Journal of Computational Physics 434 (2021) 110187

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Parallel load-balancing for combustion with spray for large-scale simulation

A. Thari [a,*], N.C.W. Treleaven [b], M. Staufer [c], G.J. Page [a]

[a] Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, LE11 3TU, United Kingdom
[b] CERFACS, CFD Team, 42 Av. Gaspard Coriolis, 31057, Toulouse, France
[c] Rolls-Royce Deutschland Ltd & Co KG, 15827 Blankenfelde-Mahlow, Germany

ARTICLE INFO

Article history:
Available online 19 February 2021

Keywords:
Task-based parallelism
Asynchronous multi-physics
Shared-memory communication
Eulerian-Lagrangian
Combustion with spray

ABSTRACT

An asynchronous task-based Eulerian-Lagrangian approach for efficient parallel multi-physics simulations that can scale for arbitrary large number of particles and non-uniformly distributed particles is presented. The parallel methodology is based on a task-based partitioning of the multi-physics problem, where each single-physics problem is considered as a task and carried out using its own set of processes. This allows the two problems to solve their governing equations concurrently; therefore, hiding the computational cost incurred of solving an additional physical solver. Applications to complex breakup mechanism leading to highly dynamic computational loading and three-dimensional swirl combustion chamber with reacting flow/spray with extremely uneven particle distribution demonstrate the improved parallel efficiency and great potential of the presented approach.

1. Introduction

Large scale computational problems are commonplace in a wide range of application areas such as the aerospace, automotive and oil industries. Parallel computing is a vital tool to ensure that realistic and accurate computational simulation may be conducted within a reasonable time frame. Parallel computing for numerical simulation such as Computational Fluid Dynamics (CFD) is usually based on the domain decomposition approach. That is, the global problem is divided into a number of local problems (or partitions) using a spatial decomposition and each are solved semi-independently on a computing process. The efficiency of this approach is impacted by any uneven work-load across the partitions and delays caused by data transfer between partitions. For CFD of aerodynamic problems using a finite volume discretisation, domain decomposition is well established and can scale to thousands of processing units. However, for multi-physics problems which solve a variety of equations – some which may not be in finite volume form – then efficient parallelisation is more challenging [20,37] and is the focus of this work.

The general model of large-scale computer systems consists of several independent computer nodes each with ownership of local memory. The nodes within a computer system are connected to a network that provides the ability to communicate among the computer nodes. Each node can either be a single processor or multiprocessor systems where each processor contains several cores. All cores have equal access to its node's memory. Nowadays, large-scale clusters are moving towards

* Corresponding author.
E-mail address: Ali.Thari@lboro.ac.uk (A. Thari).

https://doi.org/10.1016/j.jcp.2021.110187
0021-9991/© 2021 Elsevier Inc. All rights reserved.

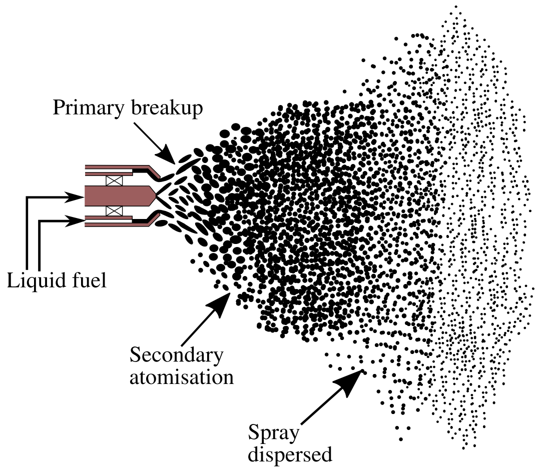

Fig. 1. An Air-blast atomiser sketching the breakup process from the liquid film into tiny dispersed droplets which eventually evaporate.

larger core counts per node[1] which implies more cores sharing same memory and less memory- and communication bandwidth per core. The Message Passing Interface (MPI) library is commonly used to communicate among cores irrespectively whether they share memory or sit at a different node. A simulation program is launched by a number of MPI processes that is usually equal to the total number of cores considered: a practice used in this work. The MPI rank is frequently used as shorthand for the MPI process ID.

A good multi-physics example is CFD modelling of gas turbine combustion. The flow-field in a combustor is highly turbulent, chemically reacting and consists of multiple fluid phases and has significant effects from radiative heat transfer. Liquid fuelled engines work by using a fuel injector to create a spray of fuel droplets, whilst the aerodynamic design creates a high swirl to provide a stable combustion zone for gaseous fuel generated from evaporation. From the fuel injector, there is primary and secondary break-up of fuel ligaments into droplets (Fig. 1), and these may then experience evaporation, condensation, collision and coalescence, before finally feeding the gaseous phase and are combusted. Whilst the initial fuel ligaments may be conceivably captured by an Eulerian grid based approach using techniques such as Volume of Fluid, the number and size of droplets are impractical to be resolved by an Eulerian grid, and so a Lagrangian particle tracking method is normally adopted. With ever tightening legislation regarding NOX and soot emissions from aeroengine gas turbines, then it is critical to have fast and accurate simulation processes to assess candidate designs. This can only be achieved if the methods exhibit good parallel scaling for very large numbers of fuel droplets.

This multi-physics combustion problem is frequently solved by means of a coupled Eulerian-Lagrangian approach [4,23,1]. The reacting gas flow is solved using a finite-volume method on an Eulerian grid while the fuel droplets/particles are tracked using a Lagrangian approach. Although such methodologies overcome issues related to extreme mesh resolutions, the droplet count near the injector is of the order of 50 million [4,5], while further downstream they evaporate into vapour. This is one of the problems for parallel computing load balancing using a geometric partitioning: within a given region the droplet density may be very different to the finite-volume density (e.g. [36]). Although it is possible for a partitioning algorithm to use a locally varying weighting to compensate, it is challenging to determine these a priori. And, in our case with a dynamically changing loading where droplets constantly are injected, atomised, moved from partition to another and eventually eliminated due to evaporation would require a dynamic repartitioning technique which can have unacceptable overheads [33].

The Eulerian-Lagrangian algorithm is generally implemented within a single program and the Eulerian and Lagrangian equations are solved sequentially [19] (i.e. within a time-step, the flow-field update is computed, then the new flow-field passed to the particle tracking and the particle update is computed). A straightforward parallelisation can be achieved by assigning all particles within one Eulerian sub-domain to its corresponding computing unit. This method is also referred as the spatial partitioning approach. Data transfer for the Lagrangian part only occurs when particles move between sub-domains. This has been a standard approach for combustion applications [29] and as noted earlier can result in unbalanced loading and poor parallel efficiency in applications where the particles are clustered within only a few number of sub domains (e.g. near the fuel injector) [4,6]. The sequential approach ignores the inherent parallelism of the problem where both flow and particle updates could be computed simultaneously using the conditions at the beginning of the time step. This is a key element of the current work.

A second parallel approach divides the droplets equally among all computer cores ignoring their spatial location [34,35,22,36]: this is referred to as a particle sharing algorithm. Although it ensures load balancing, this method requires

[1] https://www.archer2.ac.uk/about/hardware.html.

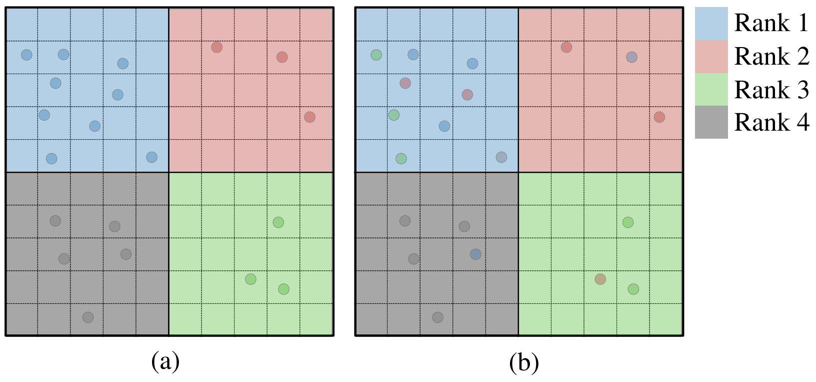

Fig. 2. (a) Spatial partitioning approach where each rank tracks the particles that are located within the same sub domain. (b) Particle sharing algorithm would only ensure an equal amount of particles of each rank regardless of their spatial location. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

collective communication in order to keep track of which Eulerian domain partition the droplets reside. On computations with large core counts, this can have an adverse effect on scalability. Some improvements have been made by maximising the ownership fraction of the droplets and processor ranks using an equalization algorithm [36]. Fig. 2 illustrates the two parallel algorithms for particle tracking, where Fig. 2b illustrates the particle sharing algorithm. In this figure the coloured boxes represent the geometric partition of the grid and each colour represents a rank, whilst the circles represent the particles which use the same colour coding to show in which rank they reside. Overwhelmingly the method of Fig. 2a is the most common, due to its conceptual and practical simplicity. However, neither of the methods in the figure are suitable for large scale computations, especially moving towards exascale.

As mentioned earlier, the benefit of a spatial based algorithm is that it reduces communication by having the local flow information and particles on the same processor. When the physics of the problem does not impose a severe non-uniform distribution of particles, this approach is very efficient. Work by Capecelatro and Desjardins [8] showed good scale-up for cases involving 382 million particles with 4096 CPUs. On the other hand, the work presented by Kaludercic [24], Pankajakshan et al. [29] has demonstrated its significant limitation when considering situations with uneven distribution of particles across the domain. One way to overcome this is by the mirror domain decomposition by Darmana et al. [12]. All cores receive a copy of the data, but each processor is only responsible for computing updates on a subset of particles – after each time step the updates are then synchronised. Unfortunately, the method was shown to be efficient only up to 32 processors due to the increased communications in the synchronisation stage. In addition, for large numbers of particles and large core counts, each processor is requiring a significant amount of memory to store information on particles that it is not computing. Another approach suggested by Frank et al. [14] is to have a servicing core for each calculation core. The servicing cores make sure to distribute an even amount of workload to each calculation core. It was demonstrated to be viable up to 64 processors, although the parallel efficiency had reduced down to 40%. Zhao et al. [40] presented a dynamic load balancing approach to alleviate the imbalance issue. The performance gain was limited after 8 processors due to the increased overhead of the repartitioning process. Another critical element to this approach is the required collective communication. A bookkeeping procedure must be carried out before each data exchange among all cores. This is to provide each CPU information on how many particles have entered its grid domain and from which ranks. A large number of these communications will return a null value because no particles have passed to its corresponding partition. Sitaraman and Grout [36] outlines this issue and presents a different method employing one-sided remote memory access (RMA) communication; an approach which is introduced in the MPI 2 standard [17].

A similar communication strategy has been suggested by Buaria and Yeung [7] and Fanfarillo and Del Vento [13] which employ the Partitioned Global Address Space (PGAS). This is simply a form of Remote Direct Memory where the memory address space is logically partitioned. Each processor has its own local portion of the global memory. A processor can retrieve information from another processor by knowing the data locality. Communication is reduced to a minimum through the utilisation of PGAS. This has been showcased for DNS problems incorporating 300 million particles and thousands of cores. However, the work only examines cases with uniform particle distribution meaning that the issue with work imbalance is not considered.

A few papers have examined a hybrid message-passing/shared memory approach (e.g. MPI-OpenMP) [39]. Here, the Eulerian solver benefits from the distributed memory through MPI and the Lagrangian solver takes advantage from the shared memory multiprocessing. The basic idea is to launch a specific number of OpenMP threads to solve the Lagrangian equations once the MPI processes have advanced the Eulerian solver one time step. The parallelism of the Lagrangian solver is limited to declared OpenMP constructs, which scales well for problems with a much small number of particles as demonstrated by [39] with applications involving up to 100,000 particles.

