2020-01-0665 Published 14 Apr 2020

Towards High-Fidelity CFD on the Cloud for the Automotive and Motorsport Sectors

Neil Ashton, Stephen Sachs, Lewis Foti, and Scott Eberhardt Amazon Web Services

Citation: Ashton, N., Sachs, S., Foti, L., and Eberhardt, S., "Towards High-Fidelity CFD on the Cloud for the Automotive and Motorsport Sectors," SAE Technical Paper 2020-01-0665, 2020, doi:10.4271/2020-01-0665.

Abstract

This paper presents the results from an investigation into the performance of OpenFOAM v1806 on the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) service for a realistic racing vehicle using a high-fidelity hybrid RANS-LES CFD approach. It is shown that AWS can provide the HPC environment to enable greater use of high-fidelity CFD methods by allowing higher core counts to reduce turn-around time. With the correct instance type - which potentially differs between meshing and solving - AWS was competitive against a high-performance Cray XC30 supercomputer, up to 1920 cores and meshes up to 280 million cells. However it is recognised that this Cray XC30 displayed superior scaling whilst containing older generation processors (Intel Ivybridge) compared to the AWS Instances (Intel Skylake). It was found that the influence of instance type was more pronounced during the meshing process within OpenFOAM (snappyHexMesh) where only the C5n.18xlarge instance was able to match the performance of the Cray XC30. This work establishes a set of best-practices and baseline configuration that will be used to look at larger models, larger core counts and to focus on other areas of the CFD workflow including post-processing.

Introduction

Aerodynamics is a critical differentiator for any road or race car company. For road-cars this is largely the drag of the vehicle, particularly in the age of electric vehicles where the drag is one of the largest contributors to affect the range of the vehicle. For motorsport, whilst minimizing drag is important, the downforce of the vehicle is more crucial, given that this has a direct influence on cornering speeds and the resulting lap time. This is particularly true for Formula 1, where the most well funded teams have more than fifty aerodynamicists looking for marginal gains over their competitors.

To find these marginal gains, aerodynamicists have two main tools; Computational Fluid Dynamics (CFD) and wind tunnel testing. Whilst track and road tests are also important tools, these are typically used later in the design process and conducting repeatable and reliable tests is a challenge. For Formula 1, track testing outside of race weekends is also heavily restricted which makes the emphasis on CFD and wind-tunnel testing even stronger.

Whilst wind-tunnel testing is arguably the most accurate way of assessing the performance of a vehicle, there are a number of factors that limit its use. Firstly the cost of hiring a suitable facility is typically in the order thousands to tens of thousands per day (depending on the facility) together with the need to bring a suitable model to test. The cost of manufacturing a model and the internal systems to compute the forces is also a significant one-off cost that can go into the tens of thousands of pounds. However once the main spine and outer shell of the car is built the cost of manufacturing the parts of the car to be tested i.e. different wing sections or body panels can be cheap with the rise of 3D printing. Another major barrier is the time required to design and manufacture these parts, with lead times being on the order of weeks to months. This means to conduct a wind-tunnel test of a road or race car the actual designs have to be completed typically several weeks to a month before the test itself. However it should be noted that a particular advantage of wind-tunnel testing is once the model is built it is possible to quickly test many different flow conditions e.g. ride height changes and flow speeds that offset some of the initial time and cost investment. Ultimately though these cost and time factors mean that wind-tunnel testing is more suited to the later phases of the design process where accuracy is prioritized over turn-around time.

Computational Fluid Dynamics (CFD) has grown in popularity over the past decade due to its ability to assess designs much more quickly than a whole wind-tunnel campaign and at a reduced cost. For the majority of road and race car companies, CFD is heavily used in the initial phases of the design work, where hundreds to thousands of designs are assessed before being manufactured and tested in a wind-tunnel to provide validation that the CFD trends are realised.

The main limiting factor with CFD is however its accuracy and resulting correlation with wind-tunnel and on-road tests. Whilst it may often be cheaper and faster than a wind-tunnel

campaign, if the results do not correlate in absolute terms or trends then it is not a helpful design tool. However it should also be noted that the ability to more easily visualize the entire flow-field in CFD is an additional advantage over wind-tunnel testing and can give extra insight into the flow physics.

We would like to note at this point that the total cost of both CFD and wind-tunnel testing is a combination of fixed and variable costs and we realise that this may influence the decision of how to split each companies use of CFD and wind-tunnel testing.

