𝑀𝑎* = 0*.*07 and a Reynolds number of

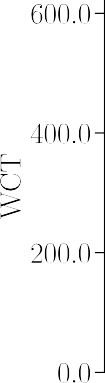
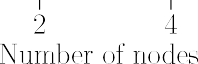
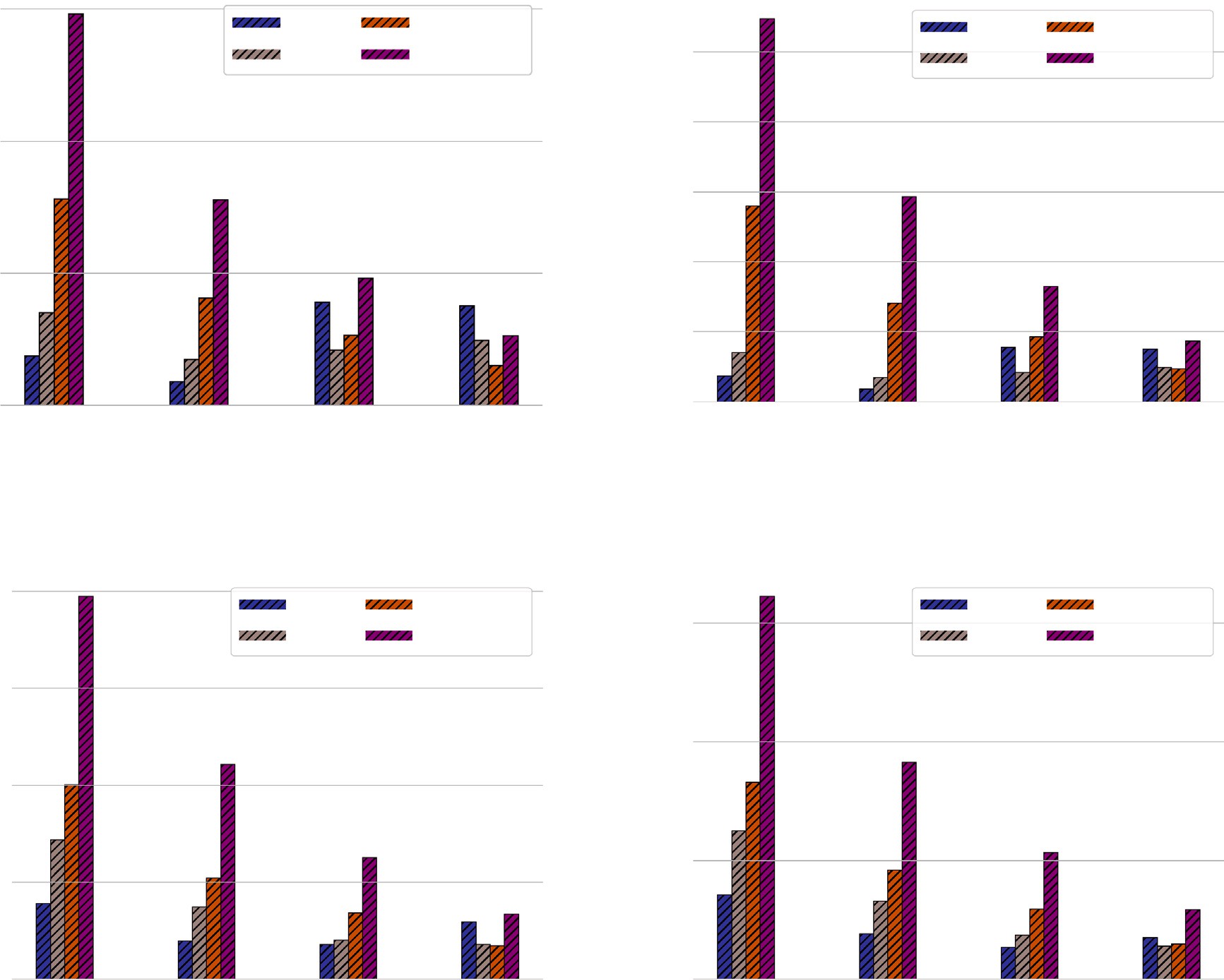
*𝑅𝑒* = 106, based on the mean aerodynamic chord.

The performance of the simulation for both on-premises cluster Ibex using Intel Cascade Lake architecture and AWS ParallelCluster using EC2 c5d instances are present in [Fig.](#_bookmark22) [7](#_bookmark22). The performance of the on-premise cluster is degraded as the number of cores increases. This is because of the interplays of two factors: the high occupancy and non-exclusivity of the Intel Cascade nodes on the Ibex cluster, and the considerable increase of the fraction of the communication time on the overall computational time for smaller and smaller local problems (communication cannot be hidden behind computations). When the size problem becomes ‘‘sufficiently’’ large, we observe that an increment of the number of nodes corresponds to a decrease in time- to-solution, as shown in [Fig.](#_bookmark23) [7(d)](#_bookmark23). However, the performance degrades again for eight nodes, where we observe a growth of the time-to- solution. In contrast, for the AWS EC2 c5d instances, the simulation is speed-up as the number of nodes increases for all orders of accuracy.

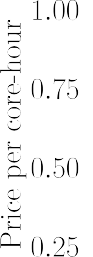
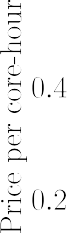
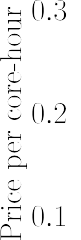
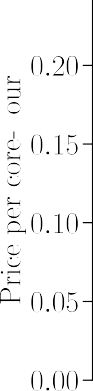
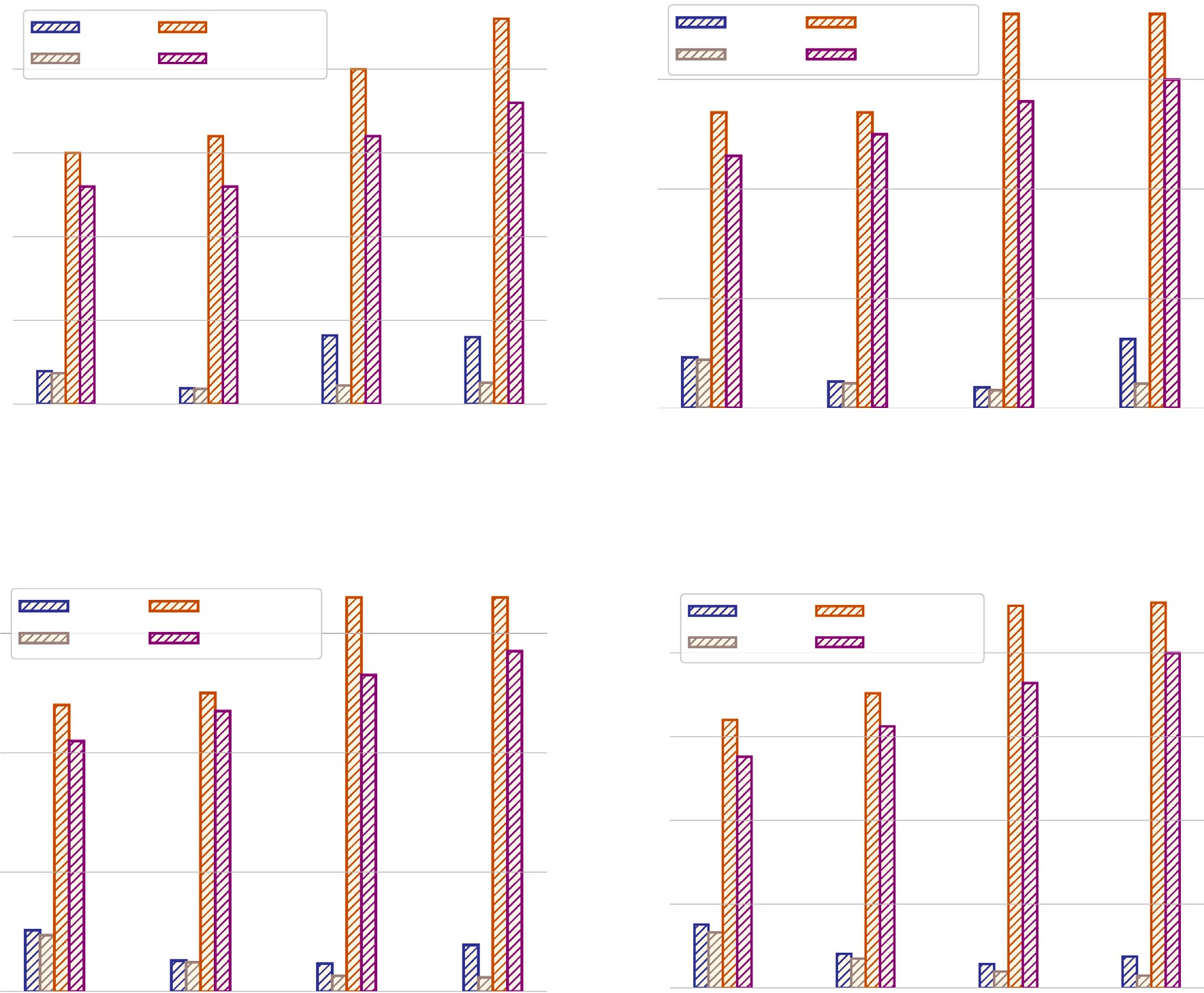


**/ig. 6.** (a) Geometry (top) and solution polynomial degree distribution (bottom) for the 65° swept delta wing test case; *𝑝* = 2 in the far-field region (yellow), *𝑝* = 5 in the region surrounding the delta wing and its support (blue), and *𝑝* = 3 elsewhere (green). (b) Average flow field past the 65° swept delta wing: the *𝑄*-criterion colored by the normalized velocity magnitude (left) and mean axial velocity (right) at *𝑥*1∕*𝑐𝑟* = 0*.*2*,* 0*.*4*,* 0*.*6*,* 0*.*8, and 0.95; the wing surface is colored using the time-averaged pressure coefficient. (For

interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 7.** Flow around a delta wing: Wall-clock time in seconds for each simulation against the number of nodes for Intel CPU architectures.