Amritkar et al.[2] present a parallel framework based solely on shared memory using OpenMP. This study combines the different parallel modes, i.e. domain decomposition and particle sharing algorithms, by exploiting the flexibility offered by

OpenMP. That is, the threads alternate between running across the blocks and particle numbers. Changing modes of parallelism is relatively cost-free in OpenMP where it would have involved large overheads with MPI. The work demonstrated appreciable improvements in performance compared to a pure MPI implementation. However, this approach is limited to systems with shared-memory configuration implying that the approach is not suitable for simulation run across multiple computer nodes. Houzeaux et al. [19] employs a hybrid approach to solve the Eulerian-Lagrangian problem. Instead of having a single code execution, a multi-code approach where each solver/code acquires separate computer nodes, is applied to enable overlapping between the Eulerian and Lagrangian calculations. OpenMP is used for the main loops in the Lagrangian Particle Tracking (LPT) solver within each MPI subdomain. Also, it applies a Dynamic Load Balancing library (DLB) which ensures a reasonable workload across the threads at run time. That is, DLB tracks every MPI process and when it detects that some are waiting or blocked, it will lend their resources/cores to spawn more OpenMP threads within the same node. The approach has been tested for large scale HPC application including up to 64 computational nodes with 16 cores per node. While applying DLB showed distinct improvement in all the cases, the computational gain between single-code and multi-code was not as definite. As aforementioned, parallelism of OpenMP applications is limited to selected sections in the code and can therefore become significantly less efficient than using a pure MPI implementation. Also, the single-code approach benefits from better data locality with respect to exchanging data between the Eulerian and Lagrangian solvers, while its counterpart requires a send-receive communication pair each time-step.

The combination of large droplet count and the poor parallel efficiency for Eulerian-Lagrangian combustion applications lead the community to use an approximation by bundling particles into parcels. That is, each parcel (represented by a single computational particle) represents a number of real particles which share similar features, such as mass, radius and temperature. These will be generated by preserving mass conservation and represent a considerably lower count compared to the actual number of droplets [4,6,23]. This obviously has a large benefit in terms of computation time, but at the cost of simulation accuracy. Tracking each individual droplet in a parallel computing framework with a domain-decomposition and spatial particle algorithm is not possible for realistic problems as the uneven load balancing will stop the parallel scaling. However, the diffusion of flames is controlled by their stretching that is directly influenced through the mixing [38,28]. Large-Eddy-Simulations (LES) is generally considered essential to model such turbulence mixing and combustion dynamics even for industrial applications [31,6,23]. Hence, a parcel approach for LES simulations is less suitable since such a method provides an averaged particle trajectory contrary to the instantaneous field provided by the LES. Apte et al. [4] presented a hybrid particle-parcel approach to overcome this and towards a more LES suited particle tracking methodology.

The novel contribution of this work is a paradigm change for parallelisation of multi-physics problems such as Eulerian-Lagrangian spray modelling in combustion applications. We will present a task parallel framework exploiting shared memory communication and large core counts per node. The novelty is to split the Eulerian-Lagrangian solvers into two separate tasks, which allows the Eulerian and Lagrangian parts to carry out their respective operations concurrently. The interchange of data between the solvers is conducted using one-sided shared memory communication and avoids the normal send-receive communication pair. The work presented here currently focuses on single-node performance, which represents a crucial first step towards achieving efficient parallel computing across multiple nodes. The aim of this work is to present a parallel framework that can achieve almost perfect single-node parallel efficiency for complex multi-physics problems by maintaining an even load balancing for arbitrary non-uniform particle distribution. This can then be expanded to multiple node machines that will enable the community to conduct cost-efficient large-scale parallel simulations for liquid fuelled aeroengine combustors.

In this work, two different parallelisation of an industrial CFD code will be evaluated to validate and verify the novel parallelisation concept. The first is the baseline code which has been extensively used by industry for simulating reacting flows and designing gas turbine aeroengines: `PRECISE-UNS` [9,3,41]. It features the common coupling methodology between the Eulerian and Lagrangian phase where the code advances selectively one time-step with the Eulerian gas field and subsequently a time-step with the Lagrangian liquid phase. The parallel methodology is also based on the common spatial particle algorithm which suffers excessively from parallel scalability for large scale simulations. The second version is strongly based on the latter in terms of physical modelling. However, the Eulerian and Lagrangian parts have been split into two separate processes that communicate together over the course of a simulation. This version will also feature the new parallel methodology presented in this work, namely a two-way coupled Asynchronous Task-based Eulerian-Lagrangian solver (ATEL). This work will be validated by comparing two different test cases to the baseline code, but also to available experimental and CFD data.

The paper is structured as follows. Section 2 provides a description of the mathematical models and the two-way coupling methodology between the Eulerian and Lagrangian phase. In Section 3, the parallel framework of ATEL is explained in great detail which begins the asynchronous and parallel task-based coupling approach. This is followed by a description of the shared memory method employed for the communication between the parallel tasks. Finally, the mapping and synchronisation between the flow and spray ranks are explained. This is followed by Section 4 where the first study-case is carried out on two-dimensional nitrogen chamber with a complex breakup mechanism. Section 5 presents the second case of three-dimensional combustor with reacting flow. Both test-cases are validated, rigorously tested and analysed for parallel performances. Concluding remarks are given in Section 6.

2. Mathematical models

The section describes the governing equations of both the gaseous and liquid phases. A Large-Eddy-Simulation (LES) approach is used with unresolved turbulence modelled by the Smagorinsky Subgrid Scale (SGS) model. The two-way coupling between the two phases is included. The relative velocity between the gas and droplet is incorporated to calculate the forces on the droplets, while each droplet is treated as a point-source through the mass, momentum and energy gas equations.

2.1. Gas phase equations

The Eulerian flow field is governed by the three-dimensional governing LES Favre-filtered Navier-Stokes equations which represents the mass, momentum and energy equations,

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial\left(\overline{\rho}\widetilde{u}_i\right)}{\partial x_i} = \dot{m}_d \tag{1}$$

$$\frac{\partial \overline{\rho}\widetilde{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{\rho}\widetilde{u}_i\widetilde{u}_j\right) + \frac{\partial \overline{p}}{\partial x_j} = \frac{\partial}{\partial x_i}\left[\overline{\rho}\left(\widetilde{u_i u_j} - \widetilde{u}_i\widetilde{u}_j\right) + \tau_{ij}\right] + S_{i,d}, \tag{2}$$

$$\frac{\partial \overline{\rho}\widetilde{h}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho}\widetilde{u}_i\widetilde{h}\right) = \frac{\partial \overline{p}}{\partial t} + \overline{u_i\frac{\partial p}{\partial x_i}} - \frac{\partial}{\partial x_i}\left[\overline{q_i} + \overline{\rho}\left(\widetilde{u_i h} - \widetilde{u}_i\widetilde{h}\right)\right] + \overline{\tau_{ij}\frac{\partial u_i}{\partial x_j}} + Q_d \tag{3}$$

where u_i, p, ρ, q_i and τ_{ij} are the gas velocity, pressure, density, energy flux and viscous tensor, respectively. The bar $(-)$ and tilde $(\sim)$ represent the unweighted and density weighted average, respectively. The energy Eq. (3) solves for the absolute enthalpy which includes the chemical formation enthalpy of species. The mass, momentum and energy source terms ($\dot{m}_p$, $\dot{S}_p$, Q_p) due to the fuel droplets will be discussed further below. Further details regarding the governing equations and turbulence modelling are given in Appendix A.

2.1.1. Combustion model

Combustion simulations with complex fuel like kerosene or diesel can involve hundreds of different species which is computationally impractical to solve for each of their respective transport equations. Therefore, chemical reduction and combustion modelling are often employed to compute the combustion processes in a more efficient manner [28,32]. In this work, a Flamelet-Generated-Manifold (FGM) with a presumed beta-Probability Density Function (β-PDF) combustion model is used, which employs a pre-computed tabulation of various thermodynamic, transport and turbulence-chemistry interaction properties of different premixed flames. The mixture fraction Z and progress variable ϕ and their respective variances are used to construct the FGM look-up table. The mixture fraction Z ranges in the interval from pure air to pure fuel $[0, 1]$. The progress variable ϕ represents the reaction progress within the flame and is defined as zero at the unburnt region. There are different ways to define the progress variable [28] and the one used here is based on the mass fraction of CO_2, H_2O and H_2 (see Appendix A). Additionally, the variances of each of the two control variables are solved as transport equations to incorporate the turbulence chemistry interaction. Effectively, this means that the combustion model adds four additional transport equations and is presented in Appendix A. When a spray model (see Section 2.2) is incorporated to the LES filtered control variable of the mixture fraction must be modified to include a source term $\dot{\omega}_Z$ that represents the amount of fuel spray that has been evaporated in the last timestep as calculated by the Lagrangian solver. The modified transport equation of mixture fraction becomes:

$$\frac{\partial \overline{\rho}\widetilde{Z}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho}\widetilde{u}_i\widetilde{Z}\right) = \frac{\partial}{\partial x_i}\left[\nu_t\frac{\partial \widetilde{Z}}{\partial x_i}\right] + \dot{\omega}_Z \tag{4}$$

2.2. Spray model

The Spray model is included in the Lagrangian spray where each droplet's trajectory is tracked using

$$m_d\frac{du_{i,d}}{dt} = F_{i,d} + G_{i,d} + H_{i,d}, \tag{5}$$

where m_d and $u_{i,d}$ are the mass and velocity of droplet d. The drag force $F_{i,d}$ is calculated by

$$F_{i,d} = C_d\left(\mathrm{Re}_d\right)\frac{1}{2}\rho|\mathbf{u}_{\mathrm{relv}}|u_{i,relv}S_d, \tag{6}$$

where $\mathbf{u}_{relv}$ is the relative velocity between the gas and liquid phase. The frontal area of the fuel droplet is denoted as S_d and Re_d is the Reynolds number based on the droplet diameter D_d and relative velocity

$$\mathrm{Re}_d = \frac{\rho|\mathbf{u}_{\mathrm{relv}}|D_d}{\mu}, \tag{7}$$

The drag coefficient C_d is a function of droplet Reynolds number Re_d [25]

$$C_d = \begin{cases} 24\left(1+0.15\mathrm{Re}_d^{0.687}\right)/\mathrm{Re}_d & \mathrm{Re}_d \leq 1000 \\ 0.424 & \mathrm{Re}_d > 1000 \end{cases} \tag{8}$$

The virtual force $G_{i,d}$ accounts for the additional fluid surrounding a droplet which needs to be accelerated when the droplet accelerates

$$G_{i,d} = -C_{vm}\rho\Omega\frac{d\left(u_{i,relv}\right)}{dt}, \tag{9}$$

where the virtual mass constant, $C_{vm} = 0.5$. The body forces on the droplet $H_{i,d}$ are neglected in this work.