Turbulence Modelling

One of the most challenging aspects of CFD is how to treat the turbulent nature of the flow. Whilst the Navier-Stokes equations exactly represent the state of a fluid flow, the computational resources to resolve (both spatially and temporally, *i.e.* Direct Numerical Simulation (DNS)) all of the relevant scales of turbulence is beyond the reach of current supercomputers and is likely to stay this way for many decades to come.[1].

Current industrial CFD methods are typically based upon Reynolds-Averaged Navier-Stokes (RANS) approaches although a growing number of automotive companies are using higher-fidelity approaches. All RANS approaches introduce some amount of modeling, and these assumptions can often limit the accuracy of RANS approaches, especially in areas of complex flow separation [2]. Although, work continues in the development of a truly accurate universal RANS model [3, 4, 5, 6, 7], there is a growing realization that the current state-of-the-art models cannot provide the necessary level of accuracy when used standalone. [8, 9]

The Need for High-Fidelity Methods

While in a RANS approach the flow is approximated as a time-averaged flow field and the entire turbulence spectrum is modeled, in higher fidelity methods, such as Large Eddy Simulation (LES) based approaches, only the smallest turbulent scales are modeled and the rest of the scales, the large scales, are resolved. Compared to a RANS approach the level of modeling and empiricism is reduced (although not completely) and, hence, the approach is closer to DNS.

However, whereas a RANS approach can be solved in a steady-state fashion because of the assumption of a statistically mean flow, LES is intrinsically unsteady and due to the need for sufficient temporal and spatial resolution the computational cost is much greater than for steady RANS approaches.

As the filter-width is typically based upon the computational grid, the computational grid must be fine enough to capture all but the smallest turbulent scales. Additionally, towards the wall, due to the no-slip condition, the gradients of the velocity and higher-moments are increasing and, thus, require finer meshes in this region to capture the relevant physics correctly. This requirement on the spatial resolution means that the computational cost of a wall-resolved LES for a full car is exceeding current available resources [1].

Hybrid RANS-LES Methods

In recent years, there has been a focus on an approaches which seek to combine the best elements of both RANS and LES approaches. Hybrid RANS-LES methods solve RANS equations in regions where the flow is attached and, thus, more simple to model; switching to an LES-type formulation in regions where the accuracy provided by RANS is inadequate, *e.g.*, separated flow regions.

The many approaches developed in this way may be broadly split into two groups, zonal and non-zonal methods [10]. In zonal methods one explicitly defines a certain portion of the flow as RANS and the other as LES; usually, via the wall distance in inner scale, y^+. In contrast, non-zonal methods employ an intrinsic function, based on a flow quantity or the mesh itself, to automatically switch between a RANS or LES approach. The most common hybrid RANS-LES method in use in the aerospace sector today is Delayed Detached-Eddy Simulation (DDES) [11].

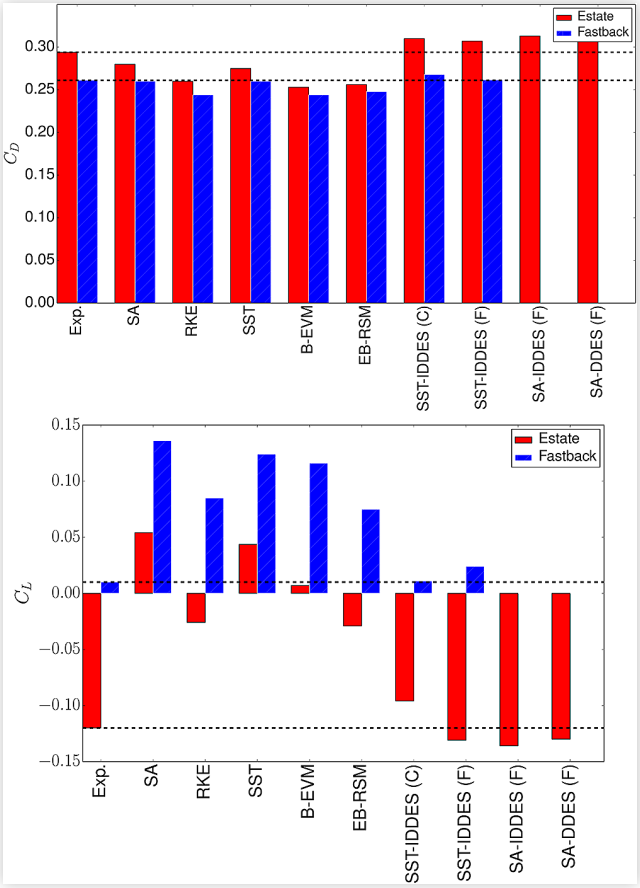
Since the development of DDES (or similar methods, such as Partially Averaged Navier-Stokes (PANS) [12], Scale-Adaptive Simulation (SAS) [13] or the many other variants in recent years) a number of alternative methods have been suggested to overcome some of the aforementioned limitations. These range from modifying the definition of the filter-width to sensitize it to 3D vortical structures that are common in the separated shear layers [14, 15, 16] to alternative formulations of the effective SGS model that the model operates under in LES mode [14].