**/ig. 8.** Flow around a delta wing: Cost performance in USD of Intel CPU architectures.

In particular, a speed-up factor of about 1.8 is delivered for both AWS EC2 c5d instances.

[Fig.](#_bookmark24) [8](#_bookmark24) shows the cost analysis of running on Ibex and the c5d instances. The runs performed with the on-premises cluster using 40 and 20 MPI threads show better cost-efficiency than the AWS EC2 c5d instances. In fact, Ibex provides up to 70% better pricing performance than c5d instances, regardless of solver accuracy and the number of nodes.

As done for the flow past the two spheres in tandem, we also study the performance of the solver and the cost of each simulation for the AMD and Arm architectures. [Fig.](#_bookmark25) [9](#_bookmark25) shows the wall-clock time in seconds for the on-premise cluster with the AMD Rome architecture, the EC2 AMD c5a instances, and the EC2 Arm c6 g instances. The performance results of the Ibex and AWS clusters is mostly matching the results obtained for the previous test case, *i.e.,* the wall-clock time decreases substantially by increasing the number of nodes. In particular, by doubling the number of nodes we observe a speed-up factor of about

1.7 for all the runs. As for the previous application, moving from the c5a.4xlarge to the c5a.8xlarge instances lead to a reduction in the wall-clock time by around 35%-to-40%. Furthermore, the smallest wall- clock time is obtained using the EC2 Arm c6 g instances, for all orders of accuracy. In addition, almost perfect scaling is observed. Doubling

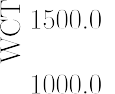
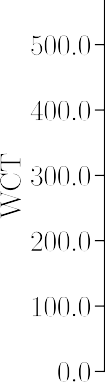
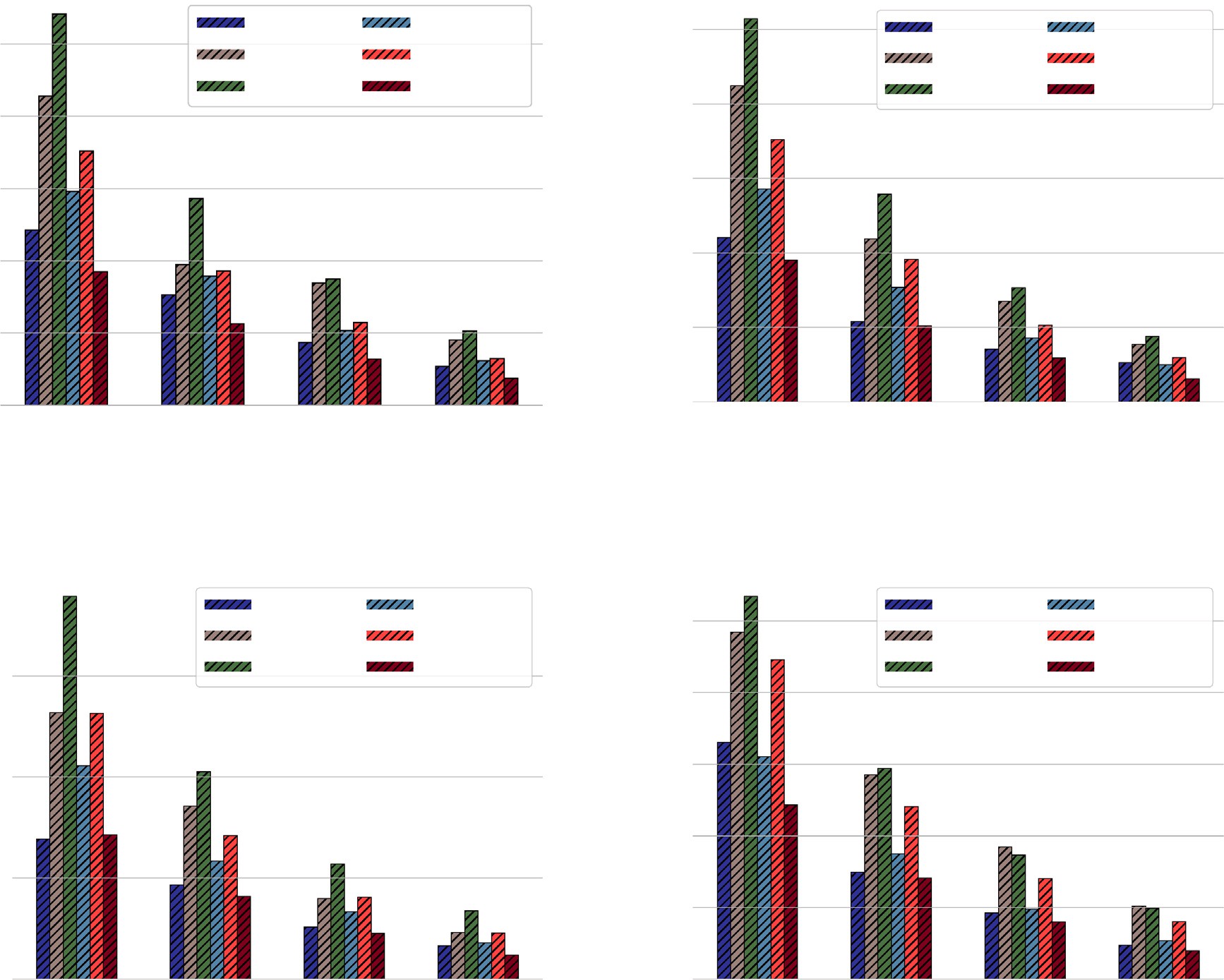
the nodes by switching from c6 g.4xlarge to c6 g.8xlarge leads to halves the solution time.

A detailed cost analysis is shown in [Fig.](#_bookmark26) [10](#_bookmark26). Among the EC2 in- stances, the compute-optimized c6 g instances perform better than the c5a instances for all the number of nodes and order of accuracy. However, on the on-premises cluster, using the largest number of physical cores possible is always the most convenient solution.

In the next section, we further assess the performance of the solver on-premises cluster Ibex and the AWS ParallelCluster by considering two more complex industrially-relevant flow problems. Precisely, we will simulate the NASA junction flow experiment and the flow past a Formula ([1](#_bookmark5)) front wing. The accurate simulation of these indus- trial problems via large eddy simulation (LES) is representative of the type of simulations that exascale will allow performing in a 24- hour turnaround, a typical requirement for industry standards (see, for instance, [[46](#_bookmark91)]). In our context, we use these test cases to explore the influence of the problem size on the performance of both clusters.

* 1. *NASA juncture flow experiment*

In this section, we simulate the NASA juncture configuration. This test case has a wing based on the DLR-F6 geometry and is equipped



**/ig. 9.** Flow around a delta wing: Wall-clock time in seconds for each simulation against the number of nodes for AMD & Arm CPU architectures.

with a leading edge horn to reduce the effect of the horseshoe vor- tex around the wing-fuselage juncture [[47](#_bookmark92)]. The Reynolds and Mach

numbers based on the freestream conditions are 2*.*4 × 106 and 0.189,

respectively. The angle of attack is −2*.*5◦. We perform simulations by

neglecting the sting and the mast. An overview of the mesh is shown

represents a different degree of approximation (*𝑝*) for the solution field. in [Fig.](#_bookmark28) [11](#_bookmark28)(a). The grid is broken up into three blocks, each of which Specifically, we utilize *𝑝* = 1 in the far-field region (dark green), *𝑝* = 3 in the region surrounding the model (dark orange), and *𝑝* = 2 in the

is ≈ 6*.*762 × 105, and the number of DOFs is ≈ 4*.*091 × 107. In [Fig.](#_bookmark28) [11](#_bookmark28)(b), remaining part (dark yellow). The total number of hexahedral elements

instantaneous velocity, *i.e.*, 1∕| ∞|. we plot the Q-criterion using isocontours colored by normalizing the

T T

In [Fig.](#_bookmark29) [12](#_bookmark29), we show the WTC for different numbers of nodes for the Ibex cluster and the AWS EC2 instances. For all the type of nodes except the Intel Cascade Lake, doubling the number of nodes yields an efficiency of about 95%. For the Intel Cascade Lake architecture with 40 MPI threads, moving from one to four nodes leads to a rapid decrease in efficiency. Eventually, for eight nodes, the time to solution increases. The fastest time-to-solution is delivered by the Cascade Lake with 20 MPI threads per node and the c6g8xlarge AWS EC2 instance. In terms of cost, the on-premise cluster simulations done on Intel architectures are much cheaper than the AWS EC2 cd5 instances, as illustrated in [Fig.](#_bookmark30) [13](#_bookmark30). For the AMD-Ibex nodes and the c5a and c6 g instances, the on-premises cluster still delivers the smallest cost.