2.2.1. Evaporation model

The evaporation model employed in this work is proposed by Chin and Lefebvre [10], where the mass of each particle is controlled by the evaporation rate

$$\frac{dm_d}{dt} = 2\pi D_d\left(\frac{\lambda}{C_p}\right)_g \ln\left(1+B_m\right), \tag{10}$$

where λ and C_p are the mean thermal conductivity and specific heat of the gas, respectively. The mass transfer number B_M is defined as

$$B_M = \frac{Y_{F_s}}{1-Y_{F_s}}, \tag{11}$$

where the mass fraction of fuel vapour at the droplet surface is given by

$$Y_{F_s} = \left(1+\left(\frac{p}{p_{F_s}}-1\right)\frac{M_A}{M_F}\right)^{-1}, \tag{12}$$

where p, p_{F_s}, M_A and M_F are the air pressure, fuel vapour pressure based on the Clausius-Clapeyron relation and molecular weight of air and fuel, respectively. As it is assumed that the droplet is spherical and the effect of convective cooling due to relative velocity is negligible, the fuel droplet temperature can be assumed to be constant and equal to its surface temperature.

$$\frac{dT_d}{dt} = \frac{Q - Q_c}{C_{p,f}m_d} \tag{13}$$

Eq. (13) describes the temperature change of the droplet's surface over a time duration dt. The heat transferred from air to fuel and the heat absorbed through evaporation are denoted as Q and Q_c, respectively.

$$Q = 2\pi D_p\left(T - T_d\right)\ln\left(1+B_M\right)/B_M \tag{14}$$

$$Q_c = \frac{dm}{dt}L \tag{15}$$

2.2.2. Spray breakup modelling

This work employs a stochastic atomisation model based on the Fokker-Planck differential equation. The solution to such equation leads to a distribution function for $x = \ln(r)$, where r is the droplet radius [16,4].

$$T\left(x,t+1\right) = \frac{1}{2}\left[1+\mathrm{erf}\left(\frac{x - x_d - \langle\xi\rangle}{\sqrt{2\langle\xi^2\rangle}}\right)\right]. \tag{16}$$

The first ξ and second moments ξ^2 need closure and the expressions derived by [4,5] are employed

$$\langle\xi\rangle = K\ln\left(\frac{\mathrm{We}_{crit}}{\mathrm{We}_d}\right) \tag{17}$$

$$\langle\xi^2\rangle = -\langle\xi\rangle\mathrm{We}_d, \tag{18}$$

where We_{crit} and We_d are the critical and droplet Weber numbers. The constant K is set to 0.6. The breakup of a parent droplet will take place when $\mathrm{We}_d > \mathrm{We}_{crit}$ and $t > t_{\mathrm{bu}}$, where t is the elapsed time of the droplet and breakup frequency $1/t_{\mathrm{bu}}$

$$t_{\mathrm{bu}} = \sqrt{1/3}\sqrt{\frac{\rho_d}{\rho}}\frac{r_{\mathrm{j}}}{|\mathbf{u}_{\mathrm{relv}}|} \tag{19}$$

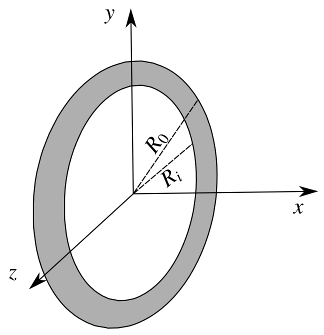

Fig. 3. A sketch of the fuel nozzle ring from which the droplets will be released at each time-step.

The distribution function spans over a wide range, which implies that newly formed droplets have a wide range of droplet sizes which makes this atomisation model sufficient for both primary and secondary breakup. A detailed derivation and discussion of this model are provided by [16,4]. This method is generally very expensive because the droplet count can quickly explode. To overcome this problem, the parcel approach can be considered where each parcel contains a number density N_d which provides the number of droplets it actually represents. With breakup, the parcel count can still become excessive, and therefore, another approximation would be to sample the diameter of the parcels and update their number densities without actually creating additional droplets to track. However, this averaged based approach is in contrast to LES applications [4]. An alternative solution is to consider the droplets partly atomised before being injected to the domain. A Rosin-Rammler distribution can be employed to provide a distribution of droplet diameters using a mean diameter d_{32}, assuming that such droplet sizes are more influenced by the evaporation rate rather than atomisation [9,21].

2.3. Spray injection

The spray injection is carried out by defining a number of injection locations N_{sl} distributed on a fuel nozzle ring as shown in Fig. 3. The number of fuel nozzle rings, their locations, thicknesses and orientations are defined at the beginning of a simulation, and the injection of all droplets will take place at each time step. The thickness of the ring is used to randomise the droplet's location in the radial and azimuthal directions. Each injection location $N_{i,\mathrm{sl}}$ will inject droplets according to a defined distribution of droplet diameters, for example, a Rosin-Rammler distribution. That is, if the distribution has N_{bin} bin sizes along with N_{sl} injection locations and N_{fnzo} fuel nozzle rings, the total number of parcels injected per time step is

$$N_{d,\Delta t} = N_{\mathrm{fnzo}} N_{\mathrm{sl}} N_{\mathrm{bin}} \tag{20}$$

The inlet spray velocity is defined using a cylindrical coordinate system with reference to the fuel nozzles. That is, for each nozzle an axial V_x, tangential V_θ and radial V_r velocity components need to be defined.

2.4. Two-way coupling Eulerian-Lagrangian model

A two-way coupled framework is relevant for vaporizing fuel spray where a portion of the mass liquid fuel (see Eq. (10)) converts to vapour at each timestep. This fuel vapour needs to be added as a source term in the mass conservation equation. The mass source term in Eq. (1) is defined as

$$\dot{m}_d = \sum_{i=1}^{N_d} \left(\frac{dm_d}{dt} \right), \tag{21}$$

where N_d is the total number of droplets. The same quantity must be added to the mixture fraction equation (see Eq. (4)) where the source term is taken to be $\dot{\omega}_Z = \dot{m}_d$. In turn, the momentum and energy source terms can be expressed as

$$S_{i,d} = \sum_{i=1}^{N_d} \left(\frac{d\left(m_d u_{i,d}\right)}{dt} \right) \tag{22}$$

$$Q_d = \sum_{i=1}^{N_d} Q_{c,i} - Q_i. \tag{23}$$

The flow coupling onto the spray is incorporated due to the use of the relative velocity when computing the force on each droplet.

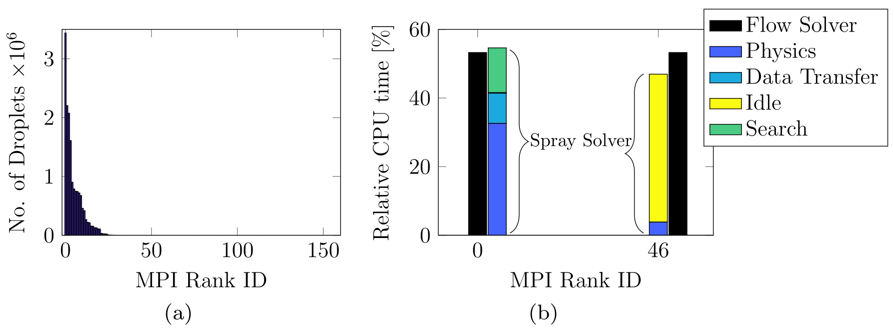

Fig. 4. The distribution of particles over the MPI processes illustrates a major critical performance problem where nearly 90% of the particles are assigned to 10% of the total processes.

2.5. Numerical approach

The governing equations are solved with a Finite Volume Method (FVM) using a second-order central difference scheme for the convective and diffusive terms with a pressure based solver using the SIMPLE [30] scheme adapted to transient problems. The time marching is conducted employing a second order backward Euler scheme. Further details of the flow solver has been presented by Anand et al. [3], Zhu et al. [41].

The gas velocity imposed on the droplets is interpolated by using a nearest neighbour method, which means the cell centre velocity is directly prescribed to every droplet that is located into that respective cell. Similarly, each particle-induced source term is directly prescribed to its associated cell centre. The effect of such approximations will not be assessed in this work. Further details of the spray model and the coupled methodology with the gas phase have been presented and validated by Cha et al. [9].

3. Parallel algorithm

As stated, the aim of this work is to develop a parallel strategy for Eulerian-Lagrangian that overcome the bottleneck associated with load balancing and poor parallelism. For large-scale combustion application, the parallel algorithm is based on domain decomposition with a spatial particle algorithm for the spray part as explained in Section 1. However, this can quickly lead to unacceptable parallel efficiency due to load imbalancing. Fig. 4a illustrates the distribution of particles for a case which is running on 160 cores using 10 nodes. Fig. 4b shows the CPU consumption for two different MPI processes, one (Rank: 0) which is extremely loaded with particles and another (Rank 46) which barely has any. It is observed that Rank 46 spends around 40% of the time waiting while other processes are carrying out the Lagrangian tracking, which is exactly what is observed for the 90% percent of ranks which have a similarly low number of droplets. This shows clearly the current parallel method's weakness in coping with large scale HPC simulation. The core count per node is generally increasing for modern HPC implying that scientific codes will have to employ more parallelism to exploit current and the future HPC architectures.

The parallel approach presented here will help to overcome the current bottleneck and can cope very efficiently with future HPC systems. This work will specifically concentrate on single-node performance, as this is believed to be a key step before multi-node HPC applications can become efficient.

3.1. An asynchronous task-based Eulerian-Lagrangian parallelism

Studying the mathematical and numerical models of both Eulerian and Lagrangian approaches reveals that the dependence between one another can be limited to one step. That is, both methodologies only require a single synchronisation, namely prior to a new time step, to advance both solvers in time. In other words, most of the mathematical operations that take place individually on both solvers can be carried out concurrently. One of the main problems with the current parallel methodology is the sequential nature of the execution which turns into a severe issue in the case of a load imbalance (see Fig. 4). It implies that the vast majority of the computational resources are waiting most of their CPU time so a minority can complete their respective tasks. This degenerates the parallel efficiency and ultimately increases the computational cost considerably. This paper proposes an Asynchronous Task-based Eulerian-Lagrangian (ATEL) framework to ensure every CPU is occupied most of the time. The flow chart of ATEL is illustrated in Fig. 5. This framework does not assume a specific engineering application, however, we will concentrate on gas turbine combustors where the spray plays a vital role in determining the engine's efficiency, emission and duability. Hence, the Lagrangian particle tracking solver is in this work also the spray solver, but in principle any Lagrangian solver can be employed within this suggested framework.

The original framework contains both the Eulerian-Lagrangian methods where each rank is responsible for a subdomain of the flow and spray models. The novel framework presented here implies that a subset of MPI ranks will be tasked to only carry out the Lagrangian tracking, while the remaining processes will be allocated to the Eulerian domain. Fig. 6

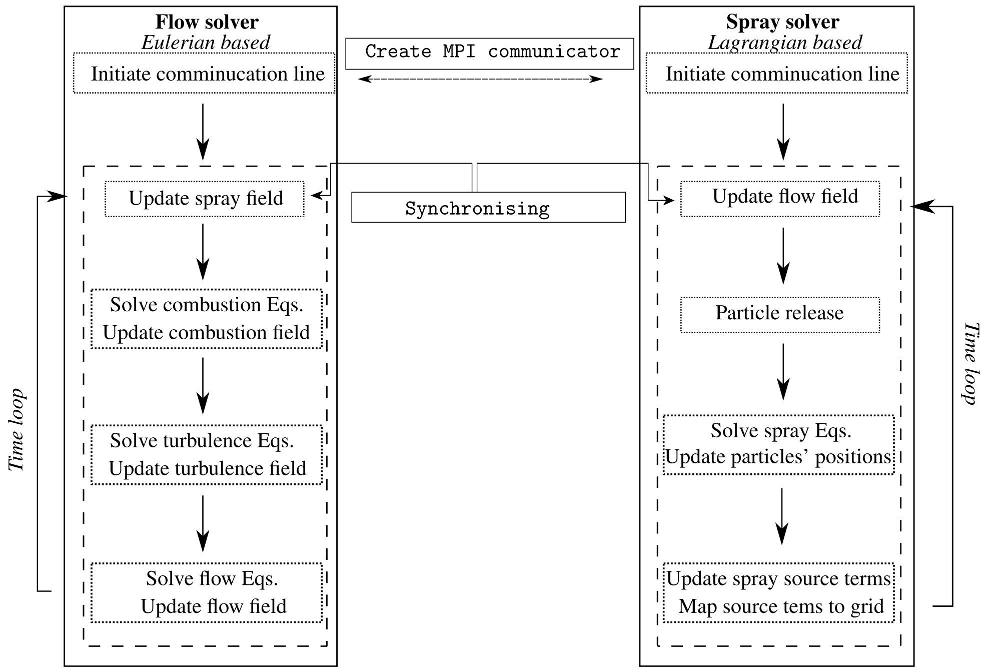

Fig. 5. Flowchart of the asynchronous coupling procedure.