The improvement that hybrid RANS-LES models can provide over RANS is nicely illustrated in Fig. 1 where the lift and drag coefficients are shown for two road vehicle configurations [17]. It can be seen that both eddy viscosity and second-moment closure RANS models fail to capture the lift coefficient, even if the drag is reasonably well captured. It is only hybrid RANS-LES methods, discussed in the next section that are able to capture both the lift and drag, which is crucial for road vehicle design. A number of other studies [18, 19, 20, 21, 22] have found similar conclusions that hybrid RANS-LES methods provide improved accuracy compared to RANS.

High-Performance Computing

Whilst the previous section has highlighted the desire to move to higher fidelity CFD approaches like the DES family of methods, the major limiting factor is the computational resources required to run these models. A RANS simulation of a road or race car would typically require 60-180M cells (depending on the wall-modelling approach which may lead to even larger meshes) [17, 23] and could be solved in a steady-state fashion in approximately 2000-5000 iterations. The exact

FIGURE 1 Drag and Lift Coefficients for two road vehicle configurations for a range of RANS and DES approaches. [17]



runtime would depend on the numerical approach (segregated/coupled, single/mixed/double precision) but approximately 4-8 hours on 256 CPU cores would be an average runtime. For a small engineering consultancy of three CFD users this would typically result in a cluster of ≈ 512 cores to facilitate a balance between turn-around time and cost. For a Formula 1 team with a much larger aerodynamics department, or a major road car company you would see clusters up to and beyond 10,000 cores to provide enough capacity for 10-25 daily CFD users.

HPC resources however become a particular bottleneck when a company wishes to move to DES-type approaches. Because of the transient nature of the approach, the time to simulate a typical road or race car with now 100-280M (or even higher) cells would take anything from 24 to 96 hours using 512 cores [24, 17, 23, 25]. This number is clearly dependent on the choice of time-step, CFD code and wall-modelling approach however these hybrid RANS-LES simulations are typically 5-10 times more computational expensive than RANS simulations [17, 26].

A greater issue within an engineering design process is the turn-around time. For many companies waiting for 48 hours for a result is too long and is often a waste of engineer time and this is particular true in the motorsport sector where the design process has to be fast to react to the large number of races in each year.

The typical requirement for a CFD turn-around time is for an engineer to set off a simulation before they leave for home and have it completed by the time they return to work the following day i.e. 12 hours. In our previous example we could run on 1024 cores if the original time was 24 hours or 4096 cores if it was originally 96 hours to reduce the time to 12 hours. Of course in reality few codes offer linear scaling so whilst 1024 cores would most likely scale linearly at 273k cells per core (considering a 280M cell mesh), at 4096 cores this would be 68k cells per core and may suffer from less than linear scaling due to an increasing contribution from networking communication.

However for the small engineering consultancy with an on-premise cluster of 512 cores, using 1024 or even 4096 cores would mean buying a much larger cluster to cater for this need. For a large road or race car company this would mean a choice between a cluster of > 50,000 cores, longer queue times or simply limiting the move to higher fidelity methods i.e. HPC becomes a bottleneck.

As we move towards even higher-fidelity methods such as wall-modelled LES and wall-resolved LES the need for ever greater HPC resources will only continue.