* 1. *Flow past a Formula* ([1](#_bookmark5)) *front wing*

Here, we consider the flow past a Formula ([1](#_bookmark5)) front wing [[48](#_bookmark93)]. We refer to this test case as the Imperial Front Wing, based on the front

by *ℎ* the distance between the ground and the lowest part of the front wing and endplate design of the McLaren 17D race car [[49](#_bookmark94)]. We denote wing endplate and by *𝑐* the chord length of the main element. The

angle of 1*.*094◦. Here we use *ℎ*∕*𝑐* = 0*.*36 which can be considered as position of the wing in the tunnel is further characterized by a pitch

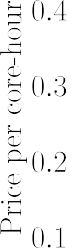
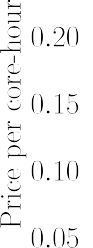
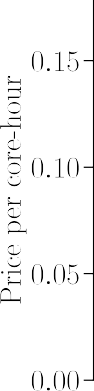
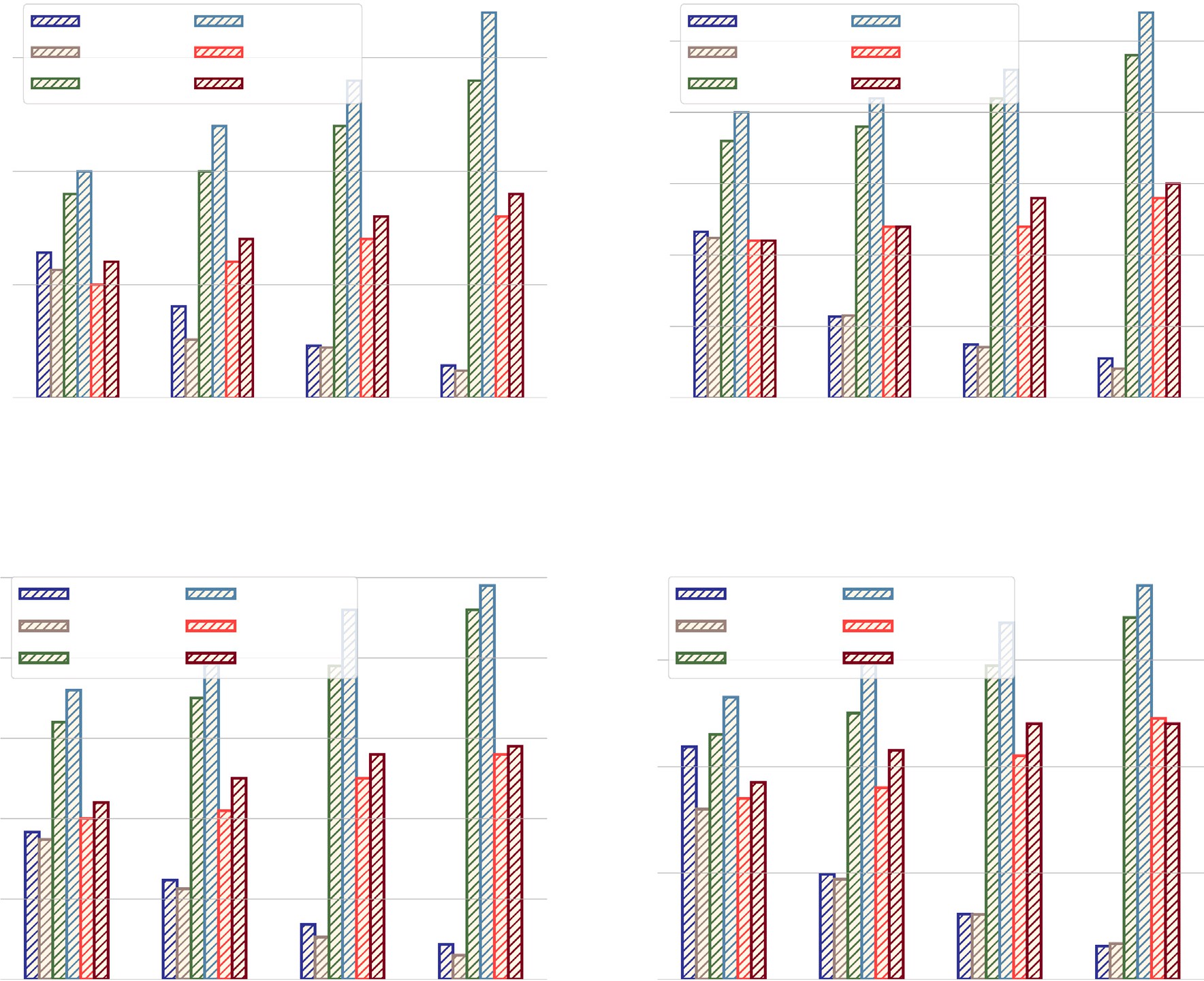
higher loads on the wing. The corresponding Reynolds number is *𝑅𝑒* = a relatively low front ride height, with high ground effect and hence

2*.*2 × 105, based on the main element chord *𝑐* of 250 mm and a free stream velocity *𝑈* of 25 m∕s. The Mach number is set to *𝑀𝑎* = 0*.*036.

This corresponds to a practically incompressible flow.

The computational domain is divided into 3*.*4 × 106 hexahedral ele- ments with a maximum aspect ratio of approximately 250. The solution polynomial degree is set to *𝑝* = 2 — a formally third-order accurate scheme. Thus, the total number of DOFs is approximately 9*.*18 × 107.

The grid is constructed using the commercial software Pointwise V18.3 released in September 2019; solid boundaries are described using a quadratic mesh. The panel of [Fig.](#_bookmark31) [14(a)](#_bookmark31) illustrates an overview of the front wing geometry and the mesh, where the contour plot of the time- averaged pressure coefficient on the surface of the front wing is shown in [Fig.](#_bookmark35) [14(b)](#_bookmark35).



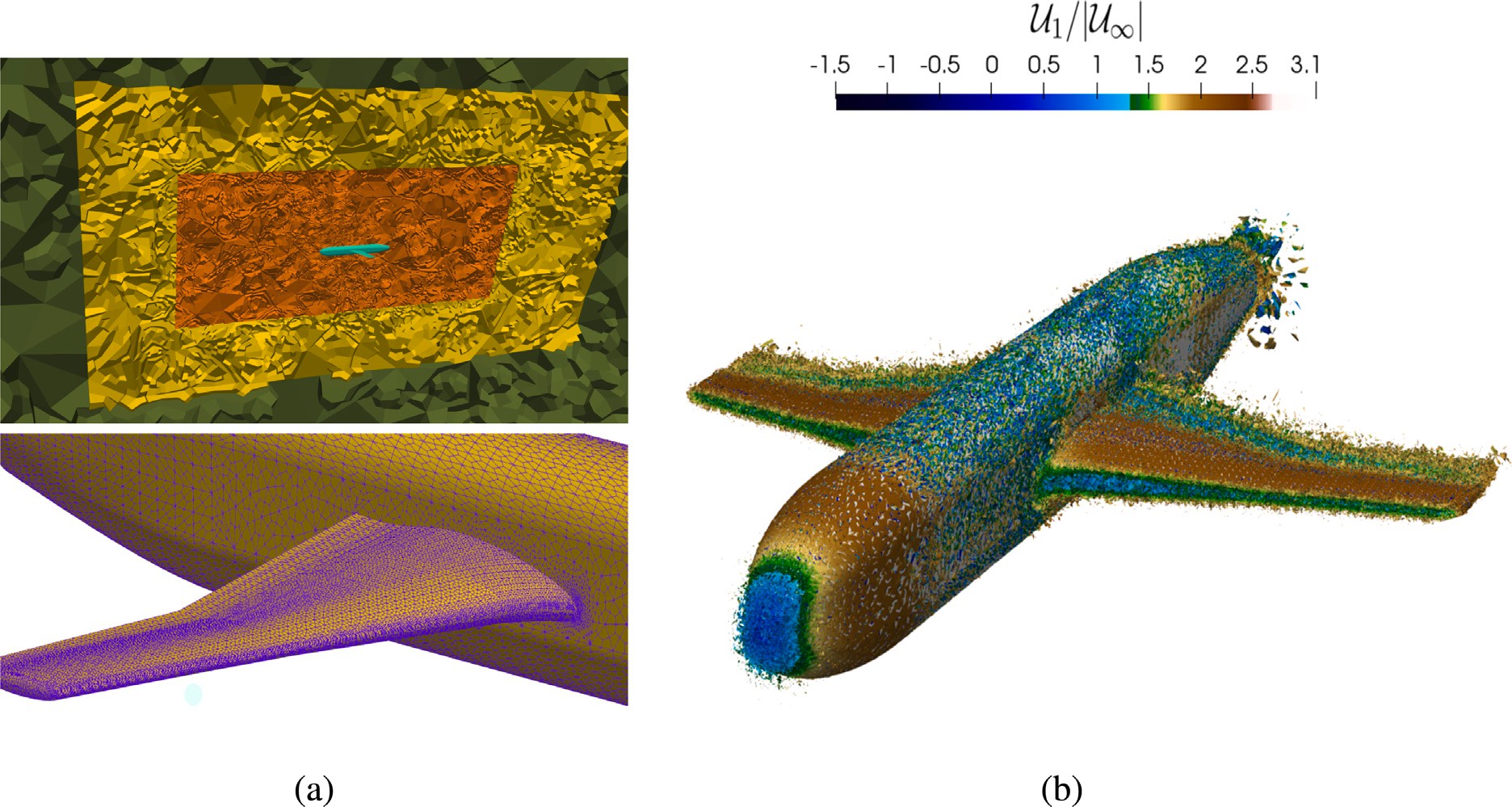
**/ig. 10.** Flow around a delta wing: Cost performance in USD of AMD & Arm CPU architectures.