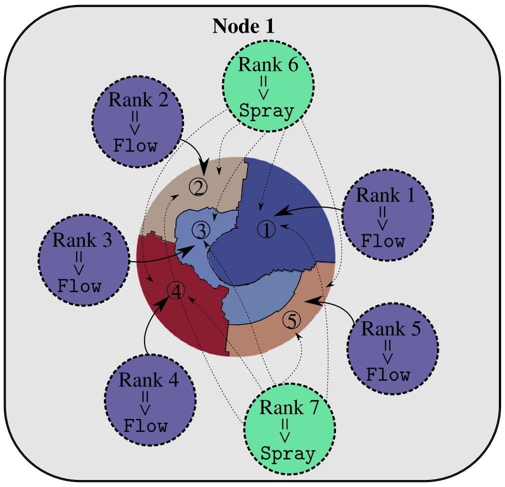

Fig. 6. An example with a single computer node split into between 6 and 2 flow and spray ranks, respectively. Each flow rank is assigned to one partition whereas the spray ranks track the particles across the entire domain.

illustrates an example with 5 MPI ranks allocated to the flow solver, where each has a partition or block of the domain. Additionally, there are 2 MPI ranks for the spray solver and they will both track their respective particles throughout the domain. Therefore, they will have access to all the 5 blocks corresponding to the other MPI ranks allocated to the flow solver. Note that the algorithm can work with arbitrary splits between the flow and spray solvers, and the rank split should be chosen such that both solvers reach the synchronisation step around the same time. It is rather straightforward to ensure an almost perfect load balancing of particle tracking among cases with multiple MPI spray ranks. In this work, the injection location N_{sl} for each fuel nozzle is split evenly among all ranks. This ensures that each spray process will inject the same number of particles per time step.

At the initialisation stage the MPI ranks are either assigned to run the flow or spray solver. During the synchronisation step, the flow field and source terms need to be communicated to the spray and flow ranks, respectively. Subsequently, each spray and flow rank carries out the various tasks concurrently until the next time step. It is worth mentioning that the asynchronous approach has a slight change in the way the system is solved. For the sequential approach the spray solver is always advanced using the flow field information from the beginning of the time-step, while the flow field is

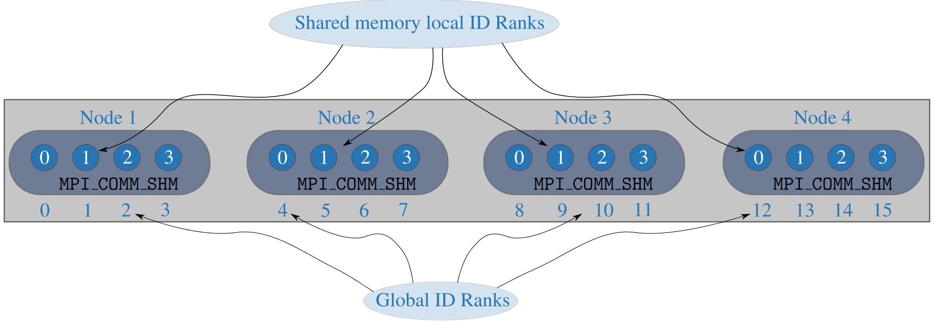

Fig. 7. The MPI shared memory concept where each node has an internal communicator and a corresponding local ID rank.

advanced using the spray information from the end of the time-step. The presented asynchronous approach consistently employs the respective field information from the beginning of the time-step for both solvers. The implications of this will be studied in Section 5, where a sequential version has been performed to assess accurately the impact of such an approach. The sequential case mimics the baseline approach in terms of the sequence in updating the flow and spray field; therefore, providing an opportunity to perform a two-step verification study of ATEL with respect to the baseline solver. Firstly, by using the sequential case study ATEL can be verified based solely on its parallel methodology, and the outcome is expected to be identical to the baseline solver since the underlaying physics is equivalent. Next, a concurrent ATEL study will be carried out to assess any potential impacts such synchronisation methodology causes with respect to the sequential approach.

3.2. *Shared memory & interprocess one-sided communication*

The two main problems with the presented task based approach are memory consumption and communication. The spray solver does not know a priori the particle's trajectory, nor the blocks it will pass through. One approach would be to transfer the mesh associated to all the blocks during initialisation, and similarly the entire flow field at each synchronisation step. In this way, the spray solver can locally keep track of where each particle is located in the Eulerian domain and very efficiently access the flow information needed. However, this method is memory intensive and does not suit the HPC trend of increasing core count per node and less memory per core. It implies that each MPI spray rank would have a copy of the entire grid and flow field. Additionally, transferring the flow variables for each partition to all spray ranks at each time-step can become a bottleneck itself and potentially spoil the speed-up gain from using the asynchronous method.

The increasing core count per node of modern HPC systems motivates the use of shared memory. In conventional MPI applications, the memory of each process is private and can be transferred through message passing. The MPI standard 3.0 [27] introduces the concept of one-sided shared memory communication, where memory within a node can be shared among multiple MPI processes, which can avoid data replication. The higher core count per node will inevitably increase the data exchange within a node, which can be performed more efficiently through direct shared memory access.

Fig. 7 illustrates the general concepts of this feature, where the MPI processes have two IDs; a local ID within a node, and a global ID associated to the global communicator `MPI_COMM_WORLD`. It is the former type of ID in conjunction with a shared memory communicator `MPI_COMM_SHM` that are introduced with this novel concept. Each node has its own `MPI_COMM_SHM` with its local MPI ranks. Each MPI process can allocate shared memory of a certain size by means of the MPI function `MPI_WIN_ALLOCATE_SHARED`. This memory can be accessed by another MPI rank through a pointer which points to the memory address of this window. The pointer can be set up by means of the MPI function `MPI_WIN_SHARED_QUERY`. Following this, the pointer holds the address of the shared window and can access the information it contains. It is the responsibility of the developer to ensure that when remotely accessing such memory, they are not being modified and the data corresponds to the correct time step.

To mimic the suggested concept of the coupled Eulerian-Lagrangian framework, a bandwidth analysis is carried out using the partitioning strategy introduced in Fig. 6. That is, there is one process which is receiving data from a number of processes. This is equivalent to having a number of flow processes sending their local flow field to a single MPI spray rank. The reverse data exchange has also been investigated and shows a similar trend and is not presented here. The size of the data exchanged is varied from around 0.1 to 100 Mb per process. The test case was run on a single node with 40 cores in the form of two Intel® Xeon® Gold 6248 at 2.50 GHz. The C++ code related to this analysis can be found in: https://github.com/akjt/SharedMemoryParallelisation.git.

Fig. 8 shows the communication cost comparison between the conventional Point to Point (P2P) and SHared Memory (SHM) communication techniques using up to $N_r = 40$ ranks, with all tests having one receiving rank and the remaining sending. Each sending MPI process communicates with the single receiving MPI process with varying message sizes and the test is repeated 100 times to obtain an average timing for each sending MPI process. Finally, the speed-up figures have been averaged over the total number of sending MPI ranks. The consumed time includes the time to fill up the buffers and until

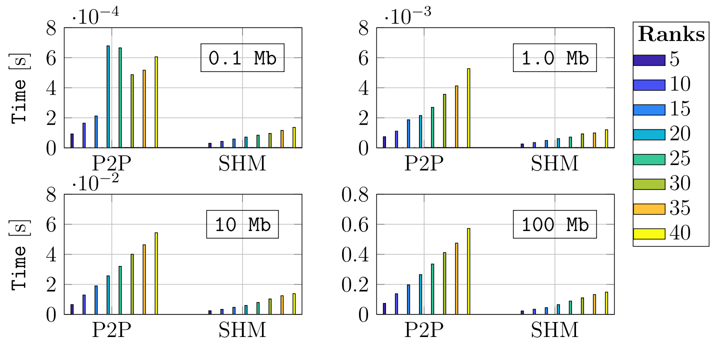

Fig. 8. Communication time comparison between P2P and SHM communication technique for varying message sizes. All analyses are tested with one receiving rank and $N_r - 1$ sending ranks.

it reaches to the receiver at the synchronisation step. In the case of SHM the latter contribution is negligible since there is no actual transferring of data, and the real cost is only the time to fill up the buffer.

It is clear that with more ranks involved the communication cost rises since the single rank needs to receive data from up to 39 ranks. Also, the speed-up using the SHM method seems to be consistently the same for all message sizes peaking at 3x faster for the case with 40 ranks.

3.3. Mapping between flow and spray

By exploiting the shared memory feature, each spray rank can be given access to all blocks of the domain without consuming any additional memory. Each spray rank defines an Array of Structure data type where each element of the array corresponds to a block of the flow domain (see Fig. 6),

$$\texttt{flowvar}\,(i_b = 1 : n_b) \rightarrow \left\{ \begin{array}{c} u_{ptr}\,(1 : n_c) \\ v_{ptr}\,(1 : n_c) \\ w_{ptr}\,(1 : n_c) \\ \vdots \end{array} \right\},$$

where n_b is the total number of blocks. The local cell count associated to a block i_b is denoted n_c. The members contained in each element of the structure correspond to pointers that are pointing to where the various flow variables are stored for the corresponding block i_b. Hence, the allocation and write-access of $\texttt{flowvar}\,(i_b)$ is carried out by the flow rank that is associated to block i_b, and the spray ranks treat them as read-only data. For the case with the conventional point-to-point communication, the members would effectively be locally allocated arrays which are transferred among MPI ranks in a conventional manner. Similarly, each spray rank has an associated data structure of source terms, which corresponds to each flow block.

$$\texttt{sprayvar}\,(i_b = 1 : n_b) \rightarrow \left\{ \begin{array}{c} mass_{ptr}\,(1 : n_c) \\ momx_{ptr}\,(1 : n_c) \\ momy_{ptr}\,(1 : n_c) \\ \vdots \end{array} \right\}$$

Notice that these source terms have already been mapped to the flow domain, as shown in the exemplified pseudo code for the mass source term in Algorithm 1. Unlike the data structure of $\texttt{flowvar}$, each spray rank allocates its own $\texttt{sprayvar}\,(i_b = 1 : n_b)$ and writes directly the source terms with the appropriate indexing, as observed in Algorithm 1 which illustrates the case for the mass source terms of each droplet. Since each spray rank covers all blocks, the source term contribution per time step requires an internal reduction step of each member of $\texttt{sprayvar}\,(i_b)$ across all spray ranks $i_{sp} = 1 \ldots N_{sp}$. Subsequently, only one spray rank needs to transfer the source terms for each block i_b to the associated flow ranks (see Algorithm 2 and 3).

From the flow rank's perspective it only requires one element for the data structures $\texttt{flowvar}(1)$ & $\texttt{sprayvar}(1)$ which correspond to its own block. That is assuming each flow rank has only a single designated block. Although, the choice of this data structure allows for the possibility to have several blocks per flow rank. As mentioned above, the flow writes directly the flow variables into $\texttt{flowvar}$ which is shared with the spray ranks, while it receives the source terms $\texttt{sprayvar}$ from one spray rank following a reduction step of $\texttt{sprayvar}$ among the spray ranks.

To illustrate how this compares with the synchronisation step defined in the flow chart Fig. 5, Algorithms 2 and 3 have been presented. In the case of the point-to-point synchronisation each flow rank would transfer its flow field at each time step to each spray rank. From the spray rank's perspective, it would have to receive the flow field associated to each block.

Algorithm 1:

```
Result: Mass source terms mapped to the flow domain.
foreach particle do
    $i_b$ ← particle.block_id // block id ;
    $i_c$ ← particle.cell_id // local cell id corresponding to the block ;
    spraysource($i_b$).mass($i_c$) ← ADD particle.mass_source ;
end
```

With a shared memory communication technique, the synchronisation becomes very simple since most of the work has been carried out during the initialisation of the shared memory windows.