Cloud Computing

Whilst cloud computing has been used in the engineering design process for many years for data storage and more traditional IT operations, its use for HPC has been questioned because of the lack of high-speed networking. This perception has begun to change in the past 5 years with cloud computing companies such as AWS having released a number of critical compute, storage and networking components to make the use of HPC in the cloud a reality for even tightly-coupled problems such as CFD. For AWS this includes their Elastic Fabric Adapter (EFA) with a custom-built operating system (OS) bypass hardware interface to enhance the performance of inter-instance communications, parallel file systems such as Amazon FSx for Lustre and compute optimized hardware such as the C5n instances. The need for much larger clusters and the ability to run at much larger core counts to reduce run-around times have made many consider cloud computing resources where you only pay for the time you use the resources. This has the dual benefit of eliminating the capital costs of a dedicated cluster and providing access to larger scale clusters than would be economical to provision on-premise. The industry is currently experimenting with a number of approaches to incorporate cloud computing into CFD workloads. These range from a hybrid approach where cloud computing services are used to provide capacity for jobs that would be delayed queuing on the on-premise cluster to full scale migration to the cloud. Access to cloud resources provides an ability to scale for specific demands as and when required and removes availability of HPC capacity from the critical path. Not only has it become economical to use a large scale cluster as and when required but if time is the critical factor it is possible to run multiple such clusters simultaneously to explore different solutions in parallel.

Aims

In order to help a customer with their migration to AWS to undertake high-fidelity DDES simulations a number of benchmarks were undertaken on a range of compute types and compared against a Cray XC30 supercomputer which represents a high specification supercomputer. The aim was to establish whether AWS could be used to allow the use of high-fidelity methods within the desired 12 hour turn-around time that would not be achievable with the customers on-premise solution. It was observed at the beginning of this work that there was little openly available research assessing the performance of cloud based HPC environments against on-premise solutions and it is hoped that this work will serve as a body of work to help this discussion and area of research.

Test-Case

CFD Setup

A racing car was used as the geometry for the studies as shown in Figure 2. A low y^+ unstructured hex dominant mesh using the pre-processing tool ANSA by BETA-CAE Systems was created which resulted in a total mesh count of 280M cells for the entire car and a 143M cell half-car model (Figure 2). This would later be used to reach lower cells-per-core values to test the strong scaling of the code. The domain extents were such that a symmetry boundary condition was applied to the side and top walls and a no-slip condition was applied to the floor with a translating velocity equal to the freestream velocity of $50ms^{-1}$. An inlet and outlet boundary condition was applied at the entrance and exit of the computational domain. A rotating velocity boundary condition was applied to the wheels and the various radiators inside the car were modelled as porous regions with an appropriate pressure drop and the exhaust was set to a value corresponding to the engines used in the particular motorsport series in question.

The open-source CFD code OpenFOAM v1806 (Double Precision Floating Point) was used for all the studies and was compiled with OpenMPI 4.0.1. The SST-DDES turbulence model was used for all simulations with the segregated pimpleFoam solver (see Robertson et al. [27] for further details on this approach and OpenFOAM specific numerical methods).

FIGURE 2 Test Geometry and computational mesh

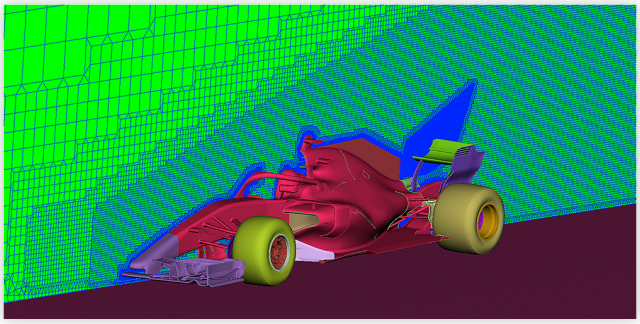
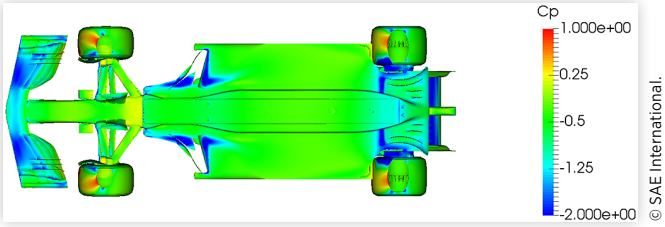


FIGURE 3 Pressure coefficient on the lower surface of the car



A time-step of $5e^{-4}$ using a second order backward difference numerical scheme was employed. A second order scheme was used for the turbulence quantities and a hybrid numerical scheme was used for the momentum equations to blend between central differencing and second order upwinding as per Travin et al. [28]. The Scotch domain decomposition method [29] was used for all simulations as previous results had shown poor scaling when using a hierarchical (i.e. constant x/y/z loading) domain decomposition approach due to poor load balancing. The Preconditioned Conjugate Gradient (PCG) method was used as the linear solver for the pressure and OpenFOAM's smoothSolver for the other variables.