The variation of the wall-clock time against the number of nodes for different types of CPU architecture is shown in [Fig.](#_bookmark32) [15](#_bookmark32). Overall, we observe that the simulations speed up as the number of CPUs increases for all architectures on the on-premises cluster and the AWS EC2 instances. These results confirm that the problem size influenced the performance of the on-premises cluster Ibex presented in the previous sections. [Fig.](#_bookmark33) [15(a)](#_bookmark33) shows the wall-clock time used by the Intel Cascade Lake architecture on the Ibex cluster and the AWS EC2 c5d instances. The simulation on 8 nodes of the on-premises cluster with 40 MPI threads produced favorable results. For all four computations, doubling the number of nodes reduces the time-to-solution by a factor of ap- proximately 1.8. However, for 8 nodes Intel Cascade Lake architecture and 40 MPI threads, this factor reduces to 1.7. In [Fig.](#_bookmark34) [15(b)](#_bookmark34), we report the performance of the Ibex cluster using AMD processes and AWS ParallelCluster using c5a and c6 g instances. As we can observe, the c6 g.8xlarge AWS EC2 instance delivers the results in the least amount of time with a parallel efficiency of approximately 70%. By comparing [Figs.](#_bookmark33) [15(a)](#_bookmark33) and [15(b)](#_bookmark34), we observe that the c6 g.8xlarge AWS EC2 instance achieve the best performance among all the CPU architectures. Although parallel performance is important, the cost of the simu- lations is a significant concern. As illustrated in [Fig.](#_bookmark36) [16](#_bookmark36), the cost of computations on the on-premises cluster is much lower than that of the AWS ParallelCluster. Across all the AWS EC2 instances, the c6 g.8xlarge

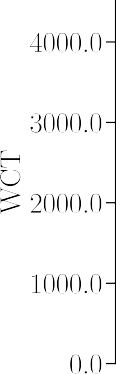
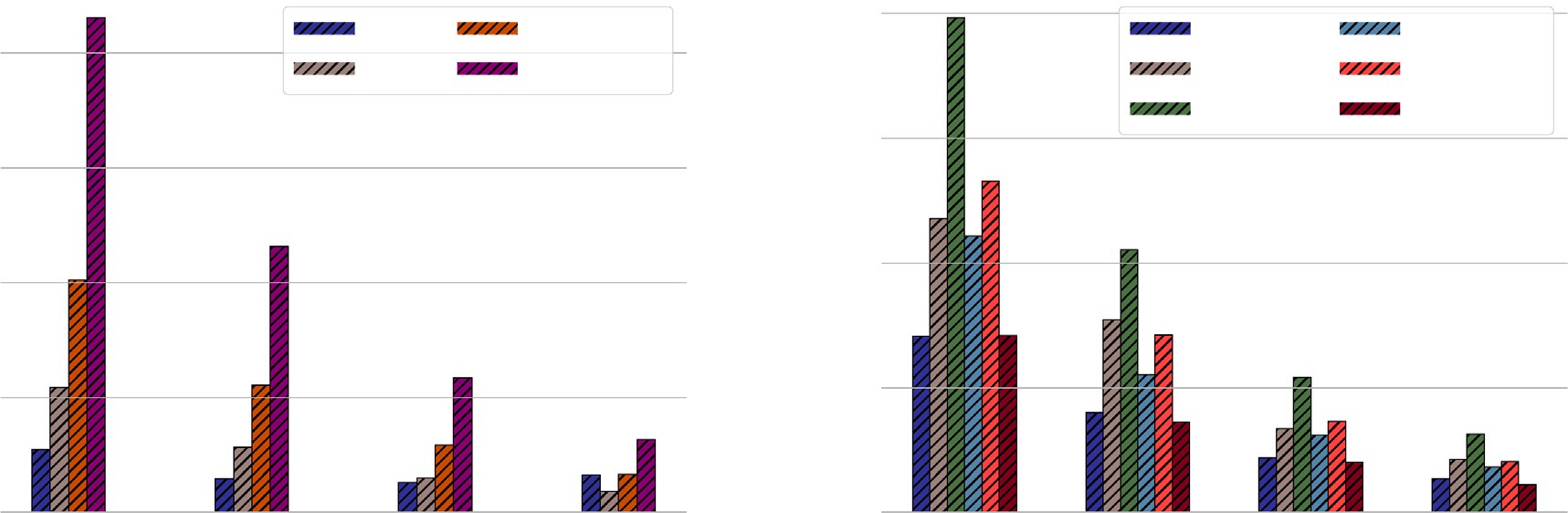
Arm architecture is the cheapest one.

# Conclusion

In this work, we evaluate the performance of a prototype of next generation high-order entropy stable solvers for compressible flows on unstructured grids on the Amazon Web Services Elastic Compute Cloud and the on-premise resource Ibex cluster hosted at KAUST. The study aims to establish the possibility of using Amazon’s cloud-based high-performance computing service to address complex computational fluid dynamics industry flow problems and propose a set of Elastic Cloud Computing instances that provide the fastest time to the solution and offer more affordable computations. In terms of time-to-solution, the Amazon Web Services Elastic Compute Cloud delivers the best performance, with the Graviton2 processors based on the Arm archi- tecture being the fastest. However, the results also indicate that the nodes based on the AMD Rome architecture of Ibex deliver very good performance, close to those observed for the Amazon Cloud service. In addition, we found that the simulations on the Ibex cluster are currently less expensive than those performed on the cloud for all orders of accuracy. Future work will demonstrate the performance at higher core counts and include post-processing elements of the computational fluid dynamics process.



**/ig. 11.** NASA juncture flow experiment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 12.** NASA juncture flow experiment: Wall-clock time in seconds for each simulation against the number of nodes.

# CRediT authorship contribution statement

**R. Al Jahdali:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Soft- ware, Validation, Visualization, Writing – original draft, Writing – review & editing. **S. Kortas:** Resources, Software. **M. Shaikh:** Re- sources, Software. **L. Dalcin:** Resources, Software, Review & editing.

**M. Parsani:** Conceptualization, Project administration, Supervision, Writing – review & editing.

# Declaration of competing interest

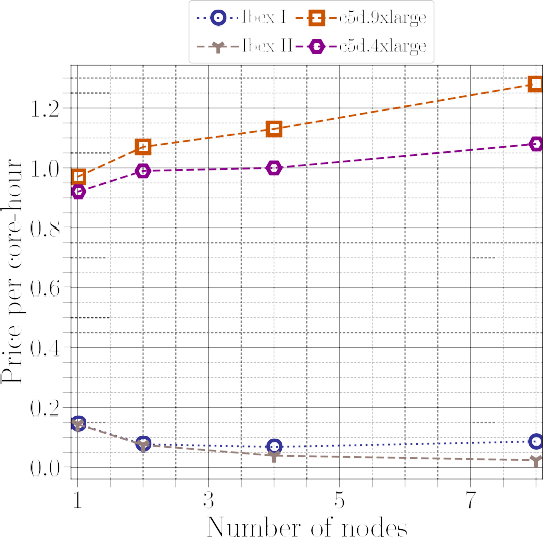
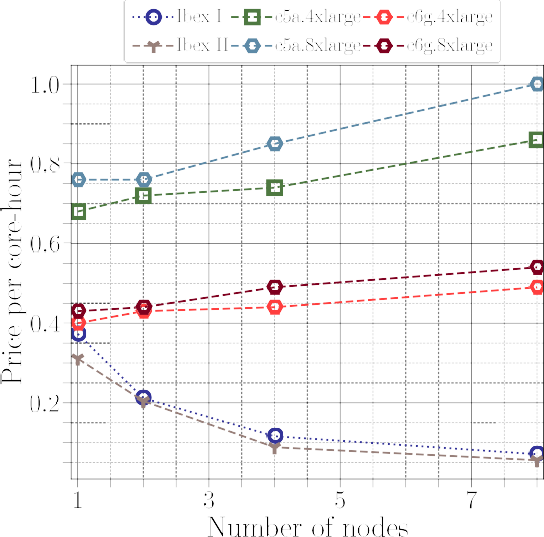
The authors declare that they have no known competing finan- cial interests or personal relationships that could have appeared to influence the work reported in this paper.

# Data availability

Data will be made available on request.

