ACS ENGINEERING Au
AN OPEN ACCESS JOURNAL OF THE AMERICAN CHEMICAL SOCIETY

pubs.acs.org/engineeringau
Article

Increasing Computational Efficiency of CFD Simulations of Reactive Flows at Catalyst Surfaces through Dynamic Load Balancing

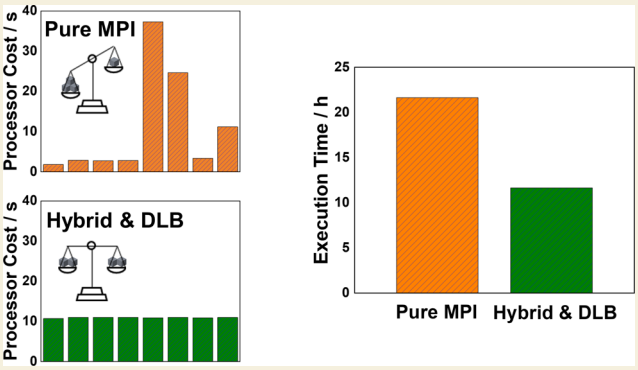
Published as part of ACS Engineering Au virtual special issue "2023 Rising Stars in Chemical Engineering".

Daniele Micale, Mauro Bracconi,* and Matteo Maestri

Cite This: *ACS Eng. Au* 2024, 4, 312−324
Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: We propose a numerical strategy based on dynamic load balancing (DLB) aimed at enhancing the computational efficiency of multiscale CFD simulation of reactive flows at catalyst surfaces. Our approach employs DLB combined with a hybrid parallelization technique, integrating both MPI and OpenMP protocols. This results in an optimized distribution of the computational load associated with the chemistry solution across processors, thereby minimizing computational overheads. Through assessments conducted on fixed and fluidized bed reactor simulations, we demonstrated a remarkable improvement of the parallel efficiency from 19 to 87% and from 19 to 91% for the fixed and fluidized bed, respectively. Owing to this improved parallel efficiency, we observe a significant computational speed-up of 1.9 and 2.1 in the fixed and fluidized bed reactor simulations, respectively, compared to simulations without DLB. All in all, the proposed approach is able to improve the computational efficiency of multiscale CFD simulations paving the way for a more efficient exploitation of high-performance computing resources and expanding the current boundaries of feasible simulations.



KEYWORDS: *reactor modeling, dynamic load balancing, fixed beds, fluidized beds, CFD, multiscale simulations*

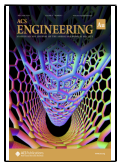
1. INTRODUCTION

The chemical and catalytic reactors are intrinsically multiscale systems whose observed functionality depends on the interplay among the phenomena spanning from the atomic to reactor scales. Our ability to fundamentally understand, analyze, and design the existing and innovative reactors strongly depends on the development of detailed multiscale models.[1] This is achieved by using a first-principles approach at each scale, i.e., solving the characteristic governing equations.

Computational fluid dynamics (CFD), able to properly account for the transport phenomena at the reactor and catalyst scales, is coupled with the accurate description of the chemical kinetics granted by microkinetic models or kinetic Monte Carlo (kMC) simulations.[2] However, the CFD of reactive flows is challenging. On the one side, the modeling of geometries of industrially relevant systems requires complex meshes characterized by a large number of cells (i.e., up to tens of millions). On the other side, detailed homogeneous and heterogeneous mechanisms involve many gas and adsorbed species to comprehensively describe the reactive environment. A large span of time and length scales is involved leading to stiff numerical problems, and the chemical source terms are strongly nonlinear with respect to temperature and possibly to local coverages. Specific numerical methods are needed to tackle this level of complexity, i.e., operator-splitting,[3] which usually requires 70−90% of the overall computational cost.[4]

Different strategies have been proposed in the literature to reduce the computational effort related to multiscale reactor simulations. On one side, on-the-fly approaches based on tabulation (e.g., in situ adaptive tabulation[4−7]) or clustering (e.g., cell agglomeration[8−11]) have been successfully employed. Both approaches aim at reducing the number of detailed chemical calculations, but they still require the full solution of chemistry in a non-negligible number of computational cells. Moreover, their accuracy relies on many tunable parameters whose definition is empirical or requires several trial-and-error analyses.[11] On the other side, machine learning is driving attention to off-the-fly tabulation of chemical source terms based on polynomial approximation,[12,13] ensemble meth-

Received: October 30, 2023
Revised: December 27, 2023
Accepted: December 28, 2023
Published: January 19, 2024



ods,[14,15] or artificial neural networks.[16−18] Such approaches revealed to be very effective in reducing the computational costs and enabled the inclusion of extremely expensive kMC simulations into CFD analysis.[14] These approaches have been employed to speed-up the computations of the reactive source terms in the context of heterogeneous chemistry, whereas their use for the approximation of the entire chemical substep in the operator-splitting approach is mainly confined to homogeneous chemistry simulations.[18−22]

In parallel, some efforts have been made to increase the computational efficiency of the simulations, resulting in a reduction of the computational cost. Such approaches are directed toward the adaptation of the current multiscale models to fully exploit the increasing computational power of the new generation of high-performance computing (HPC) which target exascale performances. This requires addressing several challenges related to pre- and postprocessing, rapid I/O, and efficient numerical methods. Indeed, CFD of reactive flows is already taking advantage of HPC systems since the computational domains, needed to describe the complex three-dimensional systems, are represented by millions of computational cells, which can be only managed through parallel computing. This is usually carried out by geometrically dividing the computational mesh into subdomains assigned to different workers (i.e., CPUs in the message-passing interface—MPI—approach). This approach is highly effective for the solution of transport equations in nonreactive conditions. However, two issues arise and require additional care when parallel computing is employed in reactive simulations.[23,24] First, the distribution of the reactive computational cells in the computational domain can be strongly inhomogeneous due to the geometry of the system. For instance, when a packed bed reactor is simulated, the heterogeneous chemistry is located on the catalytic particles that are not uniformly filling the domain. Therefore, in parallel calculations, the subdomains can hold a significantly different number of computational cells and, consequently, a significantly different computational burden. Second, the cost of chemical calculation is highly related to the local thermochemical state (e.g., partial pressure, temperature, and coverages), which can also be affected on the time-dependent distribution of the phases inside the computational domain in the case of reacting multiphase (e.g., gas−liquid or gas−solid) flows. As a result, the computational load is highly variable across the subdomains, often resulting in imbalanced conditions. In this case, the observed computational cost is related to the most overloaded worker independently of the performances of the others since, at the end of each time step or iteration, all the workers have to be synchronized to enable data exchange. Consequently, most of the workers remain idle, resulting in highly inefficient calculations.

To overcome this issue, several approaches have been proposed. The first strategy involves the run-time and continuous redecomposition of the domain, aiming at achieving a geometrical subdivision characterized by a similar load during the entire simulation.[25,26] The method has been revealed particularly effective when adaptive mesh refinement is employed since a run-time decomposition of the domain is needed due to the continuous modification of mesh density, topology, and size.[25,26] However, it is penalized by the significant cost related to the geometrical operations on the mesh, which can reduce the effectiveness when the mesh is not changing during the simulation. Alternatively, dynamic load balancing (DLB) approaches have been proposed to properly distribute the computational load without redecomposing the computational domain.[23,24] Accordingly, the load is balanced by evenly assigning proper chunks of calculation, deemed to require a similar computational cost, to each of the workers. This is obtained by exchanging the initial conditions of the chemistry calculations between the workers without any modification in the decomposed domains. These methods have shown significant success in the context of detailed simulations of combustion, while any application has been reported in the context of heterogeneous chemistry calculations. In this context, since catalytic reactors are usually characterized by immutable computational grids, the application of procedures related to the run-time and continuous redecomposition of the domain could increase the simulation cost due to the additional overheads related to the frequent geometrical reconstructions and decompositions.[24]

This work aims to develop a DLB approach suitable for detailed CFD simulations of reactive flows at catalyst surfaces. In particular, a numerical strategy that effectively and evenly distributes the heterogeneous chemistry computational load between the processors is specifically conceived for the simulation of fixed and fluidized bed reactors.[3,27] The approach is rooted in a hybrid strategy, which combines the DLB based on geometrical decomposition, where the pieces of information are exchanged by MPI, with shared memory parallelization (based on OpenMP). Accordingly, the computational domain is geometrically divided into portions, which exchange information to balance the computational load, by using the MPI protocol. On top of this, a second layer of parallelization, based on OpenMP, is considered. As such, the evolution and modification of the thermochemical composition of each portion of the domain (i.e., solution of the ODEs of the chemical substep) is further parallelized on multiple processors, as already successfully proposed in the context of Eulerian−Lagrangian CFD simulations.[28,29]

The assessment of the DLB has been initially carried out in a string reactor operated with the CO methanation over the Ni catalyst. First, the simulation has been parallelly executed to highlight the effects of the maldistribution of the computational load on the parallelization efficiency and, consequently, on the computational cost. Then, the DLB based on the MPI protocol has been introduced in the multiscale CFD framework, providing an enhancement of the parallel computing performance and improving the parallelization efficiency from 17 to 84% by using 128 CPUs. The DLB has then been combined with the hybrid parallelization providing a further enhancement of the parallelization efficiency up to 87% with 128 CPUs, leading to a 1.9-fold reduction of the computational cost. Finally, the developed strategy has been applied in the multiscale Euler−Euler simulation of an industrial fluidized bed reactor for the oxidative coupling of methane process. Also in this case, the DLB has been able to evenly distribute the computational load between the processors, leading to a 2.1-fold reduction of the computational effort, working with 128 CPUs when combined with hybrid parallelization.