Whilst the main focus on this paper is the scalability of OpenFOAM for a realistic race car geometry on AWS a full solution was made with this setup which compared favourably with other CFD tools, where the pressure coefficient underneath the car is shown in Figure 3 as an illustration of the flow-field.

To measure the solve time for this benchmarking study, each simulation was run for 500 iterations and the final 249 iterations (excluding the final iteration) were used to provide an average iteration time.

Meshing Test

An additional test was conducted to assess the performance of the snappyHexMesh meshing tool within OpenFOAM. In order to measure the meshing time, a complete snappyHexMesh mesh was conducted to create a two-car 435M cell mesh. The hierarchical domain decomposition method was used based upon previous experience of scotch being unsuitable for large cell counts during meshing. All other mesh settings were chosen based upon best-practices from previous work [18, 30].

Compute Environment

Three instances types most suitable for compute intensive work were used on AWS. The M5.x24large features Intel Xeon 8175M series at 2.5 GHz and represents a standard high core count two socket server, Z1D.12xlarge with Intel Xeon at 4.0 GHz represents the fastest clock speeds available and C5n.18xlarge with Intel Xeon at 3.0 GHz represents a fast core server with the fastest interconnect. Additionally for some of the tests the C5n.18xlarge instances were used with AWS Elastic Fabric Network (EFA) turned on. EFA is the network

interface for Amazon EC2 instances that enables users to run applications requiring high levels of inter-node communications. Its custom-built operating system (OS) bypass hardware interface enhances the performance of inter-instance communications, achieving lower latency and higher bandwidth.

OpenFOAM-v1806 in double floating point precision compiled on a CentOS-7 base image with the latest GCC-8.2.0 and OpenMPI-4.0.1 versions has been used. The MPI ranks are placed on subsequent physical cores. For the I/O local Amazon Elastic Block Storage (EBS) devices were used, such that the relatively short run time will not be influenced by I/O. However, the I/O is negligible.

Additional runs were conducted on a Cray XC30 with Intel Xeon E5 v2 12C 2.700 GHz nodes and an Aries interconnect. It should be noted that from the outset that it is appreciated that the specification of the nodes is outdated compared to the AWS instances chosen and thus it is expected that the performance would improve on the latest Cray machines using more recent processors. OpenFOAM was compiled using GCC with the Cray compiler wrappers and Cray MPI.

On neither AWS nor the Cray XC30 were any additional code tuning or optimization undertaken and we understand this may have provided additional performance from both systems.

Results

Figures 4 and 5 show the variation of iteration time (in seconds) against the cells per core for the 143M and 280M cell meshes respectively. These cells per core correspond to 480, 960 and 1920 physical cores for the three runs on each mesh. A number of statements can be made about these results:

- Firstly the Cray XC30 shows excellent strong scaling down to 73k cells per core (1920 cores) and would likely continue beyond due to the high performing interconnect.
- The M5.24xlarge is on average slower than the Cray XC30 due to a combination of 25Gbps Ethernet interconnect and 2.5 GHz clock speed (2.7 GHz on the Cray XC30)

FIGURE 4 Scaling performance of AWS instances and the Cray XC30 using the 143M cell model.

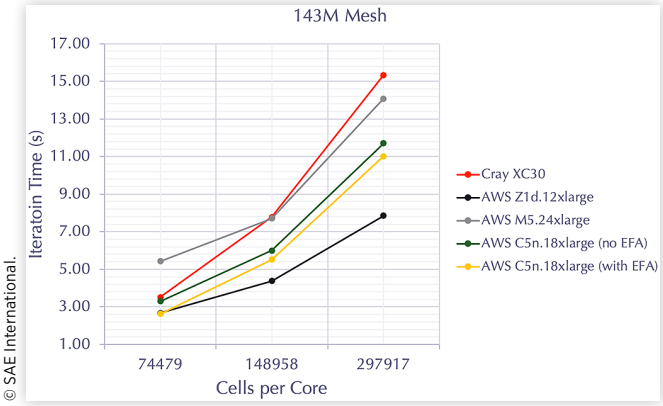
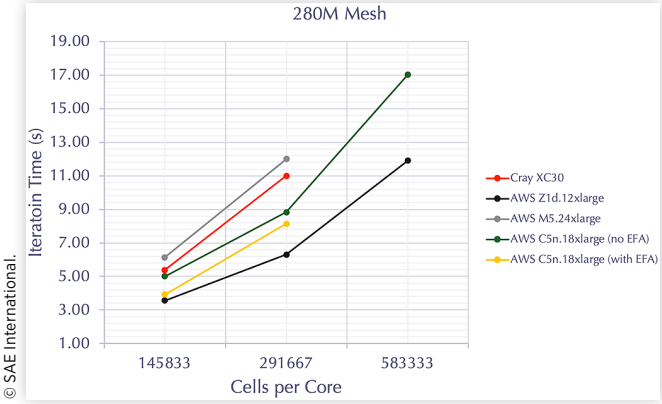


FIGURE 5 Scaling performance of AWS instances and the Cray XC30 using the 280M cell model.