Algorithm 2: Synchronisation using shared memory method.

```
if Flow Rank then
    // Wait for all spray ranks to complete the time-step.
    $i_b$ ← myblock_id ;
    BARRIER with all spray ranks $\in i_{sp} = 1 \ldots N_{sp}$.;
    COPY sprayvar($i_b$) to a local buffer;
end
if Spray Rank then
    foreach block do
        // Sum sources to Spray root rank's shared buffer.
        sprayvar$(i_b)_{i_{sp}=1}$ ← REDUCE sprayvar$(i_b)_{i_{sp}=1 \ldots N_{sp}}$
    end
    // Wait for all flow ranks to complete the time-step.
    BARRIER with all flow ranks $\in i_f = 1 \ldots N_f$.;
    COPY flowvar($i_b = 1 : n_b$) to a local buffer;
end
Advance to next time-step.
```

To ensure safe and consistent parallel execution there are two vital points to consider with shared memory communication. Firstly, before starting a new time-step, both the flow and spray must ensure their respective partners have completed their time-step and we can read the new information, which is also referred as the synchronisation step. For the point-to-point communication method this is implicitly ensured, however, a shared memory communication approach requires an explicit action. There are several ways to ensure synchronisation (see [27]). In this work, we complete this with a `MPI_BARRIER` as also indicated in Algorithm 2.

Algorithm 3: Synchronisation using point-to-point method.

```
if Flow Rank then
    // Flow rank broadcasts its local flowfield to all spray ranks.
    BROADCAST flowvar to all spray ranks.;
    // Receive sources associated to the local block.
    $i_b$ ← my_block_id ;
    RECEIVE sprayvar$(i_b)_{i_{sp}=1}$ from root spray rank.
end
if Spray Rank then
    // Each spray rank receives all local flowfields.
    foreach block do
        $i_b$ ← block_id ;
        $i_d$ ← rank_id($i_b$) ;
        RECEIVE ← flowvar($i_b$) from rank $i_d$ ;
        // Sum sources and send from root to the associated block.
        sprayvar$(i_b)_{i_{sp}=1}$ ← REDUCE sprayvar$(i_b)_{i_{sp}=1 \ldots N_{sp}}$ ;
        if Spray Root then
            SEND ← sprayvar$(i_b)_{i_{sp}=1}$ to flow rank $i_d$
        end
    end
end
Advance to next time-step.
```

Secondly, only the rank that owns the shared buffer window has write access while its associated partner(s) has read-only access. In fact, the read-only permission is restricted to be used once per time-step namely during the synchronisation among the owner rank and its partner(s). This single read-only permission is used to make a safe copy of the received data into a local buffer that can be used throughout the calculations of next iteration. The reason for this is, for example, if the spray rank is accessing the flow field for a particular cell to compute drag, the flow rank responsible for this cell could

Table 1
The parameters of the three study-cases.

Injection conditions	Case 1	Case 2	Case 3
P_{gas} [MPa]	1.1	3.0	5.0
$\dot{m}_d$ [g/s]	6.06	5.36	5.13
D_d [μm]	300	300	300
u_d [m/s]	102	90.3	86.4
t [ms]	4.0	8.0	8.0

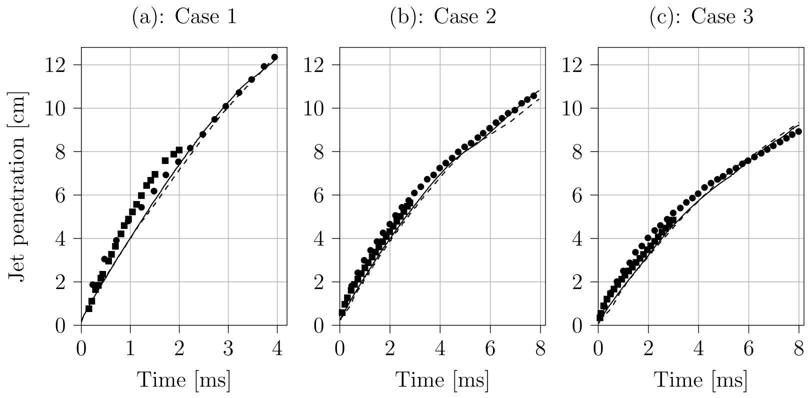

Fig. 9. A comparison of the jet penetration for all three cases. It is evident that the base line code and the ATEL are nearly identical. The stochastic breakup method will inevitably cause deviations due to different random number sequences when using different number of ranks. ● [18] ■ [4] (———) Base line (– – –) ATEL.

Table 2
The scaling of the base line code using Case 1. The parallel efficiency is calculated with respect to the case with 5 ranks. The ideal wall clock times are given in brackets.

Ranks	CPUh	Wall clock [h]	Flow CPUh	Spray CPUh	Efficiency [%]
40	48.00	1.20 (0.43)	27.10	19.80	35.63
35	48.40	1.38 (0.49)	24.05	23.39	35.33
30	39.50	1.32 (0.57)	21.30	17.38	43.29
25	34.30	1.37 (0.68)	19.60	14.03	49.85
20	34.00	1.70 (0.86)	17.19	16.27	50.29
15	22.10	1.47 (1.14)	16.10	5.60	77.38
10	28.40	2.84 (1.71)	15.50	12.67	60.21
5	17.10	3.42 (3.42)	13.18	3.81	100.00

Table 3
Performance data for the new developed parallel code ATEL using Case 1. The total MPI ranks is maintained at 40, but the split has been varied. The ideal wall clock is indicated in brackets and the baseline performance is highlighted with italic for comparison.

N_{fl}	N_{sp}	CPUh	Flow CPUh	Spray CPUh	Wall clock [h]	Efficiency [%]	Reduction CPUh	Efficiency change [pp]
40	*0*	*48.0*	*27.1*	*19.8*	*1.20 (0.43)*	*35.63*	*0.00*	*0.00*
39	1	78.1	76.1	1.95	1.95 (0.43)	21.9	-30.1	-13.7
38	2	52.1	49.5	2.61	1.30 (0.43)	32.8	-4.10	-2.80
37	3	39	36.1	2.93	0.98 (0.43)	43.8	9.00	8.22
36	4	37.3	33.6	3.73	0.93 (0.43)	45.8	10.7	10.2
35	5	34	29.8	4.25	0.85 (0.43)	50.3	14.0	14.7
34	6	33.7	28.7	5.06	0.84 (0.43)	50.7	14.30	15.1
33	7	34	28.1	5.95	0.85 (0.43)	50.3	14.0	14.7

potentially be updating that flow variable for the next time-step, which can lead to data-race issues. That is, since the flow or spray rank uses its sharing buffers directly during the iterations, it is never safe to allow its partners to read such buffer during its calculations. In practical terms it means that each flow rank i_f needs a single copy of `sprayvar`(i_b) where i_b is the block associated to i_f, and each spray rank requires a copy from all flow ranks ($i_f = 1 \ldots N_f$) the flow field associated to each block `flowvar`$(i_b = 1 : n_b)$.

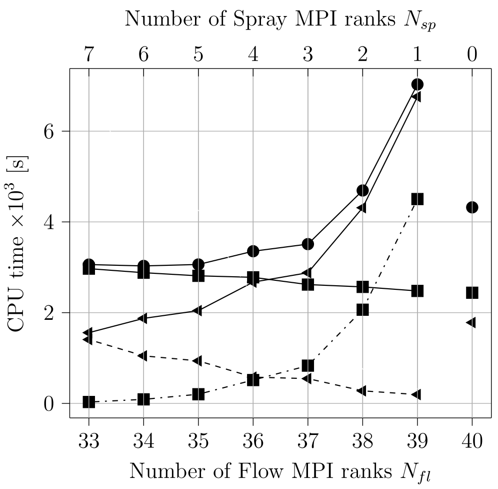

Fig. 10. The performance of the ATEL showing both solvers CPU time and their respective synchronisation time. The abscissa only indicates the number of MPI ranks for both the solvers. The base line performance data is provided for reference. (●) Wall clock base line (■) Wall clock Flow base line (◀) Wall clock Spray base line (–●–) Wall clock ATEL (–■–) Wall clock Flow ATEL (–◀–) Wall clock Spray ATEL (·■·) Waiting time for flow ATEL (-◀-) Waiting time for spray ATEL.

4. Atomising spray in a diesel engine configuration

The first case-study to validate the parallel method is based on a diesel engine configuration which has been tested experimentally by Hiroyasu and Kadota [18]. Additionally, a CFD study has been conducted by Apte et al. [4]. The closed cylindrical chamber has a length of 138 mm and a diameter of 56 mm. The diesel fuel of $\rho = 840$ kg/m^3 is injected by a single hole nozzle which is equivalent to the size of the injected droplet diameter, namely $D_d = 300$ µm. Hence, this case has only one injection location with a nozzle ring diameter of 0 m. The liquid fuel is injected into a constant pressure nitrogen chamber at $T = 294.15$ K. The flowfield is a rest and is only disturbed by the momentum exchange once the droplets are injected into the domain. This case has been tested under three different conditions summarised in Table 1. The injection velocity has been obtained using the fixed size of the injected droplets together with the mass flow rate of the liquid $\dot{m}_d$ and assuming spherical shaped droplets,

$$\dot{m}_d = \pi \frac{D_d^2}{4} \rho_d u_d \tag{24}$$

The computational domain has been discretized with $200 \times 96 \times 96$ finite volume cells and a time-step of $\Delta t = 15.0$ µs has been employed.

Fig. 9 shows the jet penetration for all three cases including the experimental results obtained by Hiroyasu and Kadota [18] and the numerical findings available by Apte et al. [4]. The comparison between the base line code and ATEL is almost identical. There two elements which inevitably will result in small deviations. Firstly, in the base-line code the spray part is advanced in time using the flow information from the previous time-step, while the flow step was based on spray information obtained from the same time-step. In ATEL, both the spray and the flow advance in time using the results from the start of the time-step. Although this source of deviation is small, it is observable when a one-to-one comparison is made. Secondly, the stochastic breakup model employed in this case-study relies on a random number generator. While each MPI rank has its own private stream, altering the number of ranks will ultimately cause different random number sequences. In comparison to the experimental results, it is evident that both the base line code and ATEL fail to capture the penetration depth during the start but do so at a later stage. Collision and coalescence are not incorporated in the spray model, which is most likely the reason for this observation, as the droplets at the early stage are clustered together [4].

All performance test cases have been conducted on a node with 2 processors of Intel® Xeon® Gold 6148 CPU @ 2.40 GHz with 20 cores each. All cases have been consistently tested up to 40 MPI ranks. Table 2 summarises the performance of the baseline code. The figures show that the spray solver performance scales with 1.5x against 3.9x for the flow solver when comparing the runs for $N_{fl} = 5$ and $N_{fl} = 40$. The performance of the spray solver also fluctuates with respect to the number of flow ranks $N_{fl} = 40$ (or number of partitions). The load balancing of the droplets is dependent on the distribution of droplets in the domain in relation to the domain decomposition, which means fewer partitions would lead to a more favourable load distribution of the droplets. Ultimately, the CPU cost of the spray solver is strongly case dependent, and therefore implies no apparent relation between increasing processing units and speed-up. As a consequence, the overall parallel efficiency degenerates considerably. This is also due to the sequential nature where the significantly less loaded processes are awaiting the remaining to complete the spray step. Notice that the presented total CPUh also includes any

Table 4
A summary of the air and fuel inlet conditions.

Inlet air pressure	Inlet air temperature	Air-fuel ratio	Fuel mass flow rate	Inlet fuel temperature	SMD
4 bar	550 K	20	2.98 g/s	350 K	6.5 µm

Table 5
A summary of the fuel nozzle details.

