applied sciences

MDPI

Article

A Heterogeneous Parallel Algorithm for Euler-Lagrange Simulations of Liquid in Supersonic Flow

Xu Liu, Mingbo Sun *, Hongbo Wang *, Peibo Li, Chao Wang, Guoyan Zhao, Yixin Yang and Dapeng Xiong

Hypersonic Technology Laboratory, National University of Defense Technology, Changsha 410073, China; liuxu19a@nudt.edu.cn (X.L.); lipeibo09@nudt.edu.cn (P.L.); wangchao19@nudt.edu.cn (C.W.); zhaoguoyan09@nudt.edu.cn (G.Z.); yangyixin@nudt.edu.cn (Y.Y.); xiongdapeng18@nudt.edu.cn (D.X.)

* Correspondence: sunmingbo@nudt.edu.cn (M.S.); whbwatch@nudt.edu.cn (H.W.)

Abstract: In spite of its prevalent usage for simulating the full-field process of the two-phase flow, the Euler–Lagrange method suffers from a heavy computing burden. Graphics processing units (GPUs), with their massively parallel architecture and high floating-point performance, provide new possibilities for high-efficiency simulation of liquid-jet-related systems. In this paper, a central processing unit/graphics processing unit (CPU/GPU) parallel algorithm based on the Euler–Lagrange scheme is established, in which both the gas and liquid phase are executed on the GPUs. To realize parallel tracking of the Lagrange droplets, a droplet dynamic management method is proposed, and a droplet-locating method is developed to address the cell. Liquid-jet-related simulations are performed on one core of the CPU with a GPU. The numerical results are consistent with the experiment. Compared with a setup using 32 cores of CPUs, considerable speedup is obtained, which is as high as 32.7 though it decreases to 20.2 with increasing droplets.

Keywords: GPU acceleration; parallel computing; droplet dynamic management; CPU/GPU hybrid computation

check for updates

Citation: Liu, X.; Sun, M.; Wang, H.; Li, P.; Wang, C.; Zhao, G.; Yang, Y.; Xiong, D. A Heterogeneous Parallel Algorithm for Euler-Lagrange Simulations of Liquid in Supersonic Flow. *Appl. Sci.* **2023**, *13*, 11202. https://doi.org/10.3390/app132011202

Academic Editor: Francesca Scargiali

Received: 22 September 2023
Revised: 8 October 2023
Accepted: 10 October 2023
Published: 12 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The scramjet engine is a hot topic in the research field of hypersonic vehicles [1–5]. Compared with gaseous fuel, liquid fuel has advantageous characteristics of high volumetric energy density, easy handling and pumping, convenient transportation, and a high degree of safety [6]. Thus, the applications of liquid fuel in scramjet engines are drawing increasing attention. Liquid fuel atomization and mixing have significant impact on the performance of scramjet engines. Crossflow jet is one way to improve the atomization and mixing characteristic [7–9]. However, such a process is highly complex. Under the action of the supersonic transverse airflow, the liquid fuel will go through multiple sub-processes such as fragmentation, atomization, mixing, etc. [10]. Therefore, a liquid jet in supersonic flow is a temporally and spatially multi-scale problem, which requires tremendous amounts of computational resources to solve numerically [11].

Numerous research has been conducted to investigate the spray structure [12–14], atomization properties [15,16], and spray distribution [12–14] of the two-phase flow. Yoo et al. [17] employed a computation domain with 2.79 × 10^6 grid points, divided into 132 blocks, to study the breakup and atomization of a water jet in a subsonic crossflow at different liquid–gas momentum flux ratios and cross-flow temperatures. Dai et al. used direct numerical simulation via the Euler–Lagrange point-source approach to investigate the turbulent modulation of particles in compressible isotropic turbulence [18], the particle dispersion in a three-dimensional spatially developing compressible mixing layer [19]. To capture the small-scale structure, each case was computationally intensive. The experiments of Lin et al. [20] and Wu et al. [21] were recreated numerically by Li et al. [13,22] using their in-house code to investigate the gas–liquid interaction and elucidate the mixing process.

The number of cells in both cases was 19.4 million, while the number of computational droplets was about 1 million and 0.55 million, respectively. Thereafter, an investigation of a liquid jet in a cavity-based supersonic combustor was carried out both experimentally [23] and numerically [14]. The numerical simulation with 26.46 million grids and 5 million droplets was performed on 756 central processing unit (CPU) cores of the Tianhe-1 for 10 flow-throughs, running for approximately 31 days. Recently, Li et al. [24] added an evaporation model to further study the spray characteristics of kerosene jet in the cavity-based supersonic combustor. In the study, the number of cells and droplets was 18 million and 0.115 million, respectively. The simulation was also implemented on 600 CPU cores of the Tianhe-1 for 10 flow-throughs, running for 36 days. In conclusion, most of the simulations of jet in a crossflow were conducted using CPU-based high-performance computing, employing enormous numbers of cores and very long computational time. Therefore, a fast, efficient, and accurate computational method to simulate a liquid jet in supersonic flow is highly sought after.