- The sustained 4.0 GHz AWS custom Intel scalable Z1D.12xlarge instance delivers the fastest iteration by some margin at high cells per core counts (480 & 960) but the higher latency interconnect takes away some of this advantage at lower cells per core and is only slightly faster than the Cray XC30 at 73k cells per core. A visual inspection of the trend suggests that the Cray XC30 would become faster at lower cells per core counts.
- The C5n.18xlarge compute optimized instance with 100Gbps interconnect (running at 3.5 GHz) lies in between the M5.24xlarge, Cray XC30 and the Z1d.12xlarge but whilst the scaling of the Z1d begins to tail off the C5n shows a better trend and by 73k cells per core it is now the same or marginally quicker than the Cray XC30 and close to Z1d. Again it is also evident from the scaling that the Cray XC30 would also be faster than the C5n.18xlarge instance at lower cells per core.
- Finally it can be seen that when EFA is turned on for the C5n.18xlarge instance then there is constant iteration time reduction across all core counts. This is most noticeable at the lowest cells per core where communication time contributes to a larger proportion of the total iteration time. With EFA the scaling performance is much closer to the performance seen with the Cray XC30 - which was chosen to represent a typical on-premise solution with custom HPC interconnect.

Extrapolating these times for running 2s of physical solve time on the 280M cell mesh using 1920 cores results in a time of **4hrs** for the Z1d.12xlarge instance, **4.4hrs** for the C5n.18xlarge instance with EFA, **6.1hrs** for the Cray XC30 and **6.8hrs** for the M5.24xlarge instance. This is in contrast to using only 480 cores (i.e. the size of a small companies cluster) of **19hrs** using C5n.18xlarge.

Cost

AWS EC2 instance costs vary depend on region and whether the instance is used on-demand, via spot or using an upfront reservation. As of 22nd November 2019 the on-demand cost in EU(Ireland) region of using 80 Z1d.12xlarge instances, 54

C5n.18xlarge instances and 40 M5.24xlarge instances, which correspond to the number required to run on 1920 cores for the extrapolated simulation times given previously would be $1,597.44, $1,043.53 and $1396.99 respectively.

However as of 22nd November 2019 the spot-price for the same instances in EU(Ireland) region would result in a total price of $479.23, $331.95 and $470.94. This is on average a 70% reduction in cost.

It is not the aim of this paper to discuss the cost differences between the Cray X30 (which as for any on-premise facility would be very difficult to calculate) and AWS but rather to illustrate the cost/run-time choice that all customers face.

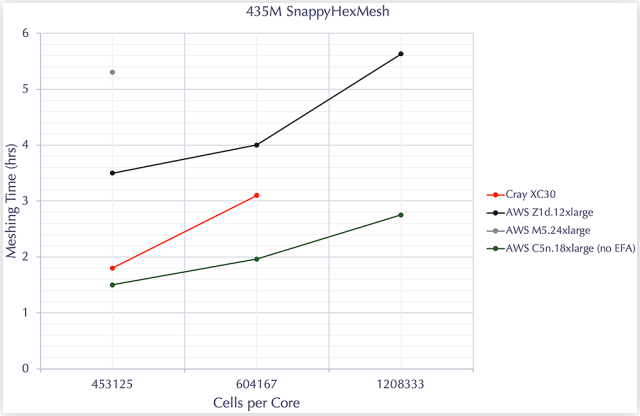
Meshing

Figure 6 shows the total meshing time using the snappyHexMesh method with OpenFOAM against the cells per core (i.e. number of cores). A racing car model was meshed in snappyHexMesh resulting in 435M cells. It can be seen that in comparison to the solver performance previously discussed, both the M5.24xlarge and Z1d.12xlarge instances are slower than the Cray XC30, which is likely due to the 25Mbps Ethernet interconnect which suffers from worse bandwidth and latency. Adding additional cores to M5 and Z1d does not significantly improve the speed. The C5n instance shows much improved meshing speed, likely due to the improved interconnect however scaling drops off at 1920 cores. Unfortunately at the time of the mesh testing EFA was not available although future work will be to assess whether this provides any improvements.