	Nozzle 1	Nozzle 2	Nozzle 3
Diameter	0.015 m	0.014 m	0.013 m
(V_x, V_θ, V_r)	(30,0,0) m/s	(30,0,-15) m/s	(30,0,-30) m/s
Fuel rate	0.993 g/s	0.993 g/s	0.993 g/s

Table 6
The L2 norm error with respect to the baseline framework. It is evident that the sequential approach is nearly identical to the baseline, while the concurrent one does lead to a slightly larger discrepancy.

	Sequential		Concurrent	
	$z = 7$ mm	$z = 30$ mm	$z = 7$ mm	$z = 30$ mm
$\|\|\overline{u}\|\|_2$	0.016%	0.012%	0.899%	0.671%
$\|\|u_{RMS}\|\|_2$	0.034%	0.058%	1.504%	2.97%
$\|\|\overline{T}\|\|_2$	0.009%	0.001%	0.221%	0.379%
$\|\|T_{RMS}\|\|_2$	0.018%	0.047%	0.054%	0.549%

overhead and housekeeping incurred during the simulation such as pre/post-processing and syncronisation; hence, it is slightly larger than the combined flow and spray CPUh.

Table 3 provides the performance data of the proposed ATEL methodology using a total of 40 MPI ranks where split among flow N_{fl} and spray N_{sp} ranks has been varied. It is clear that using fewer spray resources than required might have an adverse effect on performance, however, its potential is huge. Comparing to the baseline code, it is possible to achieve a reduction of 14 CPUh resulting in an increase of 15 percentage point efficiency (pp) and 30% reduction in wall clock time. The table clearly shows significant reductions in CPUh for the spray solver compared to the baseline case, while the CPUh for the flow solver remains unaffected. However, there is one outcome that requires more attention: the spike in CPUh for the flow solver is exclusively due to its significant idling time for the case with $N_{fl} = 39$ and $N_{sp} = 1$ (see also Fig. 10). This case does require at least two spray ranks to avoid having such a load imbalance between the solvers. Fig. 10 shows the actual compute time for both the flow and spray solvers in addition to their associated waiting time for a total number of 40 MPI ranks. The waiting time is when the flow or spray solver has been waiting for the other solver's completion of a time-step. From the flow solver's perspective, the idling time drops significantly when the number of spray ranks is increased. On the other hand, it increases for the spray modestly with spray ranks (or decreasing flow ranks). A pivotal outcome that can be observed from the ATEL performance figures is that while both solvers might have an approximately similar wall clock time independently, the total clock time does not deviate much from this, in contrast to the base line case which is the accumulation of both solvers' CPU time. This shows a potential benefit of asynchronous algorithms for future multiphysics simulations.

As a final note in regards to this test case and what has been learned, it is important to mention that this case does impose challenges to the suggested framework despite its apparent simplicity. All runs start from no droplets present in the domain. This means that all sprays ranks are doing nothing or very little during the first part of the simulation, which ultimately increases the final idling time for the spray solver. However, as the simulation progresses the flow field begins to develop, and droplet count rises quickly up to a maximum of around 14M due to the injection and breakup process. In turn, the flow ranks might experience considerable idling time depending on the number of spray ranks employed. That is the reason why an appreciable waiting time can be observed for both solver for this case-study (see Fig. 10). This implies that the load balancing is varying strongly over the course of the simulation. Nonetheless, the ATEL framework showed its robustness in dealing with such less frequent scenarios. For applied large scale LES simulations of combustion flow, these scenarios are less likely to occur since a common practice is to begin from a developed flow field with a distributed spray field, where load balancing is not as dynamic as in fact observed in this test case.

5. Reacting spray in a gas turbine combustor

This case-study examines a reacting LES simulation of a gas turbine combustor. The burner's geometry is based on the DLR Generic Single Sector Combustor (GENRIG), which has been described in detail by Freitag et al. [15] and Meier et al. [26]. This has further been examined by a reacting LES simulation carried out by Jones et al. [21]. The numerical model was validated with experimental results using a cold-case (non-reactive), and subsequently carried out an analysis of a reactive LES simulation (see [21]). In this study, a reactive LES simulation with evaporating droplets will be considered

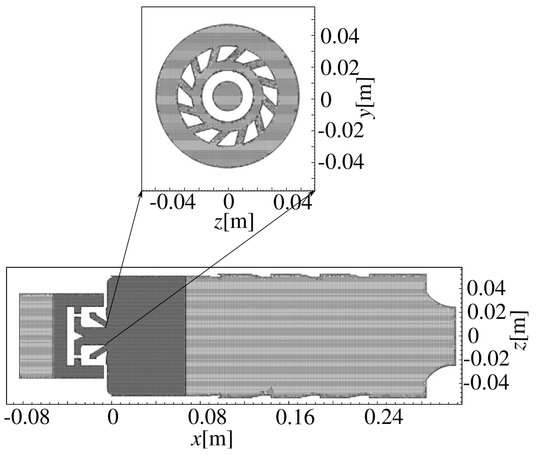

Fig. 11. A sketch of discretized domains of the combustor and burner.

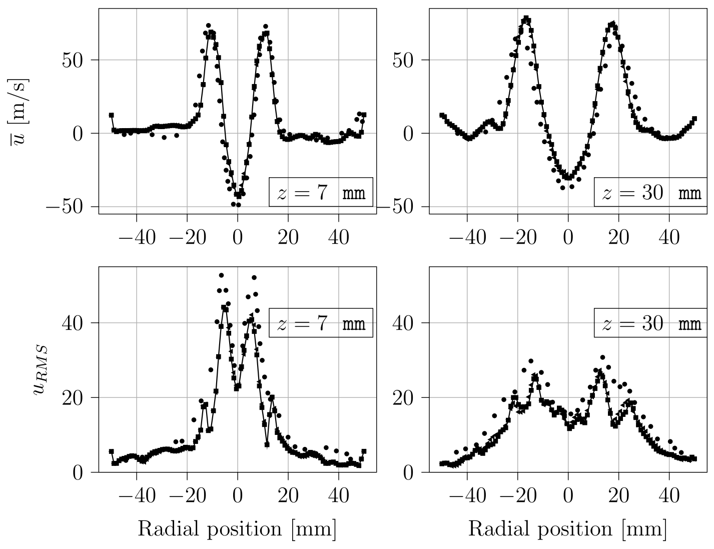

Fig. 12. The radial mean and RMS velocity profile at $z = 7$ mm (left column) and $z = 30$ mm (right column) are compared against the baseline code and [21]. A synchronised version of ATEL has also been shown to assess the impact of the asynchronous approach. —— Sequential ATEL ◂ Concurrent ATEL ▪ baseline • [21].

Table 7
The scaling of the baseline code using the DLR GENRIG test case. The parallel efficiency is calculated with respect to the case with 5 ranks. The ideal wall clock times are given in brackets.

Ranks	CPUh	Wall clock [h]	Flow CPUh	Spray CPUh	Efficiency [%]
40	1919	47.96 (16.43)	1071	832.5	34.25
35	1481	42.30 (18.77)	1006	461.1	44.38
30	1241	41.35 (21.90)	841.0	388.6	52.96
25	1070	42.79 (26.28)	766.3	293.8	61.41
20	925.1	46.25 (32.85)	682.9	233.8	71.03
15	852.6	56.84 (43.80)	680.0	164.8	77.06
10	771.4	77.14 (65.70)	600.0	165.1	85.17
5	657.0	131.4 (131.4)	575.1	76.44	100.00

only. Such cases that commonly suffer from highly non-uniform distribution of particles, the great majority of droplets are clustered near the injector making such a simulation difficult to scale for large particle count as explained in Sections 1 and 3. Firstly, the base-line and ATEL will be compared with results obtained by Jones et al. [21]. Then, a parallel performance analyses will be carried out to investigate the novel parallel methodology presented in this work by comparing the speed-up of the base-line code and ATEL. All performance test cases have been conducted on a node with 2 processors of Intel® Xeon® Gold 6148 CPU @ 2.40 GHz with 20 cores each.

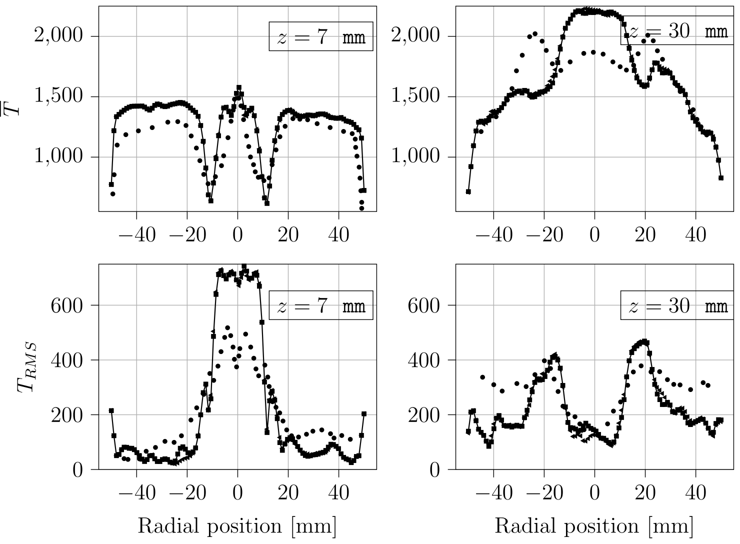

Fig. 13. The radial mean and RMS temperature profile at $z = 7$ mm (left column) and $z = 30$ mm (right column) are compared against the baseline code and [21]. A synchronised version of ATEL has also been shown to assess the impact of the asynchronous approach. —— Sequential ATEL ⬩ Concurrent ATEL ▪ baseline • [21].

Table 8
Performance data for the new developed parallel code ATEL using the DLR GENRIG test case. The total number of MPI ranks is tested at 40 and 20, where the split among the two solvers has been varied. The ideal wall clock for the corresponding total number ranks is indicated in brackets and the baseline performance is highlighted with italic for comparison.

N_{fl}	N_{sp}	CPUh	Flow CPUh	Spray CPUh	Wall clock [h]	Efficiency [%]	Reduction CPUh	Efficiency change [pp]
40	*0*	*1919*	*1071*	*832.5*	*47.96 (16.43)*	*34.25*	*0.00*	*0.00*
39	1	1435	1386	35.60	35.88 (16.43)	45.78	482.9	11.52
38	2	1183	1081	41.60	29.58 (16.43)	55.52	735.2	21.28
37	3	1182	1069	41.08	29.56 (16.43)	55.57	736.2	21.32
20	*0*	*925.1*	*682.9*	*233.8*	*46.25 (32.85)*	*71.03*	*0.00*	*0.00*
19	1	772.2	694.2	31.56	38.61 (32.85)	85.10	152.9	14.06
18	2	770.4	681.7	76.95	38.52 (32.85)	85.29	154.7	14.26

The computational domain consists of a plenum where the flow develops into a turbulent and representative inlet condition for the two radial swirlers. The domain consists of 7 million computational cells with the smallest cell size $\Delta x \approx \Delta y \approx \Delta z \approx 0.5$ mm (see Fig. 11). The inlet air pressure is 4 bar and preheated to an inlet temperature of 550 K. The spray is injected at $T_D = 350$ K with a Sauter Mean Diameter SMD $= 6.5$ μm using 10 bin sizes for the Rosin-Rammler distribution. Three fuel nozzle rings each with 60 injection locations have been applied with a centre point at $(x, y, z) = (0, 0, 0)$. The rings' diameters are respectively set to 0.015 m, 0.014 m and 0.013 m. This ensures a fair distribution of the inlet fuel injection. In this case, the droplets are assumed atomised and are mostly affected by the evaporation rate. Hence, no breakup model has been employed in this study. The total mass fuel rate is set to $\dot{m}_f = 2.98\,\text{g/s}$ and is split evenly among the three nozzles. Further details of the setup have been summarised in Tables 4 and 5. The time-step $\Delta t = 10^{-6}$ is applied for all simulations. Prior to conducting any LES sampling, a steady-state RANS simulation was conducted to provide an initial condition for the LES calculation. Then, the simulation was run for 20,000 time-steps which is around 4 flow-through times before initiating the statistical sampling. For validation purposes, the simulation was run for 10,000 time-steps accounting for around 2 flow-through times, while for the parallel performance study only 1000 time-steps were necessary to obtain representative figures for the computational cost. For the validation and performance studies, the total number of droplets stays approximately constant around 25 million active with some evaporating and others being injected at each time-step.