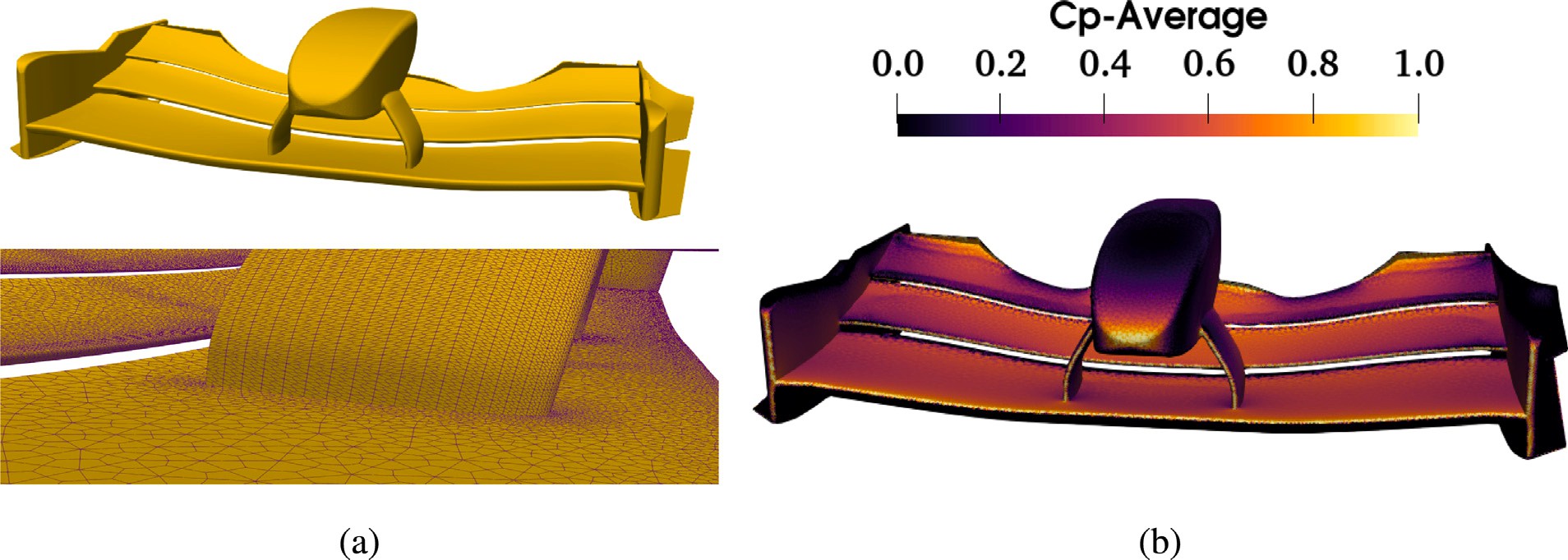
# Acknowledgments

The work described in this paper was supported by King Abdullah University of Science and Technology, Saudi Arabia through the award OSR-2019-CCF-3666. The authors are also thankful for the computing resources of the Supercomputing Laboratory and the Extreme Com- puting Research Center at King Abdullah University of Science and Technology.

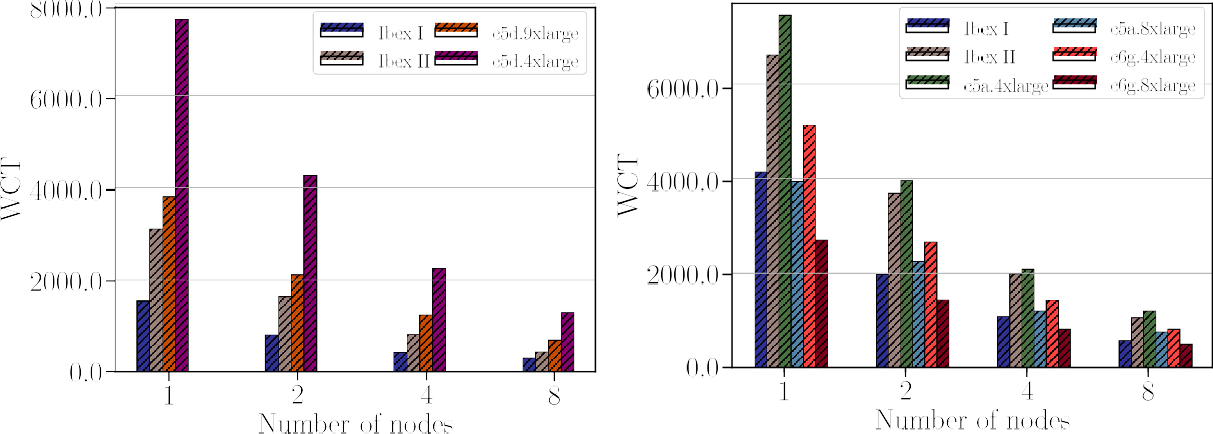
 



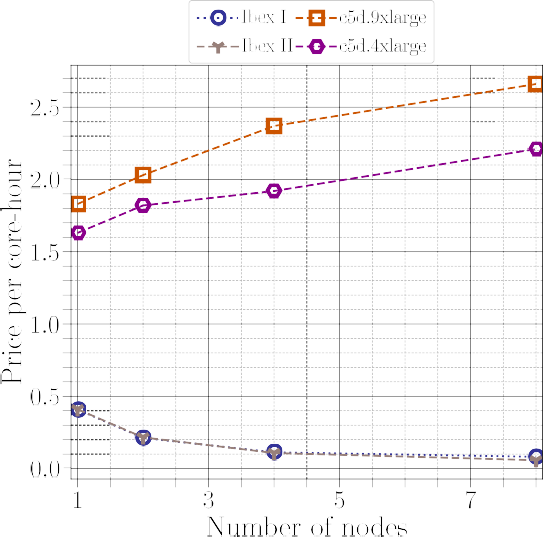
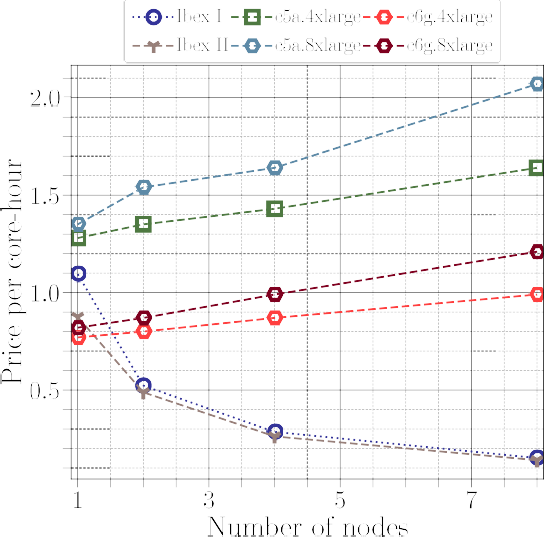
**/ig. 13.** NASA juncture flow experiment: Cost performance in USD of Ibex, and AWS EC2 instances.



**/ig. 14.** Formula ([1](#_bookmark5)) front wing (Imperial Front Wing).



**/ig. 15.** Formula ([1](#_bookmark5)) front wing: Wall-clock time in seconds for each simulation against the number of nodes.



**/ig. 16.** Formula ([1](#_bookmark5)) front wing: Cost performance in USD of Ibex, and AWS EC2 instances.

# Appendix A. An overview of the semidiscrete entropy stable spa- tial discretization

This Appendix gives some of the details of the spatial discretization implemented in the SSDC framework. As a model problem, we use the advection-diffusion equation in multiple dimensions.

* 1. *Mapping*

partition the physical domain *𝛺 ⊂* R3 into *𝐾* non-overlapping elements, In general, to solve partial differential equations numerically, we where *𝐾* is a whole number greater than zero. Then, each element

in physical space is transformed using a local and invertible curvilin- ear coordinate transformation that is compatible at shared interfaces, meaning that the push-forward element-wise mappings are continuous

across physical element in(terfaces. )To achieve that, one maps from

the reference coordinates *𝜉*1*, 𝜉*2*, 𝜉*3 ∈ [−1*,* 1]3 to the p(hysical e)le-

me(nt (see [F](#_bookmark38))[ig.](#_bookmark38) [A.17](#_bookmark38)) by the push-forward transformation *𝑥*1*, 𝑥*2*, 𝑥*3 =

*𝑋 𝜉*1*, 𝜉*2*, 𝜉*3 , which, in the presence of curved elements, is usually

a high-order degree polynomial. This procedure requires no explicit

**/ig. A.17.** Example of mapping procedure of the reference element (left) to a physical element (right).

knowledge nor construction of the pull-back mappings in unstructured mesh schemes.

*𝜉*

( )

*𝜉* 𝖲*𝜉* 2 *𝜉*

3. 𝖰 ≡ + 1 𝖤

, 𝖲*𝜉* = −

( ) T

*𝜉*

𝖲

, 𝖤*𝜉* =

(𝖤 ) T,

𝖤 = diag (−1*,* 0*,* … *,* 0*,* 1) = ***𝒆***

***𝒆*** T − ***𝒆 𝒆*** T, ***𝒆***

≡ [1*,* 0*,* … *,* 0] T, and

* 1. *Summation-by-parts operators*

*𝜉*

***𝒆****𝑁*

≡ [0*,* 0*,* … *,* 1] T.

*𝑁 𝑁*

1 1 1

The physical (spatial) domain *𝛺 ⊂* R3 with boundary *𝜕𝛺* is dis-

cretized using tensor-product elements. In this work, the derivatives in each element are discretized using one-dimensional SBP operators [[50](#_bookmark95), [51](#_bookmark96)] which are given for completeness in [Definition](#_bookmark39) [1](#_bookmark39).