All in all, this work proposes an effective strategy to improve the parallelization efficiency of multiscale CFD frameworks with a concomitant reduction of computational cost. This paves the way for fundamentally investigating and designing catalytic reactors and processes of industrially relevant scales by fully exploiting the potential of HPC facilities.

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

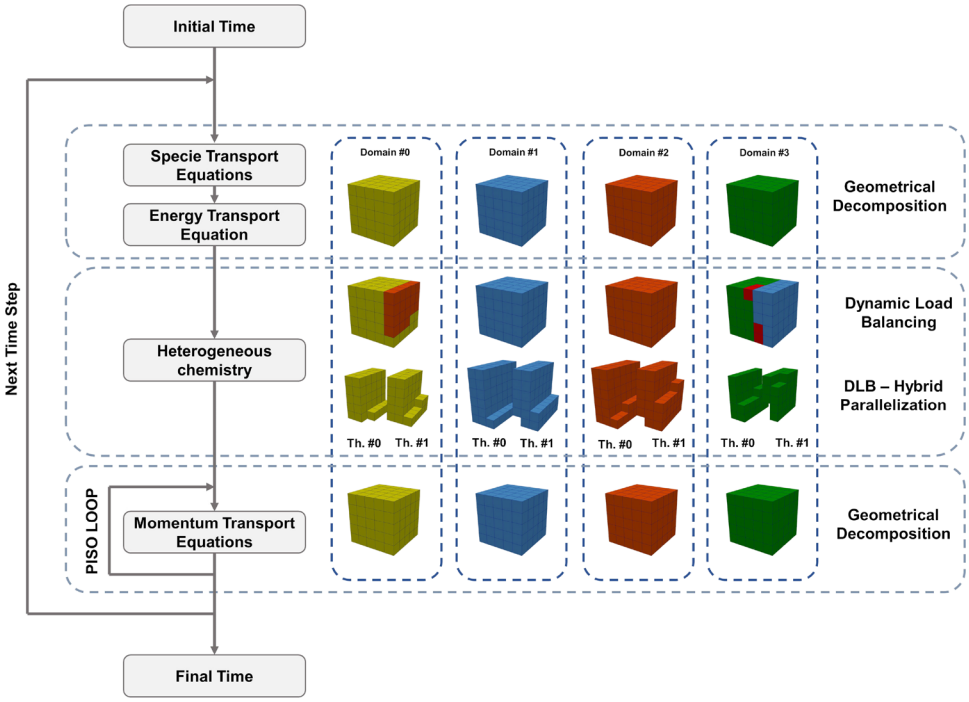


Figure 1. Dynamic load-balancing–hybrid parallelization algorithm. The computational cells are colored according to the domain responsible for their solution. Additional parallelization through OpenMP is highlighted by different threads.

2. NUMERICAL METHODS AND SIMULATION SETUP

This section describes the numerical methods and computational domains adopted in this work. Initially, the load balancing algorithm is described. Then, the introduction of the balancing procedure in CFD frameworks for the multiscale simulations of reactive flows at catalyst surfaces is discussed. Finally, the computational domains used for the analysis of the parallel computing performance and the adopted boundary conditions are presented.

2.1. Dynamic Load Balancing

The DLB is a numerical strategy whose purpose is the balancing of the computational load during parallel calculations.[23,24] This is carried out without geometrically modifying the portion of the computational domain assigned at each processor but by reallocating the computationally expensive operations among the processors pursuing the target of even computational load.

According to this numerical strategy, the balancing procedure is based on the time spent to carry out the expensive calculations during the previous time step, which is deemed to be an estimator of the computational cost in the current step. Based on these pieces of information, the average processor computational load ($\overline{L}$), i.e., the load of each processor which will produce a well-balanced condition, is evaluated as reported in eq 1:

$$\overline{L} = \frac{1}{NP}\sum_{i=1}^{NP}\sum_{n=1}^{NCC_i} l_{n,i} \tag{1}$$

where NP is the number of processors, NCC_i is the number of expensive calculations present in the ith processor, and $l_{n,i}$ is the computational load of the nth expensive calculation in the previous time step.

The processors are hence divided into two groups: overloaded, whose computational load is higher than that of $\overline{L}$, and underloaded, whose computational load is lower than $\overline{L}$. Then, the processors of the two groups are paired, starting from the most overloaded and the most underloaded, and the extent of load that each pair has to exchange (ΔL_{send}) is evaluated as follows:

$$\Delta L_{send} = \min(\overline{L} - L_u, L_o - \overline{L}) \tag{2}$$

where L_u and L_o are the computational load of the underloaded and overloaded processors, respectively.

ΔL_{send} is related to a number of expensive calculations that, when exchanged by the processor pair, will produce a well-balanced state. This number of expensive calculations is set as the minimum one whose overall cost is equal to ΔL_{send}. In doing so, the data exchange is kept at a minimum, aiming at minimizing the computational overheads related to the balancing. Finally, after all the expensive calculations are computed, the solutions are exchanged back to the original processor, which updates the computational domain.

The algorithmic implementation of the DLB is reported in Section S.1.1 of the Supporting Information.

2.2. Multiscale CFD Framework: Operator-Splitting-Based Strategies and Dynamic Load Balancing

Multiscale CFD frameworks employ domain decomposition and parallel calculations to solve the governing equation in complex geometries. In a parallel simulation, the computational grid is divided into many subdomains that are assigned to different processors, which usually communicate by means of the MPI protocol. During each time step, each processor is dedicated to solving the governing equations within its portion of the computational domain while simultaneously exchanging essential information with neighboring processors. In the case of reactive flows, the solution of the governing equation is obtained by using the operator-splitting approach. Accordingly, the transport and reaction contributions in species and energy balance equations are separated into fractional time steps.[3,27] In particular, the reaction operator does not involve any discretization in space, since the chemical process depends on the thermochemical state in each grid point. Consequently, the chemical substep corresponds to a group of decoupled ODE systems describing the temporal evolution of the local variables, i.e., species compositions, site densities, and temperature. The solution of the chemical substep is computationally expensive due to the stiffness and nonlinearity of the source terms.[3,4] In addition, similar to combustion simulations,[23,24] it is the primary source of the load imbalance in parallel calculation due to the dependency of the cost of the ODE solution on the local thermochemical state.[23,24] Thus, DLB is employed for increasing the computational efficiency of calculations related to this substep.

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312–324

To this aim, we developed a numerical library implementing the load balancing algorithm that can be used in whatever operator-splitting framework with the purpose of improving the efficiency of the ODE system solution. The library implements the DLB algorithm proposed by Tekgül et al.[24] adapted to treat reactive flow at catalyst surfaces (i.e., fixed and fluidized bed in single and multiphase conditions).

The flowchart of the algorithm is depicted in Figure 1. During each time step, the local thermochemical state in each cell (i.e., initial values of the ODEs) is collected (e.g., mass fraction, temperature, and pressure) along with the corresponding computational cost spent in the previous step. The pieces of information are sent to the DLB library, which employs the cost of the ODEs as an indicator of the computational load of the processors. Then, the DLB assigns an ensemble of the corresponding ODE systems to each processor, taking advantage of the MPI protocol for data exchange, as described in Section 2.1. Therefore, at the beginning of the integrations, each processor holds a certain number of ODEs which can be different from the number of computational cells, as shown in Figure 1.

In a pure MPI approach, the ODEs are solved sequentially in each of the processors. We identify this approach as pure MPI in the following sections. However, the fully decoupled and independent nature of ODEs results in a problem that can be easily further parallelized (called an *embarrassingly parallel* problem in the literature). In each processor, the solution of the ODEs is further subdivided among an ensemble of threads according to a shared memory parallelization approach based on the OpenMP protocol (see Figure 1), a parallel programming method where multiple processors (or cores) have access to a common memory enabling efficient multithreading. In doing so, it is possible to consider the optimal number of processors required by the transport operators, managed by MPI, and to add an additional performance boost by further subdividing the expensive ODE solution on multiple threads. This approach has been identified as hybrid parallelization in the following sections.

In this work, the hybrid parallelization and dynamic load balancing method have been implemented in the catalyticFoam framework.[3] The algorithms detailing the combination between the chemical substep and the DLB for both the MPI and the hybrid parallelization strategies are, respectively, reported in Sections S.1.2 and S.1.3 of the Supporting Information.

The simulations have been carried out on AMD EPYC 7H12-128 CPUs-256 Gb RAM by using OpenMPI 4.1.1 and OpenMP 3.1.

2.3. Computational Domains and Kinetic Mechanisms

2.3.1. String Reactor. The DLB performances in heterogeneous catalytic systems has been assessed by considering the CO methanation over a nickel-based catalyst, carried out in a string reactor, often used for kinetic measurements and already investigated by reactive CFD.[30,31] The system reactivity is accounted for by using the microkinetic mechanism proposed by Delgado et al.,[32] while the computational domain is a cylindrical reactor (length of 0.042 m and diameter of 0.0044 m), where 10 spherical pellets with a diameter of 0.0025 m are packed. The computational domain is depicted in Figure 2. The domain has been built by using the *snappyHexMesh* tool of

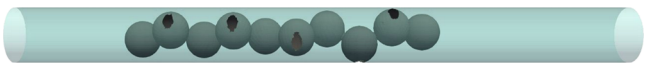


Figure 2. String reactor computational domain consisting of spherical catalytic pellets randomly inserted in a tubular reactor.

OpenFoam[33] by starting from an orthogonal mesh with a cell size equal to 250 μm and by selecting a refinement level of one for the reactor wall and of three for the catalytic pellets, and no layer has been added since a laminar flow regime is experienced in the system. In doing so, the computational grid consists of about 700,000 cells, with the minimum size equal to 31 μm.