With the overall hardware performance and knowledge base increasing, GPU has become more prevalent for computational acceleration in recent years [25]. Fast computing speed, easy maintenance, and high-power efficiency make using graphics processing units (GPUs) for scientific computing an attractive option in many fields [26,27]. Liu et al. [28] proposed an improved hybrid Euler–Lagrange method for solving the Navier Stokes equation via GPU and performed a simulation of lid-driven cavity problem with 8 million grids. Speedups of 7, 14, 21, and 27 were obtained, respectively, for one to four GPUs, with respect to two Intel Xeon E5-2630 v2 CPUs (Intel Corporation, Palo Alto, CA, USA) with 12 cores. Salvadore et al. [29] ported and optimized a Fortran-based code and performed an accurate simulation of turbulent flow on a NVIDIA Tesla S2070 (NVIDIA Corporation, Santa Clara, CA, USA). The acceleration is about 22 times faster than one AMD Opteron 2352 Barcelona chip (AMD, Sunnyvale, CA, USA) and about 11 times faster than one Intel Xeon X5650 Westmere core (Intel Corporation, Palo Alto, CA, USA). Schalkwijk et al. [30] ported existing code to multiple GPUs and achieved massive acceleration for the large eddy simulation (LES) of the planetary boundary layer. Lai et al. [31] proposed a heterogeneous parallel computational fluid dynamics (CFD) solver and conducted a multi-parallel computation.

The application of GPU acceleration in the numerical simulation of multiphase flow has also been studied. Sweet et al. [32] used four NVIDIA GPU devices to accelerate an existing particle-filled turbulence simulation code based on message passing interface (MPI). For 10^7 particles, calculation using GPU can achieve about 14 speedups compared to the original CPU execution. However, the method is one-way coupled, and the gas phase information needs to be transmitted to the devices for each iterative step, rendering the method impractical for large-scale simulations. Ge et al. [33] developed a CPU/GPU portable software library Grit and combined it with a direct numerical simulation solver to simulate the particle-laden turbulent flow and the dilute turbulent jet evaporation. Xu et al. [34] used more than 200 GPUs to achieve a quasi-real-time simulation of an industrial scale rotating drum. Xu et al. [35] developed a CPU-GPU mixed computing mode, in which the gas phase was solved by CPUs and the particle was calculated by GPUs, and they applied this method to three cases with 2500, 5000, 1000 cells, and 32,000, 64,000, 128,000 parcels. Compared with the CPU method, respective speedups of 1.05, 1.44, and 2.37 were achieved. Ikebata et al. [36] developed a GPU-accelerated model to accomplish the actual simulation of the entire toilet bowl system. At present, GPU acceleration technology has been widely used in multiphase flow, but mostly for certain expensive and parallelizable sections. To the authors' knowledge, the GPU technology is not applied to accelerate both the gas and liquid phase simultaneously.

The current work is executed as a portion of the establishment of a universal CFD code for fast simulation of the full-field process of liquid in supersonic flow. To achieve it, a CPU/GPU heterogeneous parallel algorithm is built, in which a novel method for droplet dynamic management is proposed to solve the droplets' fluctuation in the flow field, and a droplet-locating approach is developed to conform with the weakened three-dimensional relationships of the cells. The purpose of this paper is to evaluate the internal two-phase CFD code, demonstrate the successful design of a GPU-based computing system, and compare the acceleration ratios compared to the CPUs. Although the two-phase CPU/GPU heterogeneous parallel computation mode based on the Euler–Lagrange approach proposed in this paper is used for liquid in supersonic flow, it is also applicable to other Euler–Lagrange simulations.

This paper is organized as follows. The relevant physical models for the gas phase, liquid phase, phase-to-phase exchange terms, and code structure are introduced in Section 2. The computational condition, physical parameters, and performance of the algorithm are presented in Section 3, where we also present the key findings of this study. Conclusions are drawn in Section 4.

2. Algorithm and Code Structures

In this paper, a CPU/GPU heterogeneous parallel algorithm for liquid in supersonic flow is established. The Euler–Lagrange scheme, which is widely used in two-phase full-field flow, is adopted in this algorithm, wherein the gas phase and the liquid phase are solved by the Euler approach and the Lagrange approach, respectively. The interaction between the gas phase and the liquid phase is computed by a two-way coupling method.

2.1. Gas-Phase Governing Equations

The governing equations based on the integral form for the gas-phase can be written as follows [31,37,38]:

$$\frac{\partial}{\partial t}\int_{\Omega} \boldsymbol{W} d\Omega + \oint_{\partial\Omega} (\boldsymbol{F}_{\mathrm{c}} - \boldsymbol{F}_{\mathrm{v}}) dS = \int_{\Omega} \boldsymbol{Q} d\Omega, \tag{1}$$

$$\boldsymbol{W} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \boldsymbol{F}_{\mathrm{c}} = \begin{pmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ V(\rho E + p) \end{pmatrix}, \boldsymbol{F}_{\mathrm{v}} = \begin{pmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\tau_{zz} \\ n_x\Theta_x + n_y\Theta_y + n_z\Theta_z \end{pmatrix}, \boldsymbol{Q} = \begin{pmatrix} \dot{Q}_{\mathrm{m}} \\ \dot{Q}_{\mathrm{p},x} \\ \dot{Q}_{\mathrm{p},y} \\ \dot{Q}_{\mathrm{p},z} \\ \dot{Q}_{\mathrm{E}} \end{pmatrix}, \tag{2}$$

where

$$V = n_x u + n_y v + n_z w, \tag{3}$$

$$\begin{aligned} \Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k\tfrac{\partial T}{\partial x}, \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + k\tfrac{\partial T}{\partial y}, \\ \Theta_z &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + k\tfrac{\partial T}{\partial z}. \end{aligned} \tag{4}$$