Figs. 12 and 13 compare the mean and Root Mean Square (RMS) of the velocity and temperature fields along the radial direction at two different planes $z = 7$ mm (left column) and $z = 30$ mm (right column). As expected, the results obtained by the ATEL framework compares excellently with the baseline code since both are based on the same physics and modelling techniques. In comparison to [21] there are measurable deviations particularly for the temperature profiles, which is a sensitive to the combustion model employed [38]. The combustion model employed by Jones et al. [21] is based on a stochastic PDF approach while the tabulated FGM approach is used in this work (see Section 2.1.1). Additionally, we have sampled over 10,000 time-steps using $\Delta t = 10^{-6}$ against the 300,000 time-steps with $\Delta t = 10^{-7}$ used by Jones et al. [21]. However, the objective and essence of this work is to ensure that the novel parallel methodology ATEL does not deviate from the baseline framework, which itself has been extensively tested, validated and used by industry for the past many decades [9,41,3]. Therefore, the scope here is to firstly verify that the task based parallel approach produces exact results with respect to the baseline framework, and assess the implications of the asynchronous approach where the flow and spray solvers are solved consecutively, which as a result lead to slightly different synchronisation procedure (see Section 3.1). By sequential ATEL we refer to the same sequence of updating the flow and spray field; therefore, this is expected to produce the same solution compared to the baseline framework. From Figs. 12 and 13 the difference obtained from the synchronous ATEL and the baseline solver is hardly visible, while minor differences are observed for the asynchronous ATEL.

Table 6 shows the L2 norm error with respect the baseline solver which is calculated for any field variable $\boldsymbol{\psi}$ using following relations

$$\epsilon = \frac{\left|\boldsymbol{\psi}_{\text{baseline}}\right|^2 - \left|\boldsymbol{\psi}_{\text{ATEL}}\right|^2}{\left|\boldsymbol{\psi}_{\text{baseline}}\right|^2} \times 100 \tag{25}$$

$$|\boldsymbol{\psi}|^2 = \sqrt{\sum_{k=1}^{n=1} |\psi_k|^2}. \tag{26}$$

It is evident from Table 6 that the general comparison between ATEL and the baseline is adequate, and nearly identical for the sequential version, which mimics the synchronisation procedure in the baseline solver. Expectedly, it does indicate an increased, but acceptable, discrepancy with respect to the concurrent ATEL due to the more explicit nature of synchronising between the flow and spray field (see Section 3.1).

Table 7 shows the parallel scaling figures using the baseline code, while Table 8 shows the performance data for the ATEL framework. The figures associated to CPUh cover the entire simulation including pre- and post-processing, while the flow and spray CPUh are the costs of their respective time-marching, which in turn is 95% of the total computational cost. Comparing the baseline performance to those presented in Table 8, the ATEL framework improves the parallel efficiency by 21 percentage point and a total cost reduction of 735 CPUh. For a smaller number of ranks the baseline code obtains higher parallel efficiencies that can be attributed to the relative even load distribution of droplets among the decreasing number of blocks, and therefore more particles per block. Additionally, the flow-to-spray cost ratio is larger due to the increased number of cells per block. Comparing the case with 20 ranks ($N_{fl} = 19$ and $N_{sp} = 1$) shows an even higher parallel efficiency of 85% obtained by the ATEL framework against 71% by the baseline code with $N_{fl} = 20$ (see Table 8) indicating its robustness to increase the parallel performance even in cases where the parallel performance is already acceptable. Table 8 also compares the flow and spray CPUh showing a drop of nearly 800 CPUh with respect to the spray solver for the baseline case against the ATEL case on 40 ranks. Also, it is observed that the spray cost doubles going from 19-1 to 18-2 rank split. That is primarily due to the synchronisation step: The flow-to-spray cost ratio is larger with only 19 or 18 ranks, and the spray ranks consume most of their time waiting for the flow ranks to complete their time-step. It is therefore sufficient to have one spray rank when the total number of ranks reduces to 20. However, for the 40-1 rank split the flow CPUh is higher compared to the baseline case, which is attributed to the additional synchronisation time where the flow ranks wait for the spray rank to complete its time-step. By adding an additional spray rank (38-2 rank split) we observe the flow CPUh to drop back to the baseline figure. The choice of how to split among the flow and spray solvers will become relevant for further studies when larger scale multi-node simulations are conducted.

6. Concluding remarks

This paper has presented a novel parallel approach for a two-way coupled Eulerian-Lagrangian methods based on an asynchronous task-based (ATEL) methodology that overcomes the load balancing problem encountered in conventional parallel Eulerian-Lagrangian frameworks. The novel approach has been verified to near machine precision accuracy with respect to the baseline solver, and validated with experimental and numerical findings from other sources. The presented work has focused on ATEL's single node performance and the impact on using an asynchronous based coupling. The parallel performance was documented for both the baseline and ATEL frameworks, and has been investigated using a non-reacting 2D cylindrical nitrogen chamber with a complex breakup mechanism of droplets, and a complex three-dimensional combustion chamber with reacting flow and evaporating fuel droplets.

The novel approach demonstrated a significant improve in the overall parallel efficiency for both test cases. ATEL provided huge cost savings by reducing the total CPUh to 62% of the original value, and improves the parallel efficiency of the coupled Eulerian-Lagrangian as a whole by 21 percentage points compared to the baseline methodology. Additionally, ATEL has been shown to even further improve test cases which were not severely impacted by parallel performance by increasing the parallel efficiency from 71% to 85% indicating its robustness in coping with a wide range of scenarios.

The results show that the load balancing issue can be completely eliminated, and furthermore, the additional cost of solving the Lagrangian field in addition to the Eulerian field, can be almost entirely hidden by means of the task based asynchronous nature. The Eulerian solver is hardly affected by removing a single or two processing ranks out of a total of 40 – this is likely because the flow solver is memory bandwidth limited. However, these two ranks when re-allocated to solve the Lagrangian field are sufficient to cover its computational requirements. That is, the cost of adding an extra physical solver to solve a multi-physical problem can be obtained with almost no computational overhead. However, the results are for single compute node cases only, and in future work we will investigate as to whether this performance benefit to multi-node computations on large problems.

Declaration of competing interest

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

Acknowledgements

This work is financially supported by Rolls-Royce and the EPSRC (Engineering and Physical Sciences Research Council) through the Centre for Doctoral Training in Gas Turbine Aerodynamics (grant ref. EP/L015943/1). The authors are also grateful to Dr. Christopher Goddard from Rolls-Royce for providing the GENRIG test case and Dr. Ruud Eggels from Rolls-Royce Deutschland for valuable discussions.

Appendix A. Mathematical model

The viscous stress tensor and energy flux introduced in Eq. (2) and Eq. (3) read the following

$$\tau_{ij} = \mu \left(\frac{\partial \widetilde{u}_i}{\partial x_j} + \frac{\partial \widetilde{u}_j}{\partial x_i} \right) - \frac{2}{3} \mu \frac{\partial \widetilde{u}_k}{\partial x_k} \delta_{ij}, \tag{A.1}$$

$$q_i = \overline{\lambda} \frac{\partial \widetilde{T}}{\partial x_i} + \overline{\rho} \sum_{k=1}^{N} \widetilde{h}_k \overline{Y}_k \overline{V}_{k,i} \tag{A.2}$$

where μ is the dynamic viscosity and δ_{ij} is the Kronecker delta. The energy flux q consists of a heat flux due to temperature difference following the Fourier's law of diffusion, where λ is the conductivity of heat, and the diffusion of species k with enthalpies of h_k, mass fraction Y_k and diffusion velocities $V_{k,i}$ of the kth species in the ith direction. This diffusion term is provided by the FGM look-up table and where the diffusion velocity was found using the Fick's Law [28]. In this work, the energy Eq. (3) solves for the absolute specific enthalpy h, which is derived from the following relation,

$$h = h_s + \sum_{k=1}^{N} \Delta h^o_{f,k}, \tag{A.3}$$

where $\Delta h^o_{f,k}$ is the formation enthalpy for species k, and h_s is the sensible enthalpy which is given by following relation

$$h_{s,k} = \int_{T_0}^{T} C_{p,k} dT, \tag{A.4}$$

where $C_{p,k}$ is the heat capacity at constant pressure for the kth species and T is temperature.

A.1. LES model

In a LES formulation it is sought to resolve largest turbulence eddies while the unresolved scales are modelled with a sub-grid model. The filtered Navier-Stokes equations (see Eqs. (1) to (3)) are not closed due to unresolved sub-grid stresses $\left(\widetilde{u_i u_j} - \widetilde{u}_i \widetilde{u}_j\right)$ and enthalpy fluxes $\left(\widetilde{u_i h_s} - \widetilde{u}_i \widetilde{h}_s\right)$. In this work, a standard Smagorinsky model is employed for the sub-grid scales following the procedure presented in [3] and [32]. Using the Boussinesq approximation it can be shown that the unresolved sub-grid stresses can be approximated as

$$\widetilde{u_i u_j} - \widetilde{u}_i \widetilde{u}_j \approx -2\nu_t \left(\widetilde{S}_{ij} - \frac{1}{3} \widetilde{S}_{kk} \delta_{ij} \right) + \frac{2}{3} \overline{\rho} k \delta_{ij} \tag{A.5}$$

$$\widetilde{S}_{ij} = \frac{1}{2} \left(\frac{\partial \widetilde{u}_i}{\partial \widetilde{x}_j} + \frac{\partial \widetilde{u}_j}{\partial \widetilde{x}_i} \right) \tag{A.6}$$

where the subgrid scale viscosity ν_t is expressed as

$$\nu_t = C_s^2 \Delta^{4/3} l_t^{2/3} \left(2 \widetilde{S}_{ij} \widetilde{S}_{ij} \right)^{1/2}, \tag{A.7}$$

where Δ is the grid width, l_t is the turbulence length scale and $C_s = 1.2$ is the Smagorinsky model constant. The unresolved enthalpy diffusion fluxes are modelled by the gradient approach [32].

$$\widetilde{u_i h} - \widetilde{u}_i \widetilde{h} \approx -\frac{\nu_t}{\mathrm{Pr}_t} \frac{\partial \widetilde{h}}{\partial x_i} \tag{A.8}$$

where Pr_t is the turbulent Prandtl number equal to 0.9. More information on subgrid scale using the Smagorinsky model for reacting flows is referred to [32,11,3,41].