As boundary conditions, atmospheric pressure has been fixed at the outlet of the reactor, while zero-gradient has been assumed for the inlet, the lateral wall, and the pellet surface. The superficial velocity has been imposed equal to 0.4 m/s at the inlet, and a no-slip condition has been set at the tube wall and at the surface of the pellets, while a zero-gradient condition has been used at the outlet, assuming a fully developed profile. The species mass fraction and temperature have been imposed as fixed values at the inlet (CO 0.2 v/v, H_2 0.6 v/v, Ar 0.2 v/v, and 673.15 K), while a zero-gradient condition is imposed at the outlet and the reactor wall. Finally, reactive boundary conditions are used at the catalytic pellet surface for both the species mass fraction and temperature.[3,27]

2.3.2. Industrial Fluidized Bed Reactor. The oxidative coupling of methane over the La_2O_3/CaO catalyst has been selected to test the dynamic load balancing in multiscale Eulerian−Eulerian simulations of fluidized systems. The OCM reactivity has been described by using two microkinetic models for both the homogeneous and the heterogeneous chemistries, as reported in our previous work.[34] The simulations have been carried out in a computational domain, as shown in Figure 3. It is composed of a 2 m diameter cylindrical

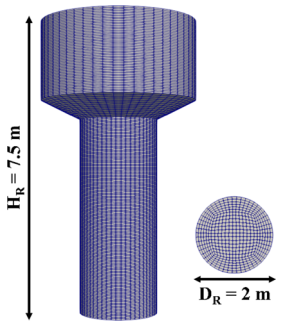


Figure 3. Industrial fluidized bed reactor computational domain.

section (5 m height), followed by an enlargement of the cross section (0.5 m height) up to a second cylindrical section (4 m diameter and 2 m height) to reduce the particle entrainment, resulting in a reactor height of 7.5 m. The computational domain has been discretized by hexahedral cells having a size equal to 500 times the average particle diameter (d_p = 150 μm), leading to a computational grid composed of about 96,000 cells.

At the beginning of the simulation, the bed is packed with a solid fraction equal to 0.62, due to the high reactor-to-particle ratio and an initial height of 1 m. The packed bed was fluidized by using an inert flow of nitrogen injected from the bottom of the reactor. Once steady fluidization is reached, the reactive feed is injected into the reactor.

The systems have been simulated in isothermal conditions. As boundary conditions, atmospheric pressure has been fixed at the top of the reactor, while a zero-gradient condition has been assumed for the lateral walls and the bottom. The superficial velocity has been imposed at the bottom of the reactor equal to 45 times the minimum fluidization velocity, and a no-slip boundary condition has been set at the lateral walls. For the solid phase velocity, the boundary condition proposed by Johnson and Jackson[35] has been imposed at the bottom of the reactor and at the lateral walls. A temperature of 1023.15 K is used. The gas-phase composition has been imposed at the bottom of the reactor according to the operating conditions of the inlet feed stream (CH_4 0.1 v/v, O_2 0.04 v/v, and N_2 0.86 v/v). The Neumann condition was imposed on the remaining boundaries for the species and temperature. Moreover, a specific surface area of the catalyst has been set equal to 1.4487×10^5 m_{cat}^2/m_{cat}^3.

3. RESULTS AND DISCUSSION

This section discusses the computational performances of multiscale CFD simulations with and without load balancing. First, the intensity of the load balancing problems is quantified and highlighted. Then, the improvement granted by the DLB as well as the hybrid parallelization strategy is discussed for the

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

fixed bed test case. Finally, the DLB has been combined with the Euler−Euler modeling approach to show the benefits that can be provided for the parallelization of fluidized systems.

3.1. Assessment of Load Maldistribution in Multiscale CFD Simulations

To quantify the potential load balancing issues related to the different distributions of the cells and operating conditions in each portion of the domain, the string reactor has been geometrically divided into eight subdomains by using two different decomposition methods. On the one hand, the Scotch method has been used. It subdivides the domain into subdomains characterized by a similar number of cells, while attempting to minimize the communications between the processors. The resulting domain is depicted in Figure 4. On the other hand, a simple decomposition of the cylindrical cross section (with four processors in the x direction and four processors in the y direction) has been also employed, as shown in Figure 4.

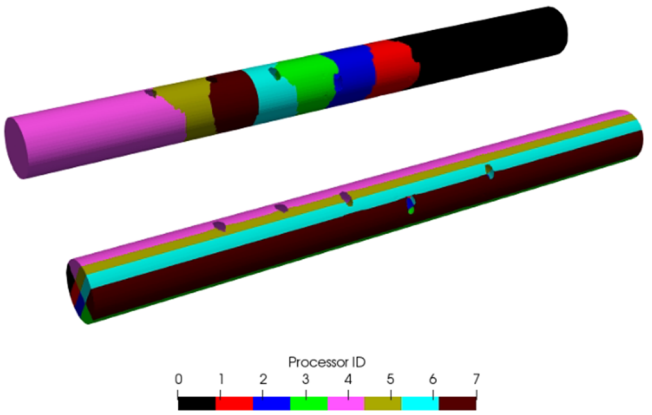


Figure 4. Domain decomposition of the string reactor case to eight processors by using the Scotch method (top) and a simple decomposition of the cylindrical cross section (four processors in x direction and four processors in y direction) (bottom).

The simulations are run following the temporal evolution of the startup of the string reactor, where the reactants are fed into an inert environment from time zero.

Figure 5 reports the trends of the computational cost of the chemistry substep in each of the subdomains for the two decomposition methods. This is a direct measure of the computational burden, and it strongly depends on the processor location and the decomposition methods.

It is worth noting that the observed computational cost of the chemistry substep corresponds to the maximum value among the processors. This occurs because all of the CPUs have to wait for the slowest processor before moving to the next steps independently of the time they spend solving the chemistry. In both methods, the computational cost as a function of the simulation time shows a characteristic trend. Initially, the computational cost is negligible. The duration of this phase is processor-dependent, and it is proportional to the geometrical distance of the subdomain from the inlet. In other words, it corresponds to the time that the reactants require to reach the reactive faces contained in the subdomain. As an example, in the case of the Scotch decomposition method (Figure 5a), processor 4, which contains the inlet patch, shows the shortest initial section, while processor 0, which contains the outlet patch, is reached after about 0.0075 s of simulation. Once the reactants enter the subdomain, the computational time initially increases, due to the gradual onset of the heterogeneous chemistry, followed by a final plateau corresponding to steady-state conditions. Both the slope and the final plateau assume different values in each CPU as a function of the number of reactive cells and the local operating conditions that influence the stiffness of the reactivity in the subdomain. In the Scotch simulation, processor 5 shows the highest computational cost which is related to the presence of intense reactivity due to high temperature and little reactant conversion in that portion of the domain, as shown in Section S.2 of the Supporting Information. Conversely, processor 0 shows the smallest computational time since the reactants have lower concentrations in this subdomain, which is located at the end of the bed. This produces a strong imbalance of the computational time among the processors during the entire simulation that can even result in processors 50 times more

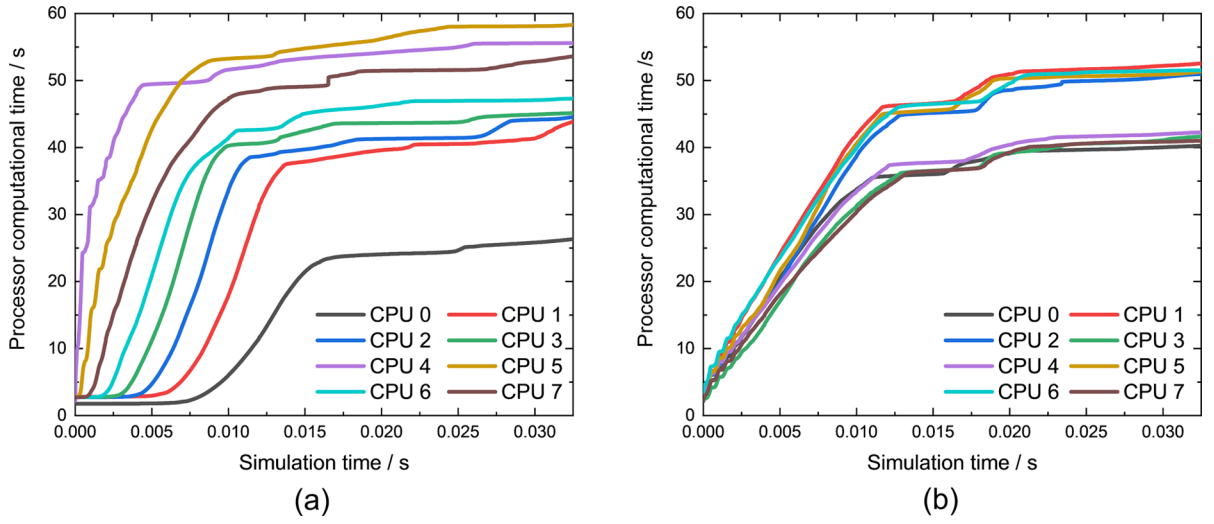


Figure 5. Computational time spent to solve the chemistry for each processor as a function of time by adopting the Scotch (a) and the simple (b) decomposition approach. The colors correspond to the domains shown in Figure 4.