Here $\boldsymbol{W}$, $\boldsymbol{F}_{\mathrm{c}}$, $\boldsymbol{F}_{\mathrm{v}}$ are the vector of conservative variables, convective fluxes, and viscous fluxes, respectively, Ω is a control volume, and V is defined as the scalar product of the velocity vector and the unit normal vector. $\dot{Q}_{\mathrm{m}}$ is the mass flow rate, $\dot{Q}_{\mathrm{p},x}$, $\dot{Q}_{\mathrm{p},y}$, $\dot{Q}_{\mathrm{p},z}$ are the momentum in the x, y, and z directions, respectively, and $\dot{Q}_{\mathrm{E}}$ is the energy source terms of the droplet donation, and the functions will be introduced in Section 2.2.

For the vector of viscous fluxes, the components of viscous stress τ_{ij} are as follows [37]:

$$\tau_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}). \tag{5}$$

The spatial discretization of Equation (1) is based on the cell-centered finite volume method. The upwind Advection upstream splitting method + UP (AUSM + UP) and central scheme are used to solve the convective fluxes and the viscous fluxes, respectively.

A Monotone upstream-centered scheme for conservation laws (MUSCL) reconstruction scheme is used, and the van Leer limiter is employed to make the scheme monotone. The $k-\omega$ shear stress transport (SST) turbulence model is adopted for the turbulent model.

A liquid in supersonic flow is a typical unsteady compressible flow problem. A dual time-scheme based on Data Parallel Lower-Upper Relaxation Method (DP-LUR) is used as follows.

Equation (1) is discretized spatially in the *i*th cell to yield.

$$\Omega_i \frac{\partial \boldsymbol{W}_i}{\partial t} + \sum_{j=1}^{N} (\widetilde{\boldsymbol{F}}_{\mathrm{c},ij} - \widetilde{\boldsymbol{F}}_{\mathrm{v},ij}) S_{ij} = Q_i, \tag{6}$$

where N is the total number of neighboring cells that share a face with cell i, S_{ij} is the area of the face shared by cell i and cell j. $\widetilde{\boldsymbol{F}}_{\mathrm{c}}$ and $\widetilde{\boldsymbol{F}}_{\mathrm{v}}$ are the second-order-accurate numerical inviscid and viscous flux vectors at the face, respectively. The time derivative is discretized with a 3-point backward-difference dispersion in Equation (6), and a pseudo time derivative term is added to the left, resulting in

$$(\frac{3\Omega_i}{2\Delta t} + \frac{\Omega_i}{\Delta \tau})\Delta \boldsymbol{W}_i^m + \Omega_i \frac{3\boldsymbol{W}_i^m - 4\boldsymbol{W}_i^n + \boldsymbol{W}_i^{n-1}}{2\Delta t} + \sum_{j=1}^{N} (\Delta \widetilde{\boldsymbol{F}}_{\mathrm{c},ij}{}^m - \Delta \widetilde{\boldsymbol{F}}_{\mathrm{v},ij}^m) S_{ij} = Res_i^m, \tag{7}$$

where the superscript m represents the mth pseudo time. Δ is the forward difference in time, and the right-hand side residual can be written as

$$Res_i^m = -\sum_{j=1}^{N} (\widetilde{\boldsymbol{F}}_{\mathrm{c},ij}^m - \widetilde{\boldsymbol{F}}_{\mathrm{v},ij}^m) + Q_i^m. \tag{8}$$

Note that

$$\begin{aligned}
&\Delta \widetilde{\boldsymbol{F}}_{\mathrm{c},ij}^m - \Delta \widetilde{\boldsymbol{F}}_{\mathrm{v},ij}^m = [\widetilde{\boldsymbol{F}}_{\mathrm{c}}(\boldsymbol{W}_i^{m+1}, \boldsymbol{W}_j^{m+1}) - \widetilde{\boldsymbol{F}}_{\mathrm{c}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^{m+1})] \\
&+[\widetilde{\boldsymbol{F}}_{\mathrm{c}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^{m+1}) - \widetilde{\boldsymbol{F}}_{\mathrm{c}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^m)] - [\widetilde{\boldsymbol{F}}_{\mathrm{v}}(\boldsymbol{W}_i^{m+1}, \boldsymbol{W}_j^{m+1}) - \boldsymbol{F}_{\mathrm{v}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^{m+1})] \\
&-[\widetilde{\boldsymbol{F}}_{\mathrm{v}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^{m+1}) - \widetilde{\boldsymbol{F}}_{\mathrm{v}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^m)].
\end{aligned} \tag{9}$$

Taylor's expansion is performed on the first and the third terms on the right-hand side of Equation (9).