**Definition 1.** A matrix operator, 𝖣*𝜉* ∈ R*𝑁*×*𝑁* , is an SBP o[pera]tor of

An SBP operator of degree *𝑝* is hence an operator that differentiates

exactly monomials up to degree *𝑝*.

In this work, we use a collocated discontinuous Galerkin approach with diagonal norm SBP operators are constructed on the Legendre– Gauss–Lobatto (LGL) nodes [[10](#_bookmark57)]. The SBP operators used in this work are explicitly constructed in [[22](#_bookmark67)]. These operators are extended to

multiple dimensions by using tensor products (*⊗*).

degree *𝑝* approximating the derivative *𝜕*

*𝜕𝜉*

nodal distribution ***𝝃*** having *𝑁𝑙* nodes, if

1. 𝖣*𝜉* ***𝝃****𝑗* = *𝑗****𝝃****𝑗*−1, *𝑗* = 0*,* 1*,* … *, 𝑝*;

on the domain *𝜉* ∈

*𝛼, 𝛽*

with

* 1. *Semidiscretization of the linear advection-diffusion equation*

2.

≡ (𝖯 )−1 𝖰 , where the norm matrix, 𝖯 , is symmetric positive

The linear convection–diffusion equation in Cartesian physical co-

*𝜉 𝜉 𝜉 𝜉*

definite; ordinates reads

∀ *𝑥*1*, 𝑥*2*, 𝑥*3

∈ *𝛺*

*𝑚*=1 *𝑚*

1 3 *𝜕*

*𝜅*

*𝜕𝜉*

*𝜅 𝑚 𝜅 𝑚*

( ) (*𝐵*) ( )

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *𝜕****𝒒*** | + ∑3 | *𝜕* (*𝑎* ***𝒒***)  *𝑚* | = ∑3 *𝜕*2(*𝑏* ***𝒒***)  *𝑚 ,* | to |  | | |
| *𝜕𝑡*  ( | *𝑚*=1 | *𝜕𝑥𝑚*  ) | *𝜕𝑥*2  *, 𝑡* ≥ 0*,* | *𝜕****𝒒*** ∑ {  *𝐽* + | | (  *𝐽 𝑙 𝑎* | ) *𝜕𝜉𝑙 𝜕* ( )}  ***𝒒*** + *𝐽 𝑎* ***𝒒*** |

(A.1)

*𝜕𝑡*

2 *𝑙,𝑚*=1

*𝜕𝜉𝑙*

*𝜕𝑥𝑚*

*𝜕𝑥𝑚 𝜕𝜉𝑙*

***𝒒*** (*𝑥*1*, 𝑥*2*, 𝑥*3*, 𝑡* = ***g***

*𝑥*1*, 𝑥*2*, 𝑥*3*, 𝑡 ,*

1 ∑3 {

( )} ∑3 ( )

∀ *𝑥*1*, 𝑥*2*, 𝑥*3) ∈ *𝜕𝛺, 𝑡* ≥ 0*,*

– *𝑎* ***𝒒*** *𝜕*

*𝐽 𝜕𝜉𝑙* =

*𝜕 𝐽*

*𝜕𝜉𝑙*

*𝜕𝜉𝑎 𝜕*(*𝑏𝑚****𝒒***)

*,*

( ) ( )

2 *𝑚*

*𝜕𝜉𝑙*

*𝜅 𝜕𝑥𝑚*

*𝜕𝜉𝑙*

*𝜅 𝜕𝑥𝑚 𝜕𝑥𝑚*

*𝜕𝜉𝑎*

***𝒒*** *𝑥*1*, 𝑥*2*, 𝑥*3*,* 0

= ***g***(0)

*𝑥*1*, 𝑥*2*, 𝑥*3*,* 0 *,*

*𝑙,𝑚*=1

*𝑙,𝑎,𝑚*=1

∀ (*𝑥*1*, 𝑥*2*, 𝑥*3) ∈ *𝛺,*

where (*𝑎𝑚****𝒒***) are the inviscid fluxes, *𝑎𝑚* are the (constant) components of

(A.6)

where the last set of terms on the left-hand side are zero by the GCL

the convection speed, *𝜕*(*𝑏𝑚****𝒒***) are the viscous fluxes, and *𝑏*

*𝜕𝑥𝑚*

*𝑚*

are the (con-

conditions ([A.4](#_bookmark46)). Then, a stable semi-discrete form can be constructed

similarly to the split form ([A.6](#_bookmark41)) by d[iscreti]zing the inviscid portion

stant and positive) diffusion coefficients. If the coefficients *𝑏𝑚* are set

to zero, we will have a purely hyperbolic system. The boundary data,

of ([A.2](#_bookmark42)) and ([A.5](#_bookmark47)) using 𝖣*𝜉𝑙*

, 𝖩*𝜅* , and

J *𝜕𝜉𝑙*

*𝜕𝑥𝑚*

, and by averaging the

*𝜅*

***g***(*𝐵*), and the initial condition, ***g***(0), are assumed to be in *𝐿*2(*𝛺*), with the further assumption that ***g***(*𝐵*) is prescribed so that linear stability (energy

results. The viscous terms are obtained from the discretization of the viscous portion of ([A.5](#_bookmark47)). This process leads to

stability) is achieved. Here, derivatives are approximated with SBP

differentiation operators defined in computational space. Therefore, we use the Jacobian of the push-forward mapping and the chain rule to

𝖩*𝜅*

d***𝒒****𝜅* + d*𝑡*

1 ∑3

2 *𝑙,𝑚*=1

2

*𝑎𝑚*

{

𝖣*𝜉𝑙*

[J *𝜕𝜉𝑙* ]

*𝜕𝑥𝑚 𝜅*

+ [J *𝜕𝜉𝑙* ]

*𝜕𝑥𝑚 𝜅*

}

𝖣*𝜉𝑙*

***𝒒****𝜅*

transform Eq. ([A.1](#_bookmark40)) from physical to computational space as

1 ∑3 {

( ) [

*𝜕𝜉* ] }

*𝜕****𝒒*** ∑3

*𝑚*

*𝜕𝜉𝑙 𝜕*

(*𝑎* ***𝒒***) ∑3

*𝜕𝜉𝑙*

*𝜕* (

*𝜕𝜉𝑎 𝜕*(*𝑏𝑚****𝒒***) )

–

*𝑙,𝑚*=1

*𝑎𝑚* diag

***𝒒****𝜅*

𝖣*𝜉𝑙*

J

*𝜕𝑥𝑚*

*𝑙*

**𝟏***𝜅* =

*𝜅*

(A.7)

*𝐽𝜅 𝜕𝑡* + *𝐽𝜅 𝜕𝑥*

*𝑙,𝑚*

=1 *𝑚*

*𝜕𝜉* = *𝐽𝜅 𝜕𝑥 𝜕𝜉*

*𝑙,𝑎,𝑚*

*𝑙* =1 *𝑚 𝑙*

*𝜕𝑥𝑚*

*𝜕𝜉𝑎*

*,* (A.2)

∑3

*𝑏* 𝖣

𝖩−1 [J *𝜕𝜉𝑙* ]

[J *𝜕𝜉𝑎* ]

where *𝐽𝜅* is the determinant of the metric Jacobian. Moving the metric

*𝑚*

*𝑙,𝑚,𝑎*=1

*𝜉𝑙 𝜅*

*𝜕𝑥𝑚 𝜅*

*𝜕𝑥𝑚*

𝖣*𝜉* ***𝒒****𝜅* + **𝐒𝐀𝐓***𝜅 ,*

*𝜅*

*𝑎*

terms *𝐽*

*𝜕𝜉𝑙*

*𝜅 𝜕𝑥𝑚*

inside the derivative, and using the product rule, leads to

where **𝟏***𝜅*

is a vector of ones of size *𝑁𝜅* . The semi-discrete form ([A.7](#_bookmark43))

*𝜕****𝒒***

∑3 *𝜕* (

*𝜕𝜉𝑙*

) ∑3

*𝜕* (

*𝜕𝜉* )

highlights a set of discrete GCL conditions (the analog of the continuous GCL conditions ([A.4](#_bookmark46)))

*𝐽𝜅 𝜕𝑡* + *𝜕𝜉*

*𝑙,𝑚*

=1 *𝑙*

*𝐽𝜅 𝜕𝑥 𝑎𝑚****𝒒***

– *𝑎𝑚****𝒒***

*𝑙,𝑚*=1 *𝑙*

*𝜕𝜉*

*𝐽𝜅 𝜕𝑥* =

∑3 [J *𝜕𝜉𝑙* ]

∑3 *𝜕* (

*𝑙*

*𝑚*

*𝑚*

*𝜕𝜉𝑙*

*𝜕𝜉𝑎 𝜕*(*𝑏𝑚****𝒒***) )

∑3 *𝜕𝜉𝑎 𝜕*(*𝑏𝑚****𝒒***) *𝜕* (

*𝜕𝜉𝑙* )

*𝑙*=1

*𝜉𝑙*

*𝜕𝑥𝑚*

**𝟏***𝜅* = **𝟎***, 𝑚* = 1*,* 2*,* 3*.* (A.8)

*𝜅*

𝖣

*𝑙,𝑎,𝑚*=1

*𝜕𝜉𝑙*

*𝐽𝜅 𝜕𝑥*

*𝜕𝑥𝑚*

*𝜕𝜉𝑎*

–

*𝑙,𝑎,𝑚*=1

*𝜕𝑥𝑚*

*𝜕𝜉𝑎*

*𝜕𝜉𝑙*

*𝐽𝜅 𝜕𝑥 .*

The satisfaction of conditions ([A.8](#_bookmark44)) will result in the telescoping, prov- ably stable, semi-discrete form