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

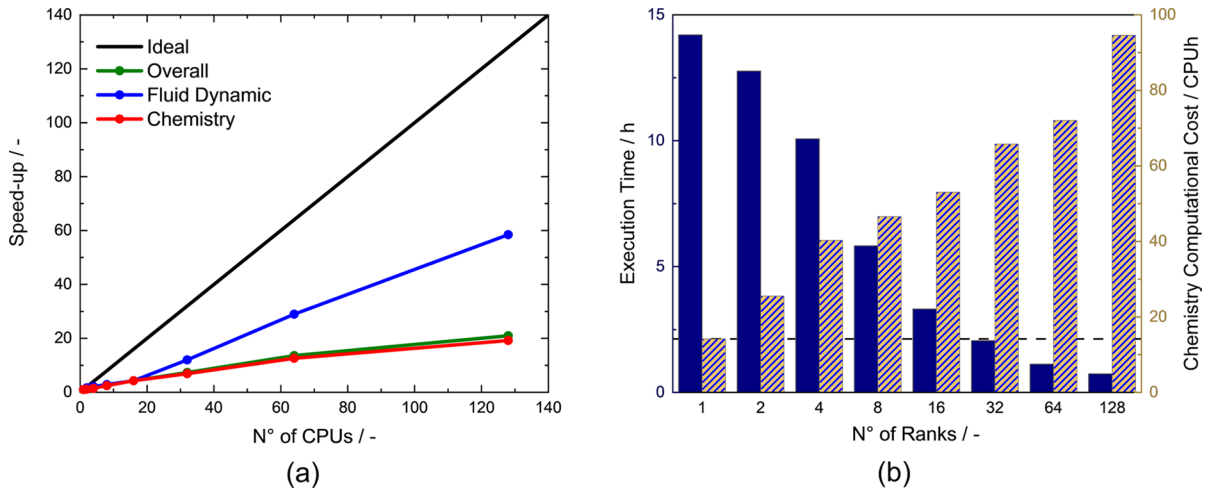


Figure 6. Strong scalability analysis performed on the string reactor computational domain: (a) trends of the speed-up as a function of the number of processors used to geometrically decompose the domain; (b) execution time and the overall chemistry computational cost as a function of the number of processors. The dashed line shows an ideal chemistry computational cost equal to 14.2 CPUh.

loaded than other ones (e.g., CPU 4 and CPU 0 at 0.005 s in Figure 5a).

The simple decomposition shows qualitatively similar profiles. However, they are all characterized by a very short section with a negligible computational cost. This is related to more efficient decomposition, which ensures that the reactive front enters in contact with the reactive faces of each processor at about the same time. Nevertheless, the trends of computational cost are coupled in two distinct groups. This is ascribed to the different numbers of reacting cells that are present in the two groups. The ones characterized by the lower computational times correspond to the topmost and bottom-most domains, which have smaller dimensions with respect to the other ones due to the tube curvature. This results, once again, in a computational burden among the processors that can differ up to 35% (e.g., CPU 7 and CPU 0 at 0.03 s in Figure 5b), resulting in an imbalanced situation.

Therefore, independent of the decomposition method, the computational load in complex domains can be strongly imbalanced. This has a relevant effect on the overall computational time, but it can also have detrimental consequences on the parallelization efficiency. Consequently, a strong scalability analysis[36] has been carried out to investigate the performance of a parallel simulation. The analysis is carried out by considering the Scotch decomposition method since it is the strategy mainly adopted in CFD simulations to subdivide general and complex domains.[37] This is due to the fact that the generated subdomains are characterized by similar cell amounts, and the number of processor−processor interfaces are minimized, leading to the highest parallelization efficiency of the transport-governing equations. Consequently, it usually represents the optimal strategy since it hinders and minimizes the overhead related to the fluid dynamics solution. The scalability has been carried out by an increasing number of subdomains from 2 to 128 and by considering the average computational cost over 1000 time steps and neglecting the simulation startup, which is characterized by additional overheads (e.g., mesh loading) that can hinder the analysis. To guarantee a fair comparison, the simulations were carried out in the same computational node. The effect of the number of CPUs has been quantified by using the speed-up factor SU_i, evaluated according to eq 3:

$$SU_i = \frac{\tau_1}{\tau_i} \tag{3}$$

where i represents the number of CPUs used to run the simulations, and τ_1 and τ_i are the averaged computational times.

Three different speed-up factors have been considered and compared to the ideal scaling performances (black line in Figure 6a). The overall speed-up (green line in Figure 6a) has been calculated by using the time spent to solve the entire time step, which accounts for both the solution of the chemical substep, the species and energy transport, and the Navier−Stokes equations. The chemistry speed-up (red line in Figure 6a) has been computed by considering the computational time of the sole chemical substep, while the fluid dynamic speed-up (blue line in Figure 6a) considers the solution of the species and energy transport and the Navier−Stokes equations.

Figure 6a shows the speed-up as a function of the number of CPUs. By increasing the number of CPUs, the speed-up monotonically increases as expected. However, the parallelization efficiency decreases as well as shown by the growing deviations between the ideal and actual speed-ups. This deviation can be related to two distinct phenomena: the increment of the communication cost between the processors and the growing imbalance of the computational load. The decrease in the parallelization efficiency on the transport side is mainly related to the increment of the communication between the processors that gradually increases with the number of CPUs due to the higher number of exchanged information at the boundaries of these subdomains.[38] This magnifies the MPI communication overheads, leading to the reduction of computational efficiency.

Differently from the transport equations, the chemistry calculations are in principle an embarrassingly parallelizable problem since, in the operator-splitting approach, they are represented by batch reactors that depend on the local cell conditions and do not require any communication between the processors. Hence, it would be expected that the parallelization efficiency of the chemical calculations should be close to a quasi-ideal speed-up. A severe reduction of the chemistry parallel performances is, however, observed in the simulations. This is due to the strong imbalance of the processor workload

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

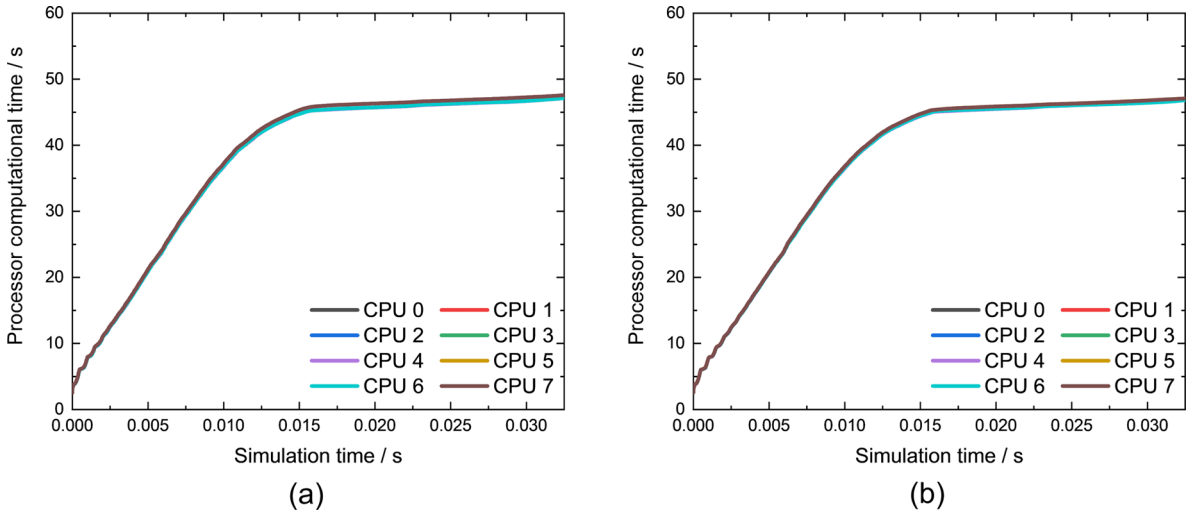


Figure 7. Computational time spent to solve the chemistry with the dynamic load balancing procedure for each processor as a function of time by adopting the Scotch (a) and the simple (b) decomposition approach.

related to both the inhomogeneous operating conditions in the reactor (i.e., different local partial pressures and temperatures) and the different numbers of reacting cell in each subdomain. Intuitively, the imbalance grows with the number of CPUs since smaller computational domains are more prone to show very different average working conditions and number of reacting elements. In this case, the most overloaded processor controls the computational time of the reaction step. Thus, it strongly affects the overall computational time needed to run the simulation.

Figure 6b shows the execution time that, as expected, decreases with the number of processors consequently to the simulation speed-up. Moreover, the overall computational cost t_i, related to the solution of the chemistry as a function of the number of subdomains, is reported. The overall chemistry computational cost, expressed in CPUh, has been calculated by multiplying the average chemistry computational time (τ_i) for the number of CPUs (i) and the number of time steps (n), as reported in eq 4.

$$t_i = \tau_i \cdot i \cdot n \tag{4}$$

This quantity corresponds to a sort of energy consumed by the simulation for the solution of the chemical substep. Ideally, the overall computational cost should be independent of the number of CPUs since more processors should proportionally require less clock time to carry out the calculations and equal to the value for a serial simulation (i.e., 14.2 CPUh, dashed line in Figure 6b). However, the computational cost increases more than linearly with the CPUs, leading to a overall computational cost 10 times higher than the ideal value by using 128 CPUs, further showing the poor parallelization efficiency. In fact, despite the spreading of the ODE integration between the processors reducing the execution time, the load imbalance hinders the parallel computing performance with a consequent increment of the simulation computational burden.

3.2. Assessment of Dynamic Load Balancing in Heterogeneous Catalytic Systems

3.2.1. Pure MPI Approach. To tackle the load imbalance problem, the catalyticFoam framework[27] has been combined with the dynamic load balancing (DLB),[23,24] as described in Section 2.1.

The assessment of the performance of the DLB approach is initially carried out by simulating the entire string reactor decomposed in eight subdomains with both the Scotch method (Figure 4) and the simple decomposition (Figure 4). The profiles of the computational cost in each processor as a function of the simulation time are reported in Figure 7.