$$\widetilde{\boldsymbol{F}}_{\mathrm{c}}(\boldsymbol{W}_i^{m+1}, \boldsymbol{W}_j^{m+1}) - \widetilde{\boldsymbol{F}}_{\mathrm{c}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^{m+1}) \approx \frac{\partial \widetilde{\boldsymbol{F}}_{\mathrm{c},ij}}{\partial W_i} \Delta \boldsymbol{W}_i^m, \tag{10}$$

$$\widetilde{\boldsymbol{F}}_{\mathrm{v}}(\boldsymbol{W}_i^{m+1}, \boldsymbol{W}_j^{m+1}) - \widetilde{\boldsymbol{F}}_{\mathrm{v}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^{m+1}) \approx \frac{\partial \widetilde{\boldsymbol{F}}_{\mathrm{v},ij}}{\partial \boldsymbol{W}_i} \Delta \boldsymbol{W}_i^m. \tag{11}$$

Substituting Equations (9)–(11) back to Equation (7), gives

$$\begin{aligned}
&(\tfrac{3\Omega_i}{2\Delta t} + \tfrac{\Omega_i}{\Delta \tau})\Delta \boldsymbol{W}_i^m + \Omega_i \tfrac{3\boldsymbol{W}_i^m - 4\boldsymbol{W}_i^n + \boldsymbol{W}_i^{n-1}}{2\Delta t} + \textstyle\sum_{j=1}^{N} (\tfrac{\partial \widetilde{\boldsymbol{F}}_{\mathrm{c},ij}}{\partial \boldsymbol{W}_i} - \tfrac{\partial \widetilde{\boldsymbol{F}}_{\mathrm{v},ij}}{\partial \boldsymbol{W}_i}) S_{ij} \\
&+\textstyle\sum_{j=1}^{N} [\widetilde{\boldsymbol{F}}_{\mathrm{c}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^m + \Delta \boldsymbol{W}_j^m) - \widetilde{\boldsymbol{F}}_{\mathrm{c}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^m)] S_{ij} \\
&-\textstyle\sum_{j=1}^{N} [\widetilde{\boldsymbol{F}}_{\mathrm{v}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^m + \Delta \boldsymbol{W}_j^m) - \widetilde{\boldsymbol{F}}_{\mathrm{v}}(\boldsymbol{W}_i^m, \boldsymbol{W}_j^m)] S_{ij} = Res_i^m.
\end{aligned} \tag{12}$$

The numerical flux vectors are discretized as

$$\widetilde{\boldsymbol{F}}_{\mathrm{c},ij} - \widetilde{\boldsymbol{F}}_{\mathrm{v},ij} \approx \frac{1}{2} [\boldsymbol{F}_{\mathrm{c},i} + \boldsymbol{F}_{\mathrm{c},j} - \lambda_{ij} (\boldsymbol{W}_j - \boldsymbol{W}_i)], \tag{13}$$

where λ_{ij} is the spectral radius of the flux Jacobian matrix, written as follows:

$$\lambda_{ij} = \left|\boldsymbol{V}\cdot\boldsymbol{n}_{ij}\right| + a + (\frac{\mu_{ij}}{pr} + \frac{\mu_{t,ij}}{pr_t})\frac{r_{ij}}{\rho_{ij}\left\|\boldsymbol{n}_{ij}\cdot(\boldsymbol{r}_j - \boldsymbol{r}_i)\right\|}, \tag{14}$$

$\boldsymbol{n}_{ij}$ is the normal vector of the shared face, $\boldsymbol{r}_i$, $\boldsymbol{r}_j$ represent the position vectors of cell i and cell j, a is the speed of sound, μ_{ij}, $\mu_{t,ij}$ are the average of kinematic and turbulent viscosities of the cell i and cell j. For closed-cell i

$$\sum_{j=1}^{N}\frac{\partial\boldsymbol{F}_i(n_{ij})}{\partial W_i}S_{ij} = 0. \tag{15}$$

Therefore,

$$\begin{aligned}(\tfrac{3\Omega_i}{2\Delta t} + \tfrac{\Omega_i}{\Delta\tau} + \tfrac{1}{2}\sum_{j=1}^{N}\lambda_{ij}S_{ij})\Delta\boldsymbol{W}_i^m &= Res_i^m - \Omega_i\tfrac{3\boldsymbol{W}_i^m - 4\boldsymbol{W}_i^n + \boldsymbol{W}_i^{n-1}}{2\Delta t} \\ -\tfrac{1}{2}\sum_{j=1}^{N}[\boldsymbol{F}(\boldsymbol{W}_j^m + \Delta\boldsymbol{W}_j^m) - \boldsymbol{F}(\boldsymbol{W}_j^m) - \lambda_{ij}\Delta\boldsymbol{W}_j^m]S_{ij}.\end{aligned} \tag{16}$$

The Lower–upper symmetric Gauss–Seidel (LU-SGS) algorithm employs a series of corner-to-corner sweeps through the flow field using the latest available data for the off-diagonal terms to solve Equation (16). However, this method is data-dependent and challenging to implement in parallel computing. To make the method parallelize effectively, we use the DP-LUR approach instead, whose main feature is replacing the Gauss–Seidel sweeps with a series of point-wise relaxation steps.