Additional Considerations

We have not considered post-processing as part of this study and we have treated the meshing and solver steps as two separate activities. However the authors realise that a production CFD workflow is typically a fully automated process with the mesh, solve and post-processing happening under a single submission script. Thus the extrapolated timings in this study would likely be longer if post-processing and the steps between the mesh and solve were included i.e. the optimum number of cores for meshing is different to solve and thus there would be time needed to redistribute the mesh between these core counts and potentially different instance types.

FIGURE 6 Meshing time compared to cells per core for Cray XC30 and several AWS instance types



Conclusions

This study has shown that Amazon Web Services can provide the HPC environment to enable greater use of high-fidelity CFD methods by allowing higher core counts to reduce turnaround time. With the correct instance type - which potentially differs between meshing and solving, AWS was competitive against a Cray XC30 supercomputer for meshes up to 280 million cells and core counts up to 1920 cores. It is noted however that this Cray XC30 machine uses older generation processor units (Intel Ivybridge) and it is expected that with equivalent processor family (Intel Skylake) then the runtime performance of the Cray system would have been even greater than reported here. This study has shown that it is possible to achieve the speed and scaling on AWS but it is also appreciated that for many CFD users the work is the final mile to move the full process to AWS. Future work will be to demonstrate the performance at higher core counts and include post-processing elements of the CFD process.

Acknowledgments

The authors would like to thank Vangelis Skarpedas of BETA-CAE Systems for the meshes used in this study.

References

1. Spalart, P.R., Jou, W.H., Strelets, M., and Allmaras, S.R., "Comments on the Feasibility of LES for Wings and on a Hybrid, RANS/L ES Approach," *Advances in DNS/LES, Proceedings of 1st AFOSR International Conference on DNS/LES* 1:137-147, 1997.
2. Jakirlić, S., Jester-Zurker, R., and Tropea, C., "Report on 9th ERCOFTAC/IAHR/COST Workshop on Refined Turbulence Modelling," ERCOFTAC Bulletin, Darmstadt University of Technology, 2002, 36-43.
3. Jakirlić, S. and Maduta, R., "Extending the Bounds of Steady RANS Closures: Toward an Instability-Sensitive Reynolds Stress Model," *International Journal of Heat and Fluid Flow* 51:175-194, 2015.
4. Manceau, R., "Recent Progress in the Development of the Elliptic Blending Reynolds-Stress Model," *International Journal of Heat and Fluid Flow* 51:195-220, 2015.
5. Olsen, M.E., "Reynolds-Stress and Triple-Product Models Applied to Flows with Rotation and Curvature," in *46th AIAA Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, Reston, Virginia, June 2016.
6. Stoellinger, M., Roy, R., and Ashton, N., "Application of an Elliptic Blending Reynolds Stress Model in Attached and Separated Flows," in *22nd AIAA Computational Fluid Dynamics Conference*, 2015.