The computational cost gradually increases with the simulation time until a plateau is obtained. It is worth noting that negligible differences are observed between the processors, independent of the decomposition method. This is related to the effect of the DLB that efficiently splits the computational burden by redistributing the solution of the chemistry substep among the processors. For example, after 0.002 s of simulations, around 57,000 (36% of the total) and 6,250 (4% of the total) reactive cells are exchanged between the processors with the Scotch and simple decomposition methods, respectively. The lower number of cells exchanged in the case of the simple decomposition is due to a minor imbalance of this setup, as shown in Section 3.1. It is worth emphasizing that by redistributing the chemical workload, the number of computational cells solved by each processor changes over time. By considering the Scotch domain decomposition, an average cell-to-CPU ratio of around 87,000 is obtained at the beginning of the simulation where the pure geometrical decomposition of the mesh associates the cells to the processors. By redistributing the chemical workload, the DLB alters the cell-to-CPU to around 75,000−92,000 at 0.002 s, 81,000−95,000 at 0.014 s, and 81,000−94,000 at 0.034 s, guaranteeing, however, even chemical workload. Consequently, this results in higher computational efficiency, reducing the costs of the simulations.

Figure 8 shows the speed-up, evaluated as the ratio between the time spent to solve the time step without and with the DLB, obtained in the chemistry calculation by the adoption of the DLB. Despite the simple geometry, a speed-up of around 1.3 has been obtained for the simple domain decomposition, while the Scotch method shows a maximum at around 0.001 s equal to 4.

The efficient redistribution of the computational burden not only speeds up the simulations but is also able to improve the scalability of the multiscale CFD simulations. To prove this, a strong scalability analysis[36] has been performed in analogy with

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

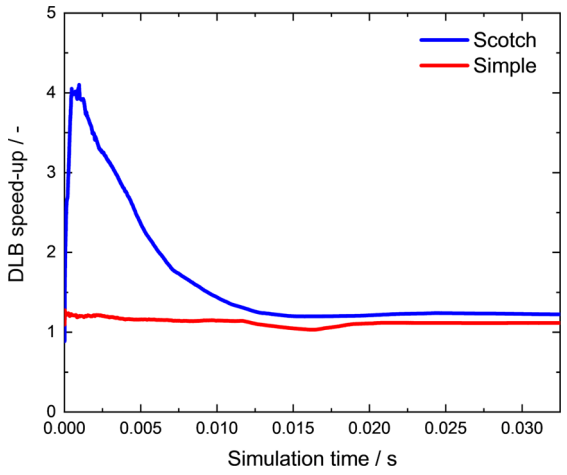


Figure 8. Chemistry speed-up as a function of time provided by the dynamic load balancing procedure in the string reactor.

Section 3.1. It is worth noticing that when the chemistry is solved with the DLB procedure, the chemical substep consists of additional operations on top of the chemistry calculations related to the DLB algorithm. An additional speed-up that accounts for the computational time spent by integrating the chemical ODE systems was introduced to quantify the effects of the DLB operations on the parallel computing performance. Figure 9a shows the scalability results that reveal that, by increasing the number of CPUs, the speed-up monotonically increases as expected. The fluid dynamic speed-up shows similar performances to the one obtained without the DLB (cf. Figure 6a). The load-balancing procedure affects the distribution of the chemical calculations among the processors, but it does not modify the domain geometrical decomposition, which controls the parallel efficiency for the solution of the transport equations. The chemistry speed-up is however characterized by higher values (see Figure 6a for reference), proving that a higher parallelization efficiency is obtained thanks to the DLB. For example, by using 128 CPUs, a 107-fold speed-up is obtained by using the DLB, in contrast to the 21-fold calculated without the DLB. The superior performances are due to the more homogeneous workload shared by the processors that compensates for the additional computational effort related to the exchange of information required by the DLB. Indeed, in this case, the chemistry parallelization efficiency is no longer only dependent on the distribution of the computational load but also on the communication cost of the MPI protocol between the processors that have to exchange information during the DLB. The intensity of these overheads can be observed by the deviations between the trends of the speed-up of the ODE integration and the chemistry in Figure 9a. The two trends are superimposed as a result of negligible communication overheads. The effect of the load balancing is also highlighted by the trend of the overall chemistry computational cost, evaluated according to eq 4, and reported in Figure 9b. The overall computational cost is slightly dependent on the number of CPUs. The cost is close to the ideal value when the number of domains is less than 32. However, a further increment of the number of subdomains gradually increases the overall computational cost, reducing the efficiency of the calculations. On one side, this additional cost is related to memory overheads occurring due to the simultaneous solution of several ODEs which increase along with the number of subdomains[39] and mainly depend on the architecture of the computational node rather than on the numerical framework.[24]

On the other hand, the additional overheads can be related to an inefficient balancing on a small amount of computational load that is more likely to occur when the number of processors is large. Indeed, the balancing is achieved by redistributing the computational load among the subdomains based on the computational cost of the previous time step, that is, the only available information. This is usually a good estimator of the cost in the following time step in each cell. However, the transport and reactive phenomena might alter the local thermochemical conditions, leading to positive or negative variation between the previous and actual computational cost of each cell to such an extent that makes the estimation imprecise. We have observed that this effect is mitigated and tends to cancel out when high cell-to-subdomain ratios are employed (i.e., when several ODE integrations are

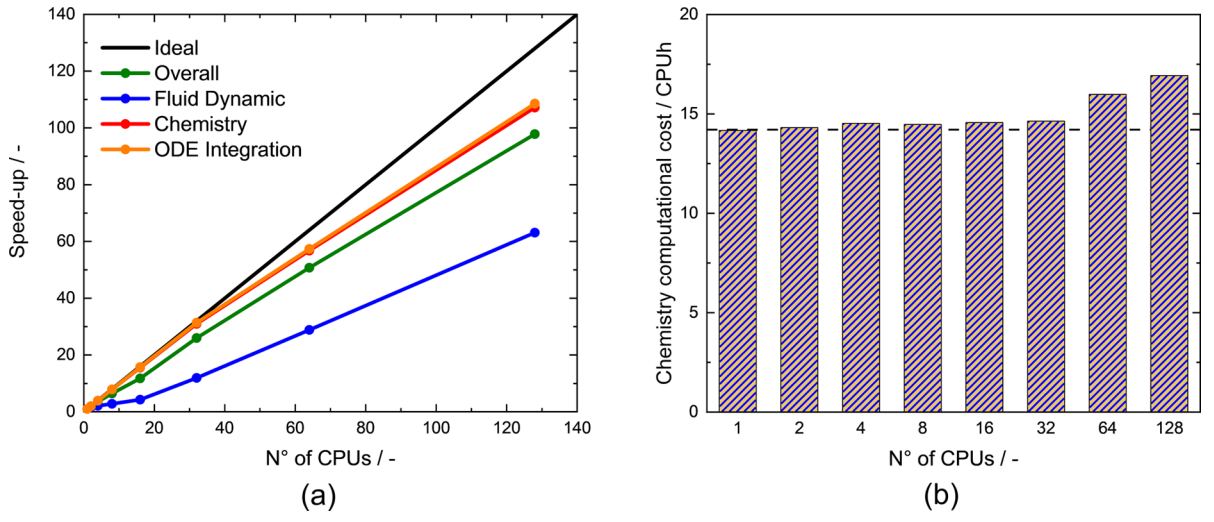


Figure 9. Strong scalability analysis performed on the string reactor computational domain with the dynamic load balancing procedure: (a) trends of the speed-up as a function of the number of processors used to geometrically decompose the domain; (b) overall chemistry computational cost as a function of the number or processors. The dashed line shows the ideal chemistry computational cost equal to 14.2 CPUh.

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

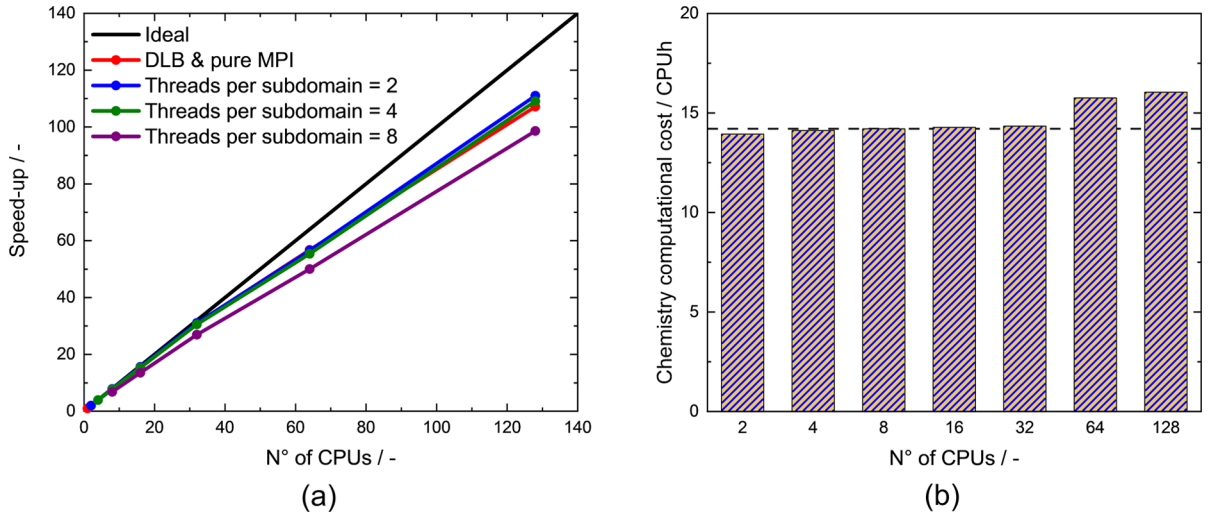


Figure 10. Strong scalability analysis performed on the string reactor computational domain with the dynamic load balancing procedure and the hybrid parallelization: (a) trends of the speed-up as a function of the number of CPUs used to solve the domain; (b) overall chemistry computational cost as a function of the number of CPUs, considering a shared memory parallelization with two threads per subdomain. The dashed line shows the ideal chemistry computational cost equal to 14.2 CPUh.

carried out in each subdomain) because overall the positive and negative variations become negligible. Consequently, this effect appears to be more evident by increasing the number of processors, i.e., the cell-to-subdomain ratio decreases due to a higher sensitivity to the local variation of the computational cost. In this simulation, the fraction of the computational time associated with the computational cost variation is smaller than the 5% of the time of the chemical step, resulting in a negligible effect on the overall performances when less than 32 CPUs are considered (i.e., about 1 and 2.5% for 2 and 32 subdomains, respectively). On the other side, these overheads rise to 12% of chemistry time when 128 CPUs are employed, affecting the overall efficiency. Consequently, despite the DLB, the parallelization efficiency of the chemistry decreases at a high number of CPUs. Hence, further enhancement of the parallelization performance of multiscale CFD simulations can be achieved by reducing these overheads.