First, by ignoring the second and third terms, we obtain

$$\Delta\boldsymbol{W}_i{}^0 = (\frac{3\Omega_i}{2\Delta t} + \frac{\Omega_i}{\Delta\tau} + \frac{1}{2}\sum_{j=1}^{N}\lambda_{ij}S_{ij})^{-1}Res_i^m. \tag{17}$$

Then, a series of $k_{\max}$ relaxation steps are made with k = 1,. . ., $k_{\max}$

$$\begin{aligned}\Delta\boldsymbol{W}_i{}^{k_{\max}} &= (\tfrac{3\Omega_i}{2\Delta t} + \tfrac{\Omega_i}{\Delta\tau} + \tfrac{1}{2}\sum_{j=1}^{N}\lambda_{ij}S_{ij})^{-1}\{Res_i^m - \Omega_i\tfrac{3\boldsymbol{W}_i^m - 4\boldsymbol{W}_i^n + \boldsymbol{W}_i^{n-1}}{2\Delta t} \\ -\tfrac{1}{2}\sum_{j=1}^{N}[\boldsymbol{F}(\boldsymbol{W}_j^m + \Delta\boldsymbol{W}_j^m) - \boldsymbol{F}(\boldsymbol{W}_j^m) - \lambda_{ij}\Delta\boldsymbol{W}_j^m]S_{ij}\}.\end{aligned} \tag{18}$$

Then

$$\Delta\boldsymbol{W}_i = \Delta\boldsymbol{W}_i{}^{k_{\max}}. \tag{19}$$

Here, $k_{\max}$ = 4 was used [39].

2.2. Liquid-Phase Equations

The Lagrange approach, which is an efficient method that supports parallel computing for full-field simulation is used to model the spray dynamics. In addition, the computational droplet is used in this paper. One computational droplet represents a collection of real droplets with the same properties (diameter, velocity, temperature, etc.) and in the same location [40]. The number of real droplets carried by each computational droplet is represented by a parameter ω_k. Since the density of the liquid phase is much larger than that of the gas phase, it is possible to ignore the unsteady drag forces in the Basset–Boussinesq–Oseen equation (BBO equation), and only the Stokes drag force is included in the liquid momentum equation. Because of the small liquid volume fraction, the gas–liquid flow can be treated as dilute two-phase flow. As a result, the collision and coalescence among droplets are also ignored [13].

Newton's second law determines the motion of each computational droplet as follows:

$$\frac{dx_{k,i}}{dt} = u_{k,i}, \tag{20}$$

$$\frac{du_{k,i}}{dt} = \frac{F_{k,i}}{m_k}, \tag{21}$$

where $x_{k,i}$, m_k, $u_{k,i}$ are the position vector, mass, and velocity of the droplet, respectively, $i = x, y, z$. The surface drag $F_{k,i}$, which is a force acting on the droplet body, can be modeled by:

$$\frac{F_{k,i}}{m_k} = \frac{3}{4}\frac{\rho_{\mathrm{g}} C_k}{\rho_{\mathrm{d}} d_k}\left|u_{\mathrm{g},i} - u_{k,i}\right|\left(u_{\mathrm{g},i} - u_{k,i}\right), \tag{22}$$

where d_k, ρ_{d}, u_k represent the droplet diameter, density, and velocity, the ρ_{g}, u_{g} are the gas-phase density and velocity, which are calculated by weighting the inverse distance between the control volume containing the droplet and its adjacent control volumes. C_k is the drag coefficient, which can be calculated by $C_{ks} = f_{kf} C_k(M_k, \mathrm{Re}_k)$. M_k and f_{kf} are the relative Mach number and the droplet deformation correction factor, respectively. The calculation of the modified drag coefficient C_{ks} is detailed in reference [22].

The effects of the droplet on the gas phase can be calculated as follows:

$$\dot{Q}_{\mathrm{m}} = -\frac{\omega_k}{\Omega}\sum_k \frac{dm_k}{dt}, \tag{23}$$

$$\dot{Q}_{\mathrm{p},x} = -\frac{\omega_k}{\Omega}\sum_k \frac{dm_k u_k}{dt}, \tag{24}$$

$$\dot{Q}_{\mathrm{p},y} = -\frac{\omega_k}{\Omega}\sum_k \frac{dm_k v_k}{dt}, \tag{25}$$

$$\dot{Q}_{\mathrm{p},z} = -\frac{\omega_k}{\Omega}\sum_k \frac{dm_k w_k}{dt}, \tag{26}$$

$$\dot{Q}_{\mathrm{E}} = -\frac{\omega_k}{\Omega}\sum_k \frac{d\left(m_k C_{pl} T_k + m_k \boldsymbol{U}_{k,i}^2\right)}{dt}, \tag{27}$$

where, m_k, T_k, are the mass and temperature of the kth droplet respectively, u_k, v_k, w_k are the velocities in x, y, and z directions of the kth droplet, C_{pl} is the specific heat of the liquid phase. Ω is the volume of the cell containing the droplet. In this paper, evaporation is not considered, therefore,

$$\dot{Q}_{\mathrm{m}} = 0, \tag{28}$$

$$\dot{Q}_E = -\frac{\omega_k}{\Omega}\sum_k \frac{d\left(m_k \boldsymbol{U}_{k,i}^2\right)}{dt}. \tag{29}$$

In the present study, the droplet stripping process is modeled by Kelvin–Helmholtz model (KH model). Then, the droplet deformation and oscillation are modeled by the Taylor Analogy Breakup model (TAB model), and the TAB model and Rayleigh–Taylor model (RT model) compete to simulate the droplet secondary breakup process. It is worth noting that both the TAB and RT models are explosion-type breakup modes, where the original droplets are treated as if they disappeared while smaller droplets are newly produced. More information can be found in reference [22]. The three-step Runge–Kutta method with third-order accuracy is used for the temporal discretization of droplet iteration.