(A.3)

*𝑚*

*𝑚*

𝖩 d***𝒒****𝜅* + 1 ∑

3

2

{

*𝑎* 𝖣

[J *𝜕𝜉𝑙* ]

+ [J *𝜕𝜉𝑙* ]

}

𝖣 ***𝒒*** =

The last terms on the left- and right-hand sides of ([A.3](#_bookmark45)) are zero via the

geometric conservation law (GCL) relations

*𝜅* d*𝑡*

*𝑚*

*𝑙,𝑚*=1

*𝜉𝑙*

*𝜕𝑥𝑚 𝜅*

*𝜕𝑥𝑚 𝜅*

*𝜉𝑙 𝜅*

(A.9)

∑3 *𝜕*

( *𝜕𝜉𝑙*

*𝜅*

*𝐽*

*𝑎*

)

= 0*, 𝑚* = 1*,* 2*,* 3*,* (A.4)

∑3

*𝑙,𝑚,𝑎*=1

*𝑏𝑚*𝖣*𝜉𝑙* 𝖩*𝜅*

−1 [J

*𝜕𝜉* ] [J

*𝜕𝑥𝑚 𝜅*

*𝑙*

*𝜕𝜉* ]

*𝜕𝑥𝑚 𝜅*

*𝑎*

𝖣*𝜉* ***𝒒****𝜅* + **𝐒𝐀𝐓***𝜅 .*

*𝑙*=1 *𝜕𝜉𝑙*

*𝜕𝑥𝑚*

At one of the *𝑖*th interfaces of the *𝜅*th element, the **𝐒𝐀𝐓**

*𝜅,𝑖*

in the normal

yielding the strong conservation form of the convection–diffusion equa- direction is defined as

tion in computational space

**𝐒𝐀𝐓**

= − 1 𝖯−1£*̂*(*𝑎* ***𝒒***

– *𝛽𝐷****𝒒***

– **𝟏** *𝑔*) *,* (A.10)

*𝜕****𝒒***

∑3 *𝜕* (

*𝜕𝜉𝑙*

) ∑3

*𝜕* (

*𝜕𝜉𝑙*

*𝜕𝜉𝑎 𝜕*(*𝑏𝑚****𝒒***) )

*𝜅,𝑖* 2

*𝑛 𝜅*

*𝜅 𝑖*

*𝐽𝜅*

*𝜕𝑡*

+

*𝑙,𝑚*=1 *𝜕𝜉𝑙*

*𝐽𝜅*

*𝜕𝑥𝑚*

*𝑎𝑚****𝒒***

=

*𝑙,𝑎,𝑚*=1 *𝜕𝜉𝑙*

*𝐽𝜅*

*𝜕𝑥𝑚 𝜕𝑥𝑚*

*𝜕𝜉𝑎*

*.* (A.5)

where 𝖯 is the norm-matrix of the SBP operator, and £*̂* is a matrix that extracts from the solution vector only the solution values associated to

Now, consider the following differentiation matrices for the three- dimensional case:

≡ I

≡ I

𝖣*𝜉* ≡ 𝖣 *⊗* I *⊗* I*𝑁 ,* 𝖣*𝜉 𝑁 ⊗* 𝖣*𝜉 ⊗* I*𝑁 ,* 𝖣*𝜉 𝑁 ⊗* I*𝑁 ⊗* 𝖣*𝜉 ,*

*𝜉 𝑁*

the LGL points that lie on the *𝑖*th interface. The scalar *𝑎𝑛* is the advection velocity projected in the normal direction, while the scalar *𝛽* is defined as *𝛽* = 2|*𝑎*|∕Pe. The acronym Pe stands for the Pèclet number. The

symbol **𝟏** represents a vector of ones of size (*𝑝* + 1)*𝑑𝑖𝑚*−1. For an interior

1 2 3 2 1

3 3 1 2

*𝑖*

interface, *𝑔* contains the information of the adjoining solution element,

where I*𝑁* is an *𝑁𝑙* × *𝑁𝑙* identity matrix and *𝑁𝑙* represents the number

*𝑙*

of LGL points per direction in a given element. The diagonal matrix

containing the metric Jacobian is defined as

whereas *𝑔* is constructed using boundary data for a boundary interface.

# References

𝖩 ≡ diag (*𝐽* (***𝝃***(1))*,* … *, 𝐽* (***𝝃***(*𝑁𝜅* ))) *,*

*𝜅 𝜅 𝜅*

[ *𝜕𝜉* ]

1. [Chung B, Cebral JR. CFD for evaluation and treatment planning of aneurysms:](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb1)

[review of proposed clinical uses and their challenges. Ann Biomed Eng](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb1)

while the diagonal matrix of the metric terms, chosen to be a discretization of

J *𝑙*

*𝜕𝑥𝑚*

, has to be

*𝜅*

[2015;43(1):122–38.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb1)

1. [Tang WM, Chan VS. Advances and challenges in computational plasma science.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb2)

( *𝜕𝜉*

*𝜕𝜉* )

[Plasma Phys. Controlled Fusion 2005;47(2):R1–34.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb2)

diag

*𝐽𝜅*  *𝑙* (***𝝃***(1))*,* … *, 𝐽𝜅*  *𝑙* (***𝝃***(*𝑁𝜅* )) *,*

1. [Schneider T, Teixeira J, Bretherton CS, Brient F, Pressel KG, Schär C,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb3)

*𝜕𝑥𝑚*

where *𝑁* ≡ *𝑁 𝑁 𝑁*

*𝜕𝑥𝑚*

is the total number of LGL nodes in the *𝜅*th

[Siebesma AP. Climate goals and computing the future of clouds. Nature Clim](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb3) [Change 2017;7(1):3–5.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb3)

*𝜅* 1 2 3

element. To construct a stable scheme, we canonically split the inviscid

terms into one half of the inviscid terms in ([A.2](#_bookmark42)) and one half of the inviscid terms in ([A.3](#_bookmark45)) (see [[22](#_bookmark67)]), while the viscous terms are handled in strict conservation form. At the continuous level, this process leads

1. [Neumann P, Düben P, Adamidis P, Bauer P, Brück M, Kornblueh L, Klocke D,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb4) [Stevens B, Wedi N, Biercamp J. Assessing the scales in numerical weather](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb4) [and climate predictions: will exascale be the rescue? Phil Trans R Soc A](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb4) [2019;377(2142):20180148.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb4)
2. [Bauer P, Thorpe A, Brunet G. The quiet revolution of numerical weather](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb5) [prediction. Nature 2015;525(7567):47–55.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb5)
3. [Richardson LF. Weather prediction by numerical process. Cambridge University](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb6) [Press; 2007.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb6)
4. [Anderson JD. Basic philosophy of CFD. In: Computational fluid dynamics.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb7) [Springer; 2009, p. 3–14.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb7)
5. [Spalart PR, Venkatakrishnan V. On the role and challenges of CFD in the](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb8) [aerospace industry. Aeronaut J 2016;120(1223):209–32.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb8)
6. [Slotnick J, Khodadoust A, Alonso J, Darmofal D, Gropp W, Lurie E, Mavriplis D.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb9) [CFD vision 2030 study: A path to revolutionary computational aerosciences.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb9) [NASA-CR-2014-218178, 2014.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb9)
7. [Parsani M, Boukharfane R, Nolasco IR, Del Rey Fernández DC, Zampini S,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb10) [Hadri B, Dalcin L. High-order accurate entropy-stable discontinuous collo-](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb10) [cated Galerkin methods with the summation-by-parts property for compressible](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb10) [CFD frameworks: Scalable SSDC algorithms and flow solver. J Comput Phys](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb10) [2020;424:109844.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb10)
8. [Peña-Monferrer C, Manson-Sawko R, Elisseev V. HPC-cloud native framework](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb11) [for concurrent simulation, analysis and visualization of CFD workflows. Future](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb11) [Gener Comput Syst 2021;123:14–23.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb11)
9. [Ashton N, Sachs S, Foti L, Eberhardt S. Towards high-fidelity CFD on the](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb12) [cloud for the automotive and motorsport sectors. In: WCX SAE world congress](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb12) [experience. SAE International; 2020.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb12)
10. [Turner M, Appa J, Ashton N. Performance of CPU and GPU HPC architectures](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb13) [for off-design aircraft simulations. In: AIAA scitech 2021 forum. 2021, p. 1–7.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb13)
11. [Al Jahdali R, Dalcin L, Parsani M. On the performance of relaxation and](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb14) [adaptive explicit runge–kutta schemes for adaptive high-order compressible flow](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb14) [simulations. J Comput Phys 2022;In press.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb14)
12. [Ranocha H, Dalcin L, Parsani M, Ketcheson DI. Optimized Runge-Kutta methods](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb15) [with automatic step size control for compressible computational fluid dynamics.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb15) [Commun Appl Math Comput 2021;1–38.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb15)
13. [Carpenter MH, Fisher TC, Nielsen EJ, Frankel S. Entropy stable spectral colloca-](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb16) [tion schemes for the Navier–Stokes equations: Discontinuous interfaces. SIAM J](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb16) [Sci Comput 2014;36(5):835–67.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb16)
14. [Parsani M, Carpenter MH, Fisher T, Nielsen E. Entropy stable staggered grid](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb17) [discontinuous spectral collocation methods of any order for the compressible](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb17) [Navier–Stokes equations. SIAM J Sci Comput 2016;38(5):3129–62.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb17)