3.2.2. Hybrid Approach. The additional overheads observed when the cell-to-subdomain ratio is small (e.g., for the number of CPUs higher than 32 in the string reactor) reduce the parallelization efficiency of the framework, hindering the overall speed-up provided by the DLB and increasing the execution time of the simulations. Hence, this work proposes a second strategy to further improve the performances by guaranteeing a high cell-to-subdomain ratio when a large number of CPUs is employed to solve the chemistry as well. The strategy aims at reducing the subdomains generated by the geometrical decomposition methods with a consequent increment in the number of ODEs to be integrated into each subdomain, without limiting the number of CPUs used to solve the integrations. This can be achieved using a hybrid parallelization approach that combines the MPI with a shared memory OpenMP. The idea is to decompose the mesh in a number of subdomains that minimize the parallelization overheads. On top of this, the ODE integration of each subdomain is solved by the combined action of different CPUs, called threads, through the OpenMP parallelization. In doing so, the ODE integrations are carried out by a certain number of CPUs which determine the simulation speed-up, but each subdomain contains a large chunk of the ODE integrations, minimizing the effects of the computational cost variations. Initially, the optimal number of threads per subdomain was identified by carrying out a scalability analysis. It has been observed that the best performances, i.e., shared memory parallelization above 80%, are achieved by considering from two to eight threads per subdomain independently from the number of subdomains. Additional details are reported in Section S.3 of the Supporting Information.

Then, a strong scalability analysis[36] of the combined hybrid parallelization and DLB has been carried out by considering a number of threads per subdomain ranging from two to eight. The trends of the chemistry speed-up are reported in Figure 10a together with the one obtained with the pure MPI parallelization (Section 3.2.1). The speeding-up factors grow along with the number of CPUs used to solve the system. On the one hand, the solution of each subdomain with two or four threads leads to speed-up values superimposed with the pure MPI when less than 64 CPUs are used. Then, superior performances are obtained for a larger number of CPUs by adopting the hybrid parallelization with two or four threads per subdomain. On the other hand, speed-up values lower than pure MPI are always obtained by solving each subdomain with eight threads. This is because the shared memory parallelization with eight threads has an efficiency of about equal to 85% (Section S.3 of the Supporting Information). In this case, the overheads related to the hybrid parallelization are higher with respect to the time related to the variations of the computational cost, leading to lower speed-up factors. Conversely, the excellent parallelization efficiency (i.e., close to 100%) achieved by solving the subdomain with two or four threads does not provide additional overheads. Thus, the hybrid parallelization does not hinder the parallelization performance, while the overheads of the DLB are negligible (i.e., until 32 CPUs), but it improves the performance when these costs limit the speed-up (i.e., 64 and 128 CPUs). In this way, a chemistry speed-up equal to 111 has been obtained by simulating the system with the hybrid parallelization (i.e., 64 processors with two threads per subdomain) in contrast with

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

the 107-fold speed-up obtained by only geometrically dividing the computational domain into 128 portions.

The improvement of the parallel computing performance can be also observed by analyzing the computational cost needed to solve the time steps using the hybrid parallelization with two threads per subdomain, as reported in Figure 10b. The computational time is only slightly dependent on the number of CPUs and is in accordance with the ideal value by using less than 32 CPUs. Then, a gradual growth is observed by further increasing the CPUs. The increment is, however, smaller with respect to the one found by only geometrically dividing the computational domain. In particular, the hybrid parallelization with 128 CPUs provides a computational time equal to 16 CPUh in contrast to the 17 CPUh spent with geometrical domain decomposition. Thus, the computational overheads decrease from 2.8 to 1.8 CPUh by adopting the hybrid parallelization.

Finally, the effects of the proposed approach on the parallel computing performance were shown by solving the entire transient behavior of the string reactor with 128 CPUs. For this purpose, three different simulations have been considered. The first two simulations used a pure MPI parallelization by geometrically decomposing the domain by means of the Scotch method and solved the chemical step with and without the DLB procedure. Conversely, the combination of the DLB and the hybrid parallelization (i.e., 64 processors with two threads per subdomain) was used in the third case. Figure 11 reports

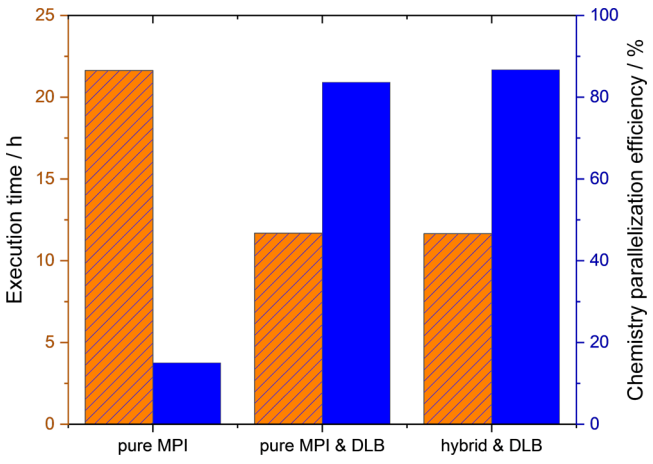


Figure 11. Execution time spent to solve the entire transient of the string reactor and chemistry parallelization efficiency.

the overall simulation execution time, accounting for the solution of both the fluid dynamics and the chemical phenomena. A 1.9-fold reduction of the execution time is achieved by using the DLB with both the geometrical and hybrid parallelization, showing the improvement provided by balancing the computational load of multiscale CFD simulations. In particular, the minimization of effects of the computational cost variations provided by the combination of hybrid parallelization and the DLB procedure, which leads to the higher chemistry parallelization efficiency reported in Figure 11, is directly proportional to the number of CPUs adopted to carry out the multiscale simulation. Consequently, it allows for improving the parallelization efficiency of computational demanding simulation performed with thousands of CPUs on HPC.

3.3. Application to an Industrial-Scale Fluidized Bed Reactor

The previous analysis shows the results of combining multiscale CFD simulations with the DLB procedure in the context of fixed bed reactors. To further exemplify the effectiveness of the approach, we extended the analysis to an industrial fluidized bed reactor. The simulations are carried out by considering a Euler−Euler (EE) framework, developed in our previous work,[34] which implements the multiphase operator-splitting methodology to solve the governing equations. Accordingly, the operators in the governing equations are separated and solved in fractional substeps. In particular, the chemical step of MOS considers both heterogeneous and homogeneous reactivities along with the interphase mass transport.

The reactive simulation is initialized by fluidizing the bed with an inert gas stream until pseudo-steady-state conditions are achieved. Then, the reactants are fed to the reactor, and the evolution of the reacting environment is considered. The computational domain, shown in Figure 3, has been decomposed by means of the Scotch method, and the time spent by each CPU to solve the reaction step has been collected.

The system was initially simulated by means of a pure MPI parallelization, dividing the system into 128 subdomains, with and without the DLB procedure, and the computational time trends of the most overloaded and underloaded processors are shown in Figure 12. When the simulation is carried out without the DLB (Figure 12a), the computational cost of the most overloaded processor immediately increases, while the most underloaded one shows negligible computational cost in the first 5 s of simulation, until the reactive front reaches the farthest subdomain from the inlet. Even after the initial transient, the computational load is also different during the pseudosteady state (i.e., after 20 s). This is due to the very different thermochemical states in the two processors. The most overloaded represents a computational domain at the bottom of the reactor which is reached by the reactants since the very beginning and is characterized by the highest solid fraction. Conversely, a computational domain on the upper part of the reactor, which is characterized by a low catalyst content, corresponds to the underloaded one. As a consequence, a severe imbalance is obtained penalizing the efficiency of the calculation. Figure 12b shows the same trends when DLB is employed. The DLB redistributes the computational burden among the processors, leading to even computational costs. This is achieved by exchanging around 13,400 computational cells (14% of the total) when strong imbalanced conditions are observed (i.e., after 1 s), while around 4,200 computational cells (4.4% of the total) are exchanged after 20 s.

Finally, three different parallelization settings have been considered. The first two settings employed a pure MPI parallelization and solved the chemical step with and without the DLB procedure, while the third one consisted of the combination of the DLB and the hybrid parallelization by considering two threads per subdomain. These settings have been tested by considering two different numbers of CPUs (i.e., 64 and 128), and the execution times of the entire transient of the industrial fluidized bed (i.e., 25 s) are reported in Figure 13.