A.2. Combustion model

The FGM combustion model employed in this work adds additional four transport equations, namely transport of mixture fraction, progress variable and their respective variances. These read:

$$\frac{\partial \overline{\rho}\widetilde{Z}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho}\widetilde{u}_i\widetilde{Z}\right) = \frac{\partial}{\partial x_i}\left[\nu_t \frac{\partial \widetilde{Z}}{\partial x_i}\right] + \dot{\omega}_Z \tag{A.9}$$

$$\frac{\partial \overline{\rho}\widetilde{\phi}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho}\widetilde{u}_i\widetilde{\phi}\right) = \frac{\partial}{\partial x_i}\left[\nu_t \frac{\partial \widetilde{\phi}}{\partial x_i}\right] + \widetilde{\dot{\omega}_c}, \tag{A.10}$$

$$\frac{\partial \overline{\rho}\widetilde{Z'^2}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho}\widetilde{u}_i\widetilde{Z'^2}\right) = \frac{\partial}{\partial x_i}\left[\nu_t \frac{\partial \widetilde{Z'^2}}{\partial x_i}\right] + C_1\overline{\rho}\nu_t\left(\frac{\partial \widetilde{Z}}{\partial x_i}\right)^2 - C_2\overline{\rho}\nu_t\widetilde{Z'^2}\Delta^{-2} \tag{A.11}$$

$$\frac{\partial \overline{\rho}\widetilde{\phi'^2}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\rho}\widetilde{u}_i\widetilde{\phi'^2}\right) = \frac{\partial}{\partial x_i}\left[\nu_t \frac{\partial \widetilde{\phi'^2}}{\partial x_i}\right] + C_3\overline{\rho}\nu_t\left(\frac{\partial \widetilde{\phi}}{\partial x_i}\right)^2 + C_4\widetilde{\phi'^2\dot{\omega}_c} - C_5\overline{\rho}\nu_t\widetilde{\phi'^2}\Delta^{-2}, \tag{A.12}$$

where model constant $C_1 = 1.215$ and the remaining $C_3 = C_3 = C_4 = C_5 = 1.0$. The progress variable and progress variable variance source terms $\widetilde{\dot{\omega}}_\phi$ and $\widetilde{\phi'^2\dot{\omega}_\phi}$ are provided by the FGM look-up table and depends on the definition of ϕ. The following definition is used in this work

$$\phi = \frac{Y_{CO_2}}{M_{CO_2}} + \frac{Y_{H_2O}}{M_{H_2O}} + \frac{Y_{H_2}}{M_{H_2}}, \tag{A.13}$$

where M is the molecular weight of each species. More information on how to construct the FGM look-up table and determine the unclosure terms in the variance transport equations is described in great detail by van Oijen et al. [28] and Poinsot and Veynante [32], respectively.

References

[1] H.J. Aguerre, N.M. Nigro, Implementation and validation of a Lagrangian spray model using experimental data of the ECN Spray G injector, Comput. Fluids 190 (2019) 30–48, https://doi.org/10.1016/j.compfluid.2019.06.004.
[2] A. Amritkar, S. Deb, D. Tafti, Efficient parallel CFD-DEM simulations using OpenMP, J. Comput. Phys. 256 (2014) 501–519, https://doi.org/10.1016/j.jcp.2013.09.007.
[3] M.S. Anand, R. Eggels, M. Staufer, M. Zedda, J. Zhu, An advanced unstructured-grid finite-volume design system for gas turbine combustion analysis, 2013, pp. 1–12.
[4] S.V. Apte, M. Gorokhovski, P. Moin, LES of atomizing spray with stochastic modeling of secondary breakup, Int. J. Multiph. Flow 29 (2003) 1503–1522, https://doi.org/10.1016/S0301-9322(03)00111-3.
[5] S.V. Apte, K. Mahesh, M. Gorokhovski, P. Moin, Stochastic modeling of atomizing spray in a complex swirl injector using large eddy simulation, Proc. Combust. Inst. 32 II (2009) 2257–2266, https://doi.org/10.1016/j.proci.2008.06.156.
[6] S.V. Apte, K. Mahesh, P. Moin, J.C. Oefelein, Large-eddy simulation of swirling particle-laden flows in a coaxial-jet combustor, Int. J. Multiph. Flow 29 (2003) 1311–1331, https://doi.org/10.1016/S0301-9322(03)00104-6.
[7] D. Buaria, P.K. Yeung, A highly scalable particle tracking algorithm using partitioned global address space (PGAS) programming for extreme-scale turbulence simulations, Comput. Phys. Commun. 221 (2017) 246–258, https://doi.org/10.1016/j.cpc.2017.08.022.
[8] J. Capecelatro, O. Desjardins, An Euler-Lagrange strategy for simulating particle-laden flows, J. Comput. Phys. 238 (2013) 1–31, https://doi.org/10.1016/j.jcp.2012.12.015.
[9] C.M. Cha, J. Zhu, N.K. Rizk, M.S. Anand, A comprehensive liquid fuel injection model for CFD simulations of gas turbine combustors, in: 43rd AIAA Aerospace Sciences Meeting and Exhibit – Meeting Papers, vol. 8, 2005, pp. 1809–1825.
[10] J.S. Chin, A.H. Lefebvre, Steady-state evaporation characteristics of hydrocarbon fuel drops, AIAA J. 21 (1983) 1437–1443, https://doi.org/10.2514/3.8264.
[11] M. Chrigui, J. Gounder, A. Sadiki, A.R. Masri, J. Janicka, Partially premixed reacting acetone spray using LES and FGM tabulated chemistry, Combust. Flame 159 (2012) 2718–2741, https://doi.org/10.1016/j.combustflame.2012.03.009.
[12] D. Darmana, N.G. Deen, J.A.M. Kuipers, Parallelization of an Euler-Lagrange model using mixed domain decomposition and a mirror domain technique: application to dispersed gas-liquid two-phase flow, J. Comput. Phys. 220 (2006) 216–248, https://doi.org/10.1016/j.jcp.2006.05.011.
[13] A. Fanfarillo, D. Del Vento, Notified access in coarray-based hydrodynamics applications on many-core architectures: design and performance, Parallel Comput. 75 (2018) 118–129, https://doi.org/10.1016/j.parco.2018.04.002.
[14] T. Frank, K. Bernert, K. Pachler, H. Schneider, Aspects of efficient parallelization of disperse gas-particle flow predictions using Eulerian-Lagrangian approach, in: 4th International Conference on Multiphase Flow, ICMF, New Orleans, Louisiana, USA, 2001.
[15] S. Freitag, U. Meier, J. Heinze, T. Behrendt, C. Hassa, Measurement of initial conditions of a kerosene spray from a generic aeroengine injector at elevated pressure, in: 23rd Annual Conference on Liquid Atomization and Spray Systems, Brno, Czech Republic, ISBN 978-80-7399-997-1, 2010.
[16] B.M. Gorokhovski, Stochastic sub-grid modeling of drop breakup for LES of atomizing spray, 2001.

[17] W. Gropp, R. Thakur, E. Lusk, Using MPI-2: Advanced Features of the Message Passing Interface, 2nd ed., MIT Press, 1999.
[18] H. Hiroyasu, T. Kadota, Fuel droplet size distribution in diesel combustion chamber, SAE Tech. Pap. 1 (1974) 2615–2624.
[19] G. Houzeaux, M. Garcia, J.C. Cajas, A. Artigues, E. Olivares, J. Labarta, M. Vázquez, Dynamic load balance applied to particle transport in fluids, Int. J. Comput. Fluid Dyn. 30 (2016) 408–418, https://doi.org/10.1080/10618562.2016.1227070.
[20] L. Jofre, R. Borrell, O. Lehmkuhl, A. Oliva, Parallel load balancing strategy for volume-of-fluid methods on 3-D unstructured meshes, J. Comput. Phys. 282 (2015) 269–288, https://doi.org/10.1016/j.jcp.2014.11.009.
[21] W.P. Jones, A.J. Marquis, K. Vogiatzaki, Large-eddy simulation of spray combustion in a gas turbine combustor, Combust. Flame 161 (2014) 222–239, https://doi.org/10.1016/j.combustflame.2013.07.016.
[22] D.K. Kafui, S. Johnson, C. Thornton, J.P.K. Seville, Parallelization of a Lagrangian-Eulerian DEM/CFD code for application to fluidized beds, Powder Technol. 207 (2011) 270–278, https://doi.org/10.1016/j.powtec.2010.11.008.
[23] H. Kahila, A. Wehrfritz, O. Kaario, M. Ghaderi Masouleh, N. Maes, B. Somers, V. Vuorinen, Large-eddy simulation on the influence of injection pressure in reacting Spray A, Combust. Flame 191 (2018) 142–159, https://doi.org/10.1016/j.combustflame.2018.01.004.
[24] B. Kaludercic, Parallelisation of the Lagrangian model in a mixed Eulerian-Lagrangian CFD algorithm, J. Parallel Distrib. Comput. 64 (2004) 277–284, https://doi.org/10.1016/j.jpdc.2003.11.010.
[25] A.B. Liu, D. Mather, R.D. Reitz, Modeling the effects of drop drag and breakup on fuel sprays, SAE Tech. Pap. (1993), https://doi.org/10.4271/930072.
[26] U. Meier, J. Heinze, S. Freitag, C. Hassa, Spray and flame structure of a generic injector at aeroengine conditions, in: Proceedings of ASME Turbo Expo 2011: Power for Land, Sea and Air, GT2011, Vancouver, Canada, 2011.
[27] Message Passing Interface Forum, MPI: a message-passing interface standard, Version 3.1, High Performance Computing Center Stuttgart , in: HLRS, 2015, p. 868, https://www.open-mpi.org/doc/v3.1/, arXiv:a.
[28] J.A. van Oijen, A. Donini, R.J. Bastiaans, J.H. ten Thije Boonkkamp, L.P. de Goey, State-of-the-art in premixed combustion modeling using flamelet generated manifolds, Prog. Energy Combust. Sci. 57 (2016) 30–74, https://doi.org/10.1016/j.pecs.2016.07.001.
[29] R. Pankajakshan, B.J. Mitchell, L.K. Taylor, Simulation of unsteady two-phase flows using a parallel Eulerian-Lagrangian approach, Comput. Fluids 41 (2011) 20–26, https://doi.org/10.1016/j.compfluid.2010.09.020.
[30] S.V. Patankar, Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing Corporation (CRC Press, Taylor & Francis Group), 1980.
[31] C.D. Pierce, P. Moin, Method for generating equilibrium swirling inflow conditions, AIAA J. 36 (1998) 1325–1327, https://doi.org/10.2514/2.518.
[32] T. Poinsot, D. Veynante, Theoretical and Numerical Combustion, 3rd ed., 2005.
[33] C. Rettinger, U. Rüde, Dynamic load balancing techniques for particulate flow simulations, Computation 7 (2019) 9, https://doi.org/10.3390/computation7010009, arXiv:1811.12742.
[34] I.F. Sbalzarini, J.H. Walther, M. Bergdorf, S.E. Hieber, E.M. Kotsalis, P. Koumoutsakos, PPM - a highly efficient parallel particle-mesh library for the simulation of continuum systems, J. Comput. Phys. 215 (2006) 566–588, https://doi.org/10.1016/j.jcp.2005.11.017.
[35] Y. Shigeto, M. Sakai, Parallel computing of discrete element method on multi-core processors, Particuology 9 (2011) 398–405, https://doi.org/10.1016/j.partic.2011.04.002.
[36] H. Sitaraman, R. Grout, Balancing conflicting requirements for grid and particle decomposition in continuum-Lagrangian solvers, Parallel Comput. 52 (2016) 1–21, https://doi.org/10.1016/j.parco.2015.10.010.
[37] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, E.D. Burness, J.M. Cela, M. Valero, Alya: multiphysics engineering simulation toward exascale, J. Comput. Sci. 14 (2016) 15–27, https://doi.org/10.1016/j.jocs.2015.12.007.
[38] D. Veynante, L. Vervisch, Turbulent combustion modeling, Prog. Energy Combust. Sci. 28 (2002) 193–266, https://doi.org/10.1016/S0360-1285(01)00017-X.
[39] S. Yakubov, B. Cankurt, M. Abdel-Maksoud, T. Rung, Hybrid MPI/OpenMP parallelization of an Euler-Lagrange approach to cavitation modelling, Comput. Fluids 80 (2013) 365–371, https://doi.org/10.1016/j.compfluid.2012.01.020.
[40] X. Zhao, J. Wang, S. Zhang, Parallel CFD-DEM for fluid-particle systems, in: ASME 2004 Heat Transfer/Fluids Engineering Summer Conference, vol. 3, 2004, pp. 1–10.
[41] J. Zhu, A. Gupta, C. Nastase, M.S. Anand, Development of an LES approach for compressible reacting flows, in: 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, 2013.