7. Ashton, N. and Stoellinger, M.K., "Computation of Turbulent Flow in a Rotating Pipe using the Elliptic Blending Reynolds Stress Model," in *46th AIAA Fluid Dynamics Conference*, June 2016, 1-11.
8. Haase, W., Braza, M., and Revell, A., "DESider - A European Effort on Hybrid RANS-LES Modelling," . In: *Vol. 103 of Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, (Springer-Verlag, 2007).
9. Schwamborn, D. and Strelets, M., "ATAAC - An EU-Project Dedicated to Hybrid RANS-LES Methods," . In: *Progress in Hybrid RANS-LES Modelling, Vol. 117 of Notes on Num. Fluid Mech. and Multidisc. Design*, 2012, 59-75.
10. Fröhlich, J. and von Terzi, D., "Hybrid LES/RANS Methods for the Simulation of Turbulent Flows," *Progress in Aerospace Sciences*, 44, 2008, 349-377.
11. Spalart, P.R., Deck, S., Shur, M.L., Squires, K.D. et al., "A New Version of Detached-Eddy Simulation, Resistant to Ambiguous Grid Densities," *Theor. Comput. Fluid Dynam.* 20(3):181-195, 2006.
12. Girimaji, S.S., "Partially-Averaged Navier-Stokes Model for Turbulence: A Reynolds-Averaged Navier-Stokes to Direct Numerical Simulation Bridging Method," *Journal of Applied Mechanics* 73(3):413, 2006.
13. Menter, F.R. and Egorov, Y., "Description of SST-SAS Model," Tech. rep., Ansys, 2009.
14. Mockett, C., Fuchs, M., Garbaruk, A., Shur, M. et al., "Two Non-Zonal Approaches to Accelerate RANS to LES Transition of Free Shear Layers in DES," *Progress in Hybrid RANS-LES Modelling* 187-201, 2015.
15. Shur, M.L., Spalart, P.R., Strelets, M.K., and Travin, A.K., "An Enhanced Version of des with Rapid Transition from RANS to les in Separated Flows," *Flow, Turbulence and Combustion* 95(4):709-737, 2015.
16. Deck, S., "Recent Improvements in the Zonal Detached Eddy Simulation (ZDES) Formulation," *Theoretical and Computational Fluid Dynamics* 10, 2011.
17. Ashton, N., West, A., Lardeau, S., and Revell, A., "Assessment of RANS and DES Methods for Realistic Automotive Models," *Computers & Fluids* 128:1-15, 2016.
18. Ashton, N., Unterlechner, P., and Blacha, T., "Assessing the Sensitivity of Hybrid RANS-LES Simulations to Mesh Resolution, Numerical Schemes and Turbulence Modelling within an Industrial CFD Process," 4:1-11, 2018.
19. Guilmineau, E., Deng, G., Leroyer, A., Queutey, P., Visonneau, M., and Wackers, J., "*Assessment of Hybrid RANS-LES Formulations for Flow Simulation around the Ahmed Body*," *Computers & Fluids*, 2017.
20. Lietz, R., Larson, L., Bachant, P., Goldstein, J. et al., "An Extensive Validation of an Open Source Based Solution for Automobile External Aerodynamics," SAE Technical Paper 2017-01-1524, 2017, https://doi.org/10.4271/2017-01-1524.
21. Jakirlic, S., Kutej, L., Unterlechner, P., and Tropea, C., "Critical Assessment of Some Popular Scale-Resolving Turbulence Models for Vehicle Aerodynamics," *SAE Int. J. Passeng. Cars - Mech. Syst.* 10(1):235-250, 2017, https://doi.org/10.4271/2017-01-1532.
22. Gaylard, A.P., Kabanovs, A., Jilesen, J., Kirwan, K., and Lockerby, D.A., "Simulation of Rear Surface Contamination for a Simple Bluff Body," *Journal of Wind Engineering and Industrial Aerodynamics* 165, 2017(January 2016):13-22.
23. Ashton, N., Unterlechner, P., and Blacha, T., "Assessing the Sensitivity of Hybrid RANS-LES Simulations to Mesh Resolution, Numerical Schemes and Turbulence Modelling within an Industrial CFD Process," SAE Technical Paper 2018-01-0709, 2018, https://doi.org/10.4271/2018-01-0709.
24. Ashton, N. and Revell, A., "Comparison of RANS and DES Methods for the DrivAer Automotive Body," SAE Technical Paper 2015-01-1538, 2015, https://doi.org/10.4271/2015-01-1538.
25. Islam, A., Gaylard, A., and Thornber, B., "A detailed Statistical Study of Unsteady Wake Dynamics from Automotive Bluff Bodies," *Journal of Wind Engineering and Industrial Aerodynamics* 171:161-177, October 2017.
26. Ashton, N. and Revell, A., "Key Factors in the Use of DDES for the Flow Around a Simplified Car," *International Journal of Heat and Fluid Flow* 54:236-249, 2015.
27. Robertson, E., Choudhury, V., Bhushan, S., and Walters, D.K., "Validation of OpenFOAM Numerical Methods and Turbulence Models for Incompressible Bluff Body Flows," *Computers and Fluids* 123:122-145, 2015.
28. Travin, A., Shur, M., Strelets, M., and Spalart, P.R., "Physical and Numerical Upgrades in the Detached-Eddy Simulation of Complex Turbulent Flows," in *Proceedings of the 412th Euromech Colloquium on LES and Complex Transitional and Turbulent Flows*, Munich, 2000.
29. Chevalier, C. and Pellegrini, F., "PT-Scotch: A Tool for Efficient Parallel Graph Ordering," *Parallel Computing* 34(6-8):318-331, 2008.
30. Ashton, N., "Recalibrating Delayed Detached-Eddy Simulation to Eliminate Modelled-Stress Depletion," in *23rd AIAA Computational Fluid Dynamics Conference*, 2017, 1-14.

ISSN 0148-7191