When 64 CPUs are used, the pure MPI-DLB provides a computational speed-up of around 1.4 times by distributing the

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

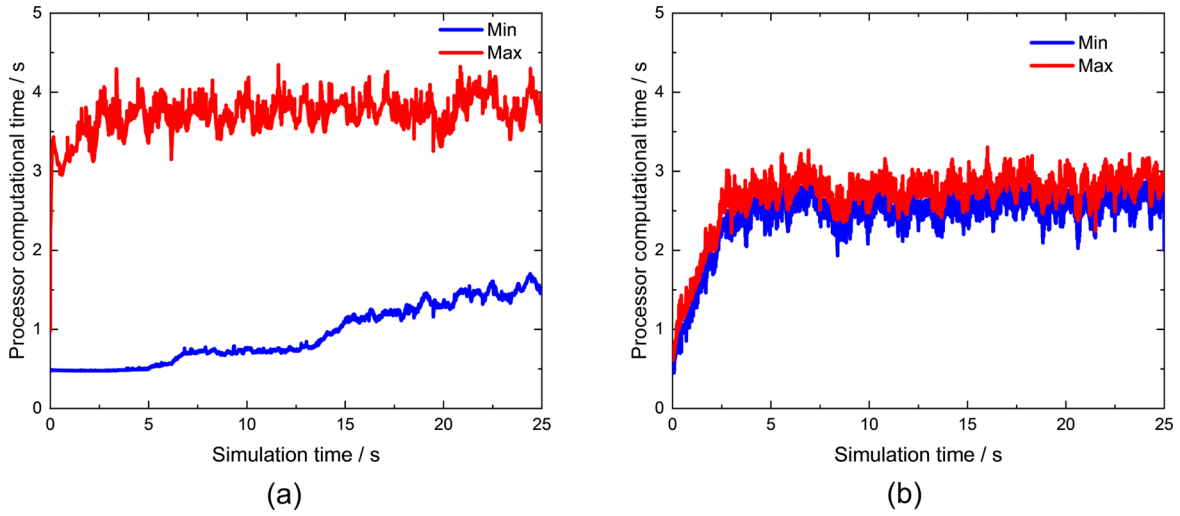


Figure 12. Computational time spent by the processors to solve the chemistry as a function of time by adopting the Scotch decomposition approach without (a) and with (b) the dynamic load-balancing procedure.

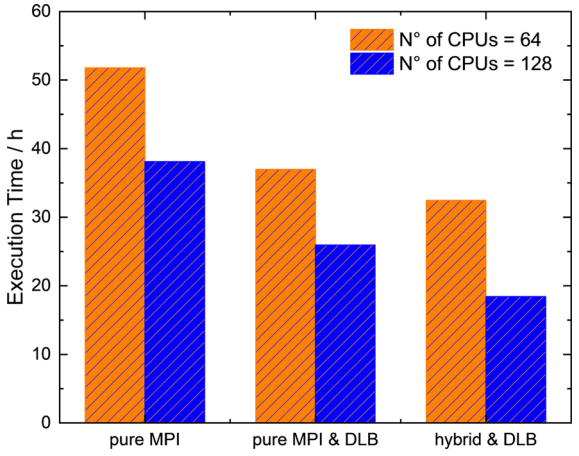


Figure 13. - Execution time spent to solve the entire transient of the industrial fluidized bed reactor by using 64 and 128 CPUs.

computational burden of the stiff ODE integrations between the processors, while the benefit rises to 1.6 when the combination between the DLB and the hybrid parallelization is adopted. The improvements are even more pronounced when the simulation is carried out by means of 128 CPUs. Indeed, a 2.1-fold reduction of the execution time is obtained by combining the hybrid parallelization with the DLB with respect to the 1.5-fold obtained in the case of the pure MPI-DLB. This is related to the capability of the hybrid approach to mitigate the additional overheads that can occur during the balancing procedure, which reduces the efficiency in the pure MPI-DLB approach, which is expected to provide further benefits in larger and more complex simulations.

The DLB is thus able to improve the parallel computing performances also in the simulations of fluidized systems, and its combination with the hybrid parallelization is able to achieve even more computational cost benefits, extending the current boundary of multiscale simulations.

4. CONCLUSIONS

In this work, the catalyticFoam multiscale CFD framework has been combined with both a dynamic load-balancing procedure and hybrid (MPI and OpenMP) parallelization to improve their parallel computing performance. The intensity of the load imbalance obtained by geometrically decomposing the domain is initially quantified. In particular, the load imbalance is strongly dependent on the strategy selected to geometrically decompose the domain (e.g., simple and Scotch) and can also lead to a 10-fold increment of the overall chemistry computational cost compared with the balanced simulation.

The DLB improves the distribution of the computational load strongly, leading to a similar computational cost in all the processors independently of the decomposition method. This also enhances the computing performance of the multiscale CFD simulations by improving the parallel efficiency from 17 to 84%, providing an overall simulation speed-up of 1.9 times in the case of a string reactor for the CO methanation.

However, the analysis of the overall computational costs highlights the presence of nonreducible overheads when the number of CPUs is large. These overheads are ascribed to inefficient balancing due to the variation of the expected computational cost, the key parameter for the balancing algorithm. Thus, a hybrid parallelization approach has been proposed to reduce these overheads, where a shared memory parallelization approach has been added on top of the conventional MPI protocol for the solution of the chemical substep. By combining the hybrid parallelization and the DLB, a further enhancement of the parallelization performance is obtained, which results in similar speed-ups but higher parallelization efficiencies, which are above 86% in the case of the string reactor.

The hybrid−DLB has also been employed in a multiscale Euler−Euler simulation of a fluidized bed reactor. It was revealed to provide higher computational performance, leading to a 2.1-fold reduction of the execution time with a concomitant increment of the parallelization efficiency up to 91%.

The hybrid−DLB approach is revealed to be an effective strategy to improve the parallelization efficiency of multiscale CFD simulations fostering the exploitation of the HPC facilities. However, several developments are still required to take full advantage of the new generation of supercomputers (i.e., exascale) for the CFD simulations of catalytic reactors. To effectively manage massively parallel high-performance systems, current numerical methods must undergo significant

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324

enhancement, which requires joint efforts with mathematicians and computer scientists. Key aspects include advanced numerical methodologies, efficient preprocessing (e.g., domain decomposition), postprocessing (e.g., data reconstruction), and rapid I/O as well as optimizing communication across processors and ensuring efficiency at large scales. On the other hand, the CFD community should also explore the opportunity to take advantage of graphical processing units (GPUs) that have allowed breakthrough improvements in machine learning and atomistic calculations. GPUs are expected to provide significant benefits in the solution of both the linear system and the ODE integration. However, careful management of data transfers and offloading between CPUs and GPUs is essential to achieving high performances.

As a whole, this work proposes an effective strategy to improve the parallelization efficiency of multiscale reactive CFD frameworks, leading to a more efficient exploitation of the HPC facilities.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acsengineeringau.3c00066.

Detailed description of the algorithm implementation; concentration and temperature distribution into the string reactor; and assessment of the hybrid parallelization performances (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Mauro Bracconi − *Laboratory of Catalysis and Catalytic Processes, Dipartimento di Energia, Politecnico di Milano, 20156 Milano, Italy;* orcid.org/0000-0001-7643-3214; Email: mauro.bracconi@polimi.it

Authors

Daniele Micale − *Laboratory of Catalysis and Catalytic Processes, Dipartimento di Energia, Politecnico di Milano, 20156 Milano, Italy;* orcid.org/0000-0002-4343-1201

Matteo Maestri − *Laboratory of Catalysis and Catalytic Processes, Dipartimento di Energia, Politecnico di Milano, 20156 Milano, Italy;* orcid.org/0000-0002-8925-3869

Complete contact information is available at: https://pubs.acs.org/10.1021/acsengineeringau.3c00066

Author Contributions

CRediT: **Daniele Micale** conceptualization, data curation, formal analysis, investigation, methodology, software, writing-original draft; **Mauro Bracconi** conceptualization, data curation, formal analysis, funding acquisition, investigation, methodology, software, supervision, writing-original draft, writing-review & editing; **Matteo Maestri** conceptualization, formal analysis, funding acquisition, investigation, methodology, project administration, supervision, writing-original draft, writing-review & editing.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The project team leading this work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement no. 814416/ReaxPro: "Software Platform for Multiscale Modelling of Reactive Materials and Processes"). Computational time at CINECA (Italy) and SURF (the Netherlands) is acknowledged. Maksim Masterov and Satish Kamath (SURF) are gratefully acknowledged for the fruitful discussions. This invited contribution is part of the ACS Engineering Au special issue for the 2023 Rising Stars in Chemical Engineering.

■ REFERENCES

(1) Wehinger, G. D.; Ambrosetti, M.; Cheula, R.; Ding, Z.-B.; Isoz, M.; Kreitz, B.; Kuhlmann, K.; Kutscherauer, M.; Niyogi, K.; Poissonnier, J.; Réocreux, R.; Rudolf, D.; Wagner, J.; Zimmermann, R.; Bracconi, M.; Freund, H.; Krewer, U.; Maestri, M. Quo Vadis Multiscale Modeling in Reaction Engineering? − A Perspective. *Chem. Eng. Res. Des.* **2022**, *184*, 39−58.

(2) Bracconi, M. Intensification of Catalytic Reactors: A Synergic Effort of Multiscale Modeling, Machine Learning and Additive Manufacturing. *Chem. Eng. Process. - Process Intensif.* **2022**, *181*, No. 109148.

(3) Micale, D.; Ferroni, C.; Uglietti, R.; Bracconi, M.; Maestri, M. Computational Fluid Dynamics of Reacting Flows at Surfaces: Methodologies and Applications. *Chemie Ing. Technol.* **2022**, *94* (5), 634−651.