2.3. CPU/GPU Heterogeneous Parallel Algorithm

2.3.1. CPU/GPU Heterogeneous Parallel Architecture

The CPU is suitable for logical and data-dependent operations, while the GPU excels at implementing data-intensive and highly parallel tasks. The workload assignment process for this study is illustrated in Figure 1. The computational operations are divided into branch instruction tasks, single instruction tasks, and data exchange tasks. By taking into consideration their relative efficiency and total computational cost, the three types of tasks are suitably executed on the CPU, GPU, and CPU, respectively. In other words, the pre-processing, data exchange, and post-processing are performed on the CPU, whereas both the gas phase iteration and the droplet phase iteration are executed on the GPU.

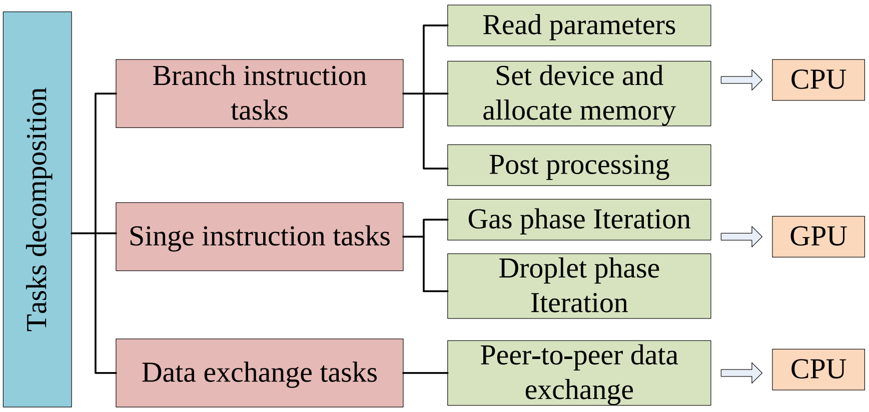

Figure 1. Scheme of tasks decomposition.

The schematic diagram of the CPU/GPU two-phase computing mode is demonstrated in Figure 2. At the macro-scale, the CPU is responsible for general control, while the GPU takes charge of the computation of the gas and droplet iteration. At the micro level, the cells and droplets are stored one-dimensionally, and every cell and droplet is treated as an individual unit that is computed with a thread. Working in tandem, they constitute the CPU/GPU heterogeneous two-phase computation mode.

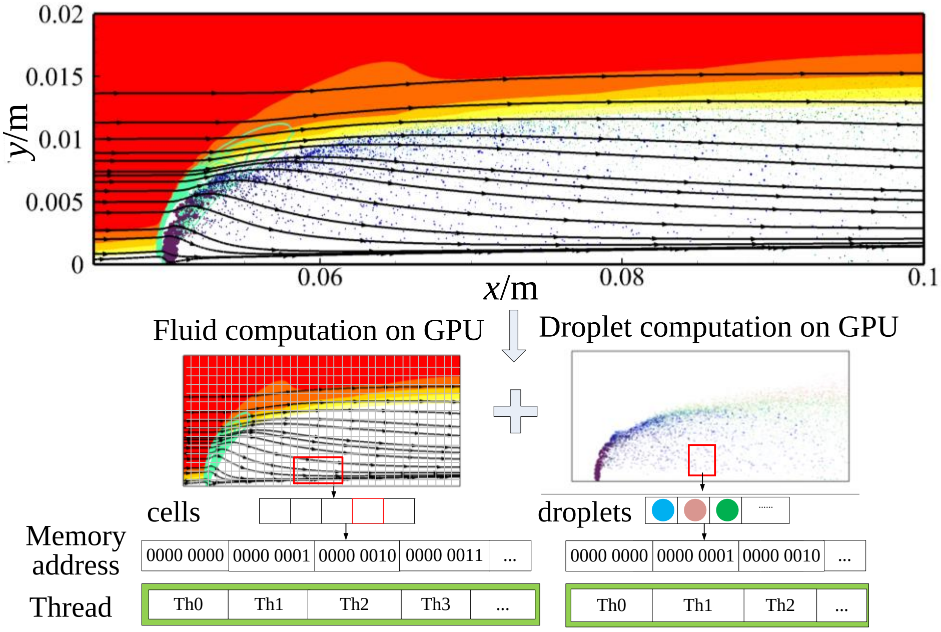

Figure 2. Scheme diagram of the CPU/GPU computation mode.

Compared with the CPU mode, the major distinction of the proposed method is that the sequential tasks and operations in the gas and droplet iteration are performed using the parallel architecture of the GPU cores. Figure 3 illustrates a brief flow diagram of the CPU/GPU mode and the CPU mode.

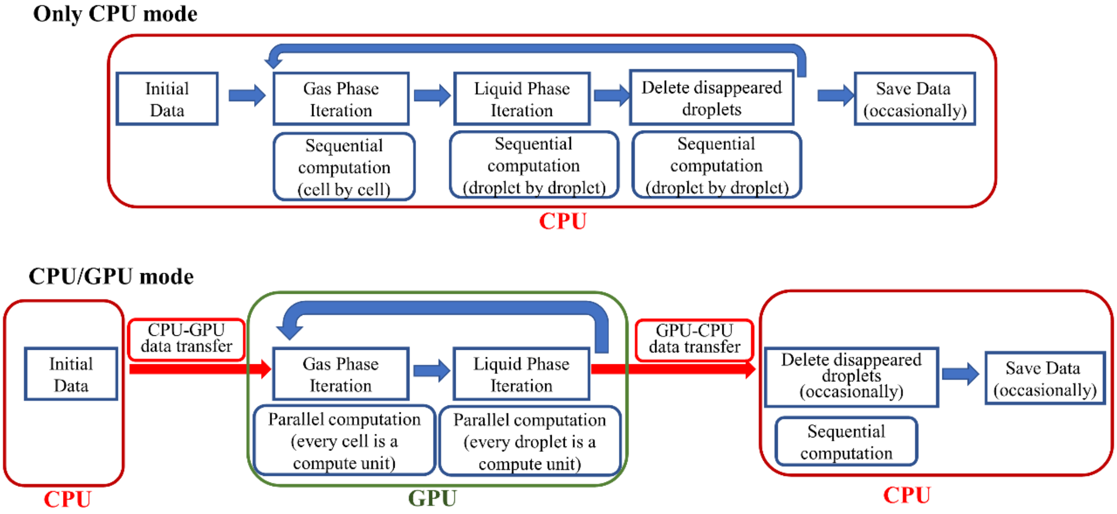
Only CPU mode
Initial Data
Gas Phase Iteration
Liquid Phase Iteration
Delete disappeared droplets
Save Data (occasionally)
Sequential computation (cell by cell)
Sequential computation (droplet by droplet)
Sequential computation (droplet by droplet)
CPU
CPU/GPU mode
Initial Data
CPU-GPU data transfer
Gas Phase Iteration
Liquid Phase Iteration
Parallel computation (every cell is a compute unit)
Parallel computation (every droplet is a compute unit)
GPU-CPU data transfer
Delete disappeared droplets (occasionally)
Save Data (occasionally)
Sequential computation
CPU
GPU
CPU

Figure 3. Simplified flow diagram of the CPU/GPU computation mode and only CPU mode.

The parallel computing using C++ language is based on the Compute Unified Device Architecture (CUDA) model, which contains a massive library of custom functions and structures related to the simulation of multiphase flows. In order to achieve high-speed and large-scale simulation of the gas–liquid supersonic flow, the code is established in several aspects, which we shall detail below.

2.3.2. A Novel Method for Dynamic Management of Droplets

The liquid spray is a multi-scale and multi-physics process. During the injection, atomization, and evaporation processes, the number of droplets fluctuates. Thus, an efficient method to dynamically manage the droplets is required. In this paper, we propose a novel scheme to manage the droplets, illustrated in Figure 4. Firstly, a struct is defined that includes different types of droplet information, such as position, velocity, cell number, etc. Secondly, a global struct type array on GPU and CPU is created whose memory is redundantly allocated, and its value can be estimated by the following formula:

$$\text{Max } memory = \frac{\frac{L}{U_\infty} \cdot Q_\text{d}}{\frac{4}{3}\pi (D_\text{Drop}/2)^3 \cdot \rho_\text{d} \cdot \omega_k}, \tag{30}$$

where L is the length from the droplet injection position to the exit of the computational domain, U_∞ is the velocity of the incoming flow, Q_d is the mass flow rate of the droplet, D_d is the flux average diameter of the droplet, which is 10 µm in this paper, in accordance to the reference [16,20].

When a droplet enters the flow, it is first stored in the array on the GPU and is then involved in the droplet kernel computation. If it undergoes the processes of droplet stripping, droplet breakup due to deformation and oscillation, as well as droplet secondary breakup, the new small droplets will be produced and recorded in a temporary array on GPU and then transferred to the CPU. After the screening, they will be transferred to a temporary array on GPU and then added to the unassigned global array on the GPU. If a droplet undergoes the processes of droplet breakup due to deformation and oscillation, the droplet secondary breakup, or leaving the computational domain at the outflow resulting in the disappearance of the droplet, it will be marked to be deleted later. When the max capacity of the array is reached, the global array on the GPU will be transferred to the

CPU global array to subtract elements by a method based on two pointer method, which is demonstrated in Figure 5. For example, there are seven droplets in the illustrated array, in which the gray elements are the one marked for deletion. Two indices named src and dst are created and both point to the beginning of the array. The scr is responsible for traversing through the elements in the array and checking whether they are marked. If the element points by src is marked, the src will point to the next one. If not, the element will be stored where the dst points, and then both the src and dst point to the next. After seven steps, the element in the end of the array will be stored where the dst points, and the removal process is finished.

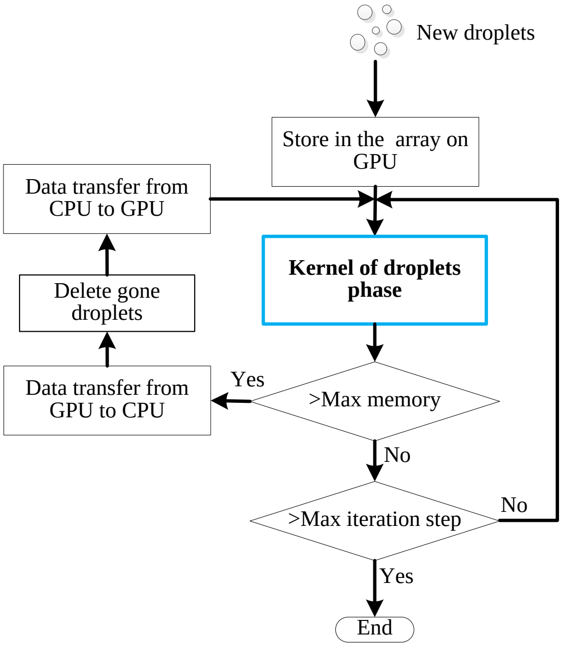

Figure 4. Dynamic management of droplets.

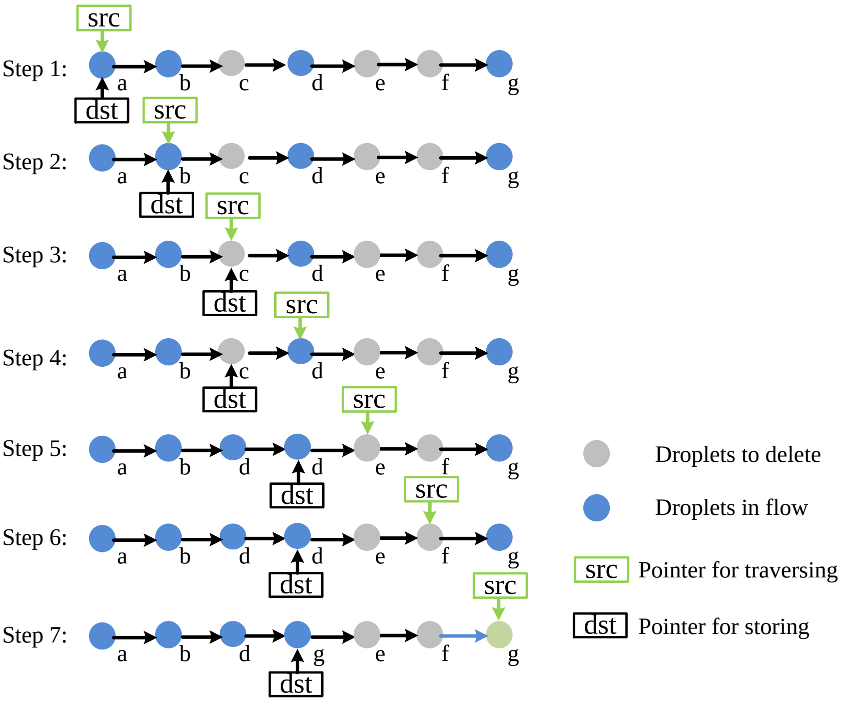

Figure 5. Droplets removal process (the circles are different elements in the array, the gray circle is marked to delete).

2.3.3. A Method for Droplet Location

Of particular note is that, in our CPU/GPU hybrid computation mode, the three-dimensional relationships of the cells are weakened, making it hard to directly locate droplets and find their neighbors. Therefore, the information about which two cells share the *i*th face and which faces enclose the cell are stored to enhance the adjacent relationships of the cells. Then, combine the particle-locating algorithm of Chorda [41] to address the droplets. The process is illustrated in Figure 6. As mentioned above, the cell Ω_a has the information of its enclosed faces, while the face has the information of which two cells share it. Figure 6a shows the criterion for determining whether the droplet is within the given cell. The determination criterion can be described as:

$$Scalar_i = \overrightarrow{\Omega \mathrm{F}_i} \cdot \overrightarrow{\mathrm{F}_i \mathrm{d}}, \tag{31}$$

$$Inoroutface_i = \begin{cases} 0, Scalar_i \le 0 \\ 1, Scalar_i > 0, \end{cases} \tag{32}$$

$$Inoroutcell = \sum_{i}^{N_F} Inoroutface_i. \tag{33}$$

where the $\overrightarrow{\Omega \mathrm{F}_i}$ is the vector of the cell-center pointing to the face-center, $\overrightarrow{\mathrm{F}_i \mathrm{d}}$ is the vector of the face-center pointing to the droplet. $Inoroutface_i > 0$ indicates the droplet is out of the *i*th face, while $Inoroutface_i = 0$ indicates the droplet is in the face. N_F is the total number of cell faces. Thus, when $Inoroutcell = 0$, the droplet is in the given cell. The initial cell containing the droplet is calculated by traversing the entire cells.

Limited by the weakened three-dimensional relationships, a search method for the droplet is developed by traveling through the cells by droplet trajectory, which joins the initial and final droplet position. Figure 6b gives an example of one path. The key point is to choose which face of the current cell the droplet exits from and then use the face to locate the neighboring cell. The search process is repeated until the cell is addressed.