[ter MH. An entropy stable *ℎ*∕*𝑝* non-conforming discontinuous Galerkin method](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb18) [18] [Friedrich L, Winters AR, Del Rey Fernández DC, Gassner GJ, Parsani M, Carpen-](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb18)

[with the summation-by-parts property. J Sci Comput 2018;77(2):689–725.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb18)

[stable *ℎ*∕*𝑝*-nonconforming discretization with the summation-by-parts property](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb19) [19] [Fernández DCDel Rey, Carpenter MH, Dalcin L, Zampini S, Parsani M. Entropy](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb19)

[for the compressible Euler and Navier–Stokes equations. SN Partial Differ Equ](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb19) [Appl 2020;1(2):1–54.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb19)

1. [Parsani M, Carpenter MH, Nielsen EJ. Entropy stable wall boundary conditions](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb20) [for the three-dimensional compressible Navier–Stokes equations. J Comput Phys](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb20) [2015;292:88–113.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb20)
2. [Dafermos CM. Hyperbolic conservation laws in continuum physics. Berlin:](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb21) [Springer-Verlag; 2010.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb21)
3. [Carpenter MH, Parsani M, Fisher TC, Nielsen EJ. Entropy stable staggered grid](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb22) [spectral collocation for the Burgers’ and compressible Navier-Stokes equations.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb22) [NASA TM-2015-218990, 2015.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb22)
4. [Svärd M. A convergent numerical scheme for the compressible Navier–Stokes](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb23) [equations. SIAM J Numer Anal 2016;54(3):1484–506.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb23)

[Gassner GJ, Parsani M. Entropy-stable *𝑝*-nonconforming discretizations with](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb24) [24] [Fernández DCDel Rey, Carpenter MH, Dalcin L, Fredrich L, Winters AR,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb24)

[the summation-by-parts property for the compressible Navier–Stokes equations.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb24) [Comput & Fluids 2020;210:104631.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb24)

1. [Nolasco IR, Dalcin L, Fernández DCDel Rey, Zampini S, Parsani M. Opti-](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb25) [mized geometrical metrics satisfying free-stream preservation. Comput & Fluids](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb25) [2020;207:104555.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb25)
2. [Parsani M, Carpenter MH, Nielsen EJ. Entropy stable discontinuous interfaces](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb26) [coupling for the three-dimensional compressible Navier–Stokes equations. J](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb26) [Comput Phys 2015;290:132–8.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb26)

[stable *ℎ*∕*𝑝* non-conforming discretization with the summation-by-parts property](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb27) [27] [Del Rey Fernández DC, Carpenter MH, Dalcin L, Zampini S, Parsani M. Entropy](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb27)

[for the compressible Euler and Navier–Stokes equations. SN Partial Differ Equ](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb27) [Appl 2020;1(2):1–54.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb27)

1. [Crean J, Hicken JE, Del Rey Fernández DC, Zingg DZ, Carpenter MH. Entropy-](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb28) [stable summation-by-parts discretization of the Euler equations on general curved](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb28) [elements. J Comput Phys 2018;356:410–38.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb28)
2. [Dalcin L, Rojas D, Zampini S, Del Rey Fernández DC, Carpenter MH, Parsani M.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb29) [Conservative and entropy stable solid wall boundary conditions for the com-](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb29) [pressible Navier–Stokes equations: Adiabatic wall and heat entropy transfer. J](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb29) [Comput Phys 2019;397:108775.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb29)
3. [Svärd Magnus, Özcan Hatice. Entropy-stable schemes for the Euler equations](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb30) [with far-field and wall boundary conditions. J Sci Comput 2014;58(1):61–89.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb30)
4. [Mengaldo Gianmarco, De Grazia Daniele, Witherden Freddie, Farrington Antony,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb31) [Vincent Peter, Sherwin Spencer, Peiro Joaquim. A guide to the implementa-](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb31) [tion of boundary conditions in compact high-order methods for compressible](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb31) [aerodynamics. In: 7th AIAA theoretical fluid mechanics conference. 2014, p.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb31) [2923.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb31)
5. [Butcher JC. Numerical methods for ordinary differential equations. Chichester:](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb32) [John Wiley & Sons Ltd; 2016.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb32)
6. [Ranocha H, Sayyari M, Dalcin L, Parsani M, Ketcheson DI. Relaxation](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb33) [Runge-Kutta methods: Fully-discrete explicit entropy-stable schemes for the](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb33) [compressible Euler and Navier–Stokes equations. SIAM J Sci Comput](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb33) [2020;42(2):A612–38.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb33)
7. [Rogowski M, Dalcin L, Parsani M, Keyes DE. Performance analysis](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb34) [of relaxation runge–kutta methods. Int J High Perform Comput Appl](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb34) [2022;10943420221085947.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb34)
8. [Balay S, Abhyankar S, Adams MF, Benson S, Brown J, Brune P, Buschelman K,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb35) [Constantinescu E, Dalcin L, Dener A, Eijkhout V, Gropp WD, Hapla V, Isaac T,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb35) [Jolivet P, Karpeev D, Kaushik D, Knepley MG, Kong F, Kruger S, May DA,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb35) [McInnes L Curfman, Mills R Tran, Mitchell L, Munson T, Roman JE, Rupp K,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb35) [Sanan P, Sarich J, Smith BF, Zampini S, Zhang H, Zhang H, Zhang J. PETSc/TAO](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb35) [users manual. Technical report ANL-21/39 - revision 3.16, Argonne National](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb35) [Laboratory; 2021.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb35)
9. [Knepley MG, Karpeev DA. Mesh algorithms for PDE with Sieve I: Mesh](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb36) [distribution. Sci Program 2009;17(3):215–30.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb36)
10. [Abhyankar S, Brown J, Constantinescu EM, Ghosh D, Smith BF, Zhang H.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb37) [PETSc/TS: A modern scalable ODE/DAE solver library. Technical report, 2018.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb37)
11. [Burstedde C, Wilcox LC, Ghattas O. p4est: Scalable algorithms for par-](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb38) [allel adaptive mesh refinement on forests of octrees. SIAM J Sci Comput](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb38) [2011;33(3):1103–33.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb38)
12. [Isaac T, Burstedde C, Wilcox LC, Ghattas O. Recursive algorithms for distributed](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb39) [forests of octrees. SIAM J Sci Comput 2015;37(5):C497–531.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb39)
13. Isaac T, Knepley MG. Support for non-conformal meshes in PETSc’s DMPlex interface. 2015, arXiv preprint [arXiv:1508.02470](http://arxiv.org/abs/1508.02470).
14. Cenaero. HiOCFD5, 5th international workshop on high-order CFD methods. 2018, URL [https://how5.cenaero.be](https://how5.cenaero.be/).
15. [Kimball E, Whitaker T, Kevrekidis YG, Benziger JB. Drops, slugs, and flooding](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb42) [in polymer electrolyte membrane fuel cells. AIChE J 2008;54(5):1313–32.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb42)
16. [Cebeci T, Kafyeke F. Aircraft icing. Annu Rev Fluid Mech 2003;35(1):11–21.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb43)
17. [Madani S, Amirfazli A. Oil drop shedding from solid substrates by a shearing](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb44) [liquid. Colloids Surf A 2014;441:796–806.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb44)
18. [Hummel D, Redeker G. A new vortex flow experiment for computer code](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb45) [validation. In: RTO/AVT symposium on vortex flow and high angle of attack](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb45) [aerodynamics, meeting proc. RTO-MP-069. 2003, p. 8–31.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb45)
19. [Mengaldo Gianmarco, Moxey David, Turner Michael, Moura Rodrigo Costa,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb46) [Jassim Ayad, Taylor Mark, Peiro Joaquim, Sherwin Spencer. Industry-relevant](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb46) [implicit large-eddy simulation of a high-performance road car via spectral/hp](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb46) [element methods. SIAM Rev 2021;63(4):723–55.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb46)
20. [Rumsey CL, Morrison JH. Goals and status of the NASA juncture flow experiment.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb47) [NATO, STO-MP-AVT-246, 2016.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb47)
21. [Pegrum JM. Experimental study of the vortex system generated by a formula 1](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb48) [front wing (Ph.D. thesis), Imperial College London; 2007.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb48)
22. [Buscariolo FF, Hoessler J, Moxey D, Jassim A, Gouder K, Basley J, Murai Y,](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb49) [Assi GRS, Sherwin SJ. Spectral/hp element simulation of flow past a formula](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb49) [one front wing: validation against experiments. 2019.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb49)
23. [Svärd M, Nordström J. Review of summation-by-parts schemes for initial](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb50) [boundary–value problems. J Comput Phys 2014;268:17–38.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb50)
24. [Del Rey Fernández DC, Hicken JE, Zingg DW. Review of summation-by-parts](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb51) [operators with simultaneous approximation terms for the numerical solution of](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb51) [partial differential equations. Comput & Fluids 2014;95:171–96.](http://refhub.elsevier.com/S2590-0056(22)00101-1/sb51)