(4) Bracconi, M.; Maestri, M.; Cuoci, A. In Situ Adaptive Tabulation for CFD Simulation of Heterogeneous Reactors Based on Operator-Splitting Algorithm. *AIChE J.* **2017**, *63*, 95−104.

(5) Lu, L.; Pope, S. B. An Improved Algorithm for in Situ Adaptive Tabulation. *J. Comput. Phys.* **2009**, *228* (2), 361−386.

(6) Kumar, A.; Mazumder, S. Adaptation and Application of the In Situ Adaptive Tabulation (ISAT) Procedure to Reacting Flow Calculations with Complex Surface Chemistry. *Comput. Chem. Eng.* **2011**, *35* (7), 1317−1327.

(7) Uglietti, R.; Bracconi, M.; Maestri, M. Coupling CFD-DEM and Microkinetic Modeling of Surface Chemistry for the Simulation of Catalytic Fluidized Systems. *React. Chem. Eng.* **2018**, *3* (4), 527−539.

(8) Goldin, G. M.; Ren, Z.; Zahirovic, S. A Cell Agglomeration Algorithm for Accelerating Detailed Chemistry in CFD. *Combust. Theory Model.* **2009**, *13* (4), 721−739.

(9) Rebughini, S.; Cuoci, A.; Dixon, A. G.; Maestri, M. Cell Agglomeration Algorithm for Coupling Microkinetic Modeling and Steady-State CFD Simulations of Catalytic Reactors. *Comput. Chem. Eng.* **2017**, *97*, 175−182.

(10) Zhou, L.; Zhong, L.; Qin, W.; Zhao, W.; Wei, H. Application of Cell Agglomeration Algorithm Coupled with Dynamic Adaptive Chemistry for Transient Engine Simulation of Diesel Fuel. *Fuel* **2018**, *234*, 1313−1321, DOI: 10.1016/j.fuel.2018.07.155.

(11) Uglietti, R.; Bracconi, M.; Maestri, M. Development and Assessment of Speed-up Algorithms for the Reactive CFD-DEM Simulation of Fluidized Bed Reactors. *React. Chem. Eng.* **2020**, *5* (2), 278−288.

(12) Matera, S.; Maestri, M.; Cuoci, A.; Reuter, K. Predictive-Quality Surface Reaction Chemistry in Real Reactor Models: Integrating First-Principles Kinetic Monte Carlo Simulations into Computational Fluid Dynamics. *ACS Catal.* **2014**, *4* (11), 4081−4092.

(13) Klingenberger, M.; Hirsch, O.; Votsmeier, M. Efficient Interpolation of Precomputed Kinetic Data Employing Reduced Multivariate Hermite Splines. *Comput. Chem. Eng.* **2017**, *98*, 21−30.

(14) Bracconi, M.; Maestri, M. Training Set Design for Machine Learning Techniques Applied to the Approximation of Computationally Intensive First-Principles Kinetic Models. *Chem. Eng. J.* **2020**, *400* (April), No. 125469.

(15) Partopour, B.; Paffenroth, R. C.; Dixon, A. G. Random Forests for Mapping and Analysis of Microkinetics Models. *Comput. Chem. Eng.* **2018**, *115*, 286−294.
(16) Hayes, R. E.; Mok, P. K.; Mmbaga, J.; Votsmeier, M. A Fast Approximation Method for Computing Effectiveness Factors with Non-Linear Kinetics. *Chem. Eng. Sci.* **2007**, *62* (8), 2209−2215.
(17) Döppel, F. A.; Votsmeier, M. Efficient Neural Network Models of Chemical Kinetics Using a Latent Asinh Rate Transformation. *React. Chem. Eng.* **2023**, *8* (10), 2620−2631.
(18) Sharma, K. G.; Kaisare, N. S.; Goyal, H. A Recurrent Neural Network Model for Biomass Gasification Chemistry. *React. Chem. Eng.* **2022**, *7* (3), 570−579.
(19) Ouyang, Y.; Vandewalle, L. A.; Chen, L.; Plehiers, P. P.; Dobbelaere, M. R.; Heynderickx, G. J.; Marin, G. B.; Van Geem, K. M. Speeding up Turbulent Reactive Flow Simulation via a Deep Artificial Neural Network: A Methodology Study. *Chem. Eng. J.* **2022**, *429*, No. 132442.
(20) Plehiers, P. P.; Symoens, S. H.; Amghizar, I.; Marin, G. B.; Stevens, C. V.; Van Geem, K. M. Artificial Intelligence in Steam Cracking Modeling: A Deep Learning Algorithm for Detailed Effluent Prediction. *Engineering* **2019**, *5* (6), 1027−1040.
(21) Han, X.; Jia, M.; Chang, Y.; Li, Y. An Improved Approach towards More Robust Deep Learning Models for Chemical Kinetics. *Combust. Flame* **2022**, *238*, No. 111934.
(22) Chi, C.; Janiga, G.; Thévenin, D. On-the-Fly Artificial Neural Network for Chemical Kinetics in Direct Numerical Simulations of Premixed Combustion. *Combust. Flame* **2021**, *226*, 467−477.
(23) Muela, J.; Borrell, R.; Ventosa-Molina, J.; Jofre, L.; Lehmkuhl, O.; Pérez-Segarra, C. D. A Dynamic Load Balancing Method for the Evaluation of Chemical Reaction Rates in Parallel Combustion Simulations. *Comput. Fluids* **2019**, *190*, 308−321.
(24) Tekgül, B.; Peltonen, P.; Kahila, H.; Kaario, O.; Vuorinen, V. DLBFoam: An Open-Source Dynamic Load Balancing Model for Fast Reacting Flow Simulations in OpenFOAM. *Comput. Phys. Commun.* **2021**, *267*, No. 108073.
(25) Rettenmaier, D.; Deising, D.; Ouedraogo, Y.; Gjonaj, E.; De Gersem, H.; Bothe, D.; Tropea, C.; Marschall, H. Load Balanced 2D and 3D Adaptive Mesh Refinement in OpenFOAM. *SoftwareX* **2019**, *10*, No. 100317.
(26) Fattebert, J. L.; Richards, D. F.; Glosli, J. N. Dynamic Load Balancing Algorithm for Molecular Dynamics Based on Voronoi Cells Domain Decompositions. *Comput. Phys. Commun.* **2012**, *183* (12), 2608−2615.
(27) Maestri, M.; Cuoci, A. Coupling CFD with Detailed Microkinetic Modeling in Heterogeneous Catalysis. *Chem. Eng. Sci.* **2013**, *96*, 106−117.
(28) Wang, X.; Guo, L.; Ge, W.; Tang, D.; Ma, J.; Yang, Z.; Li, J. Parallel Implementation of Macro-Scale Pseudo-Particle Simulation for Particle−Fluid Systems. *Comput. Chem. Eng.* **2005**, *29* (7), 1543−1553.
(29) Ma, J.; Deng, X.; Hsiao, C.-T.; Chahine, G. L. Hybrid Message-Passing Interface-Open Multiprocessing Accelerated Euler−Lagrange Simulations of Microbubble Enhanced HIFU for Tumor Ablation. *J. Biomech. Eng.* **2023**, *145* (7), No. 071005.
(30) Fernengel, J.; Bolton, L.; Hinrichsen, O. Characterisation and Design of Single Pellet String Reactors Using Numerical Simulation. *Chem. Eng. J.* **2019**, *373*, 1397−1408.
(31) Wehinger, G. D.; Kreitz, B.; Goldsmith, C. F. Non-Idealities in Lab-Scale Kinetic Testing: A Theoretical Study of a Modular Temkin Reactor. *Catalysts* **2022**, *12* (3), 349.
(32) Delgado, K. H. *Surface Reaction Kinetics for Oxidation and Reforming of H2, CO, CH4 over Nickel-Based Catalysts, Karlsruher Institut für Technologie (KIT)*, 2014.
(33) Weller, H. G.; Tabor, G.; Jasak, H.; Fureby, C. A Tensorial Approach to Computational Continuum Mechanics Using Object-Oriented Techniques. *Comput. Phys.* **1998**, *12* (6), 620.
(34) Micale, D.; Uglietti, R.; Bracconi, M.; Maestri, M. Coupling Euler-Euler and Microkinetic Modeling for the Simulation of Fluidized Bed Reactors: An Application to the Oxidative Coupling of Methane. *Ind. Eng. Chem. Res.* **2021**, *60* (18), 6687−6697.
(35) Johnson, P. C.; Jackson, R. Frictional-Collisional Constitutive Relations for Granular Materials with Application to Plane Shearing. *J. Fluid Mech.* **1987**, *176*, 67−93.
(36) Amdahl, G. M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In *Proceedings of the April 18−20, 1967, spring joint computer conference*; 1967; pp 483−485.
(37) Nazari, R.; Ansari, A.; Herrmann, M.; Adrian, R. J.; Kirian, R. A. Numerical and Experimental Investigation of Gas Flow Field Variations in Three-Dimensional Printed Gas-Dynamic Virtual Nozzles. *Front. Mech. Eng.* **2023**, *8*, No. 958963, DOI: 10.3389/fmech.2022.958963.
(38) Wang, Y. X.; Zhang, L. L.; Liu, W.; Cheng, X. H.; Zhuang, Y.; Chronopoulos, A. T. Performance Optimizations for Scalable CFD Applications on Hybrid CPU+MIC Heterogeneous Computing System with Millions of Cores. *Comput. Fluids* **2018**, *173*, 226−236.
(39) Fialko, S. Y. Iterative Methods for Solving Large-Scale Problems of Structural Mechanics Using Multi-Core Computers. *Arch. Civ. Mech. Eng.* **2014**, *14* (1), 190−203.

https://doi.org/10.1021/acsengineeringau.3c00066
ACS Eng. Au 2024, 4, 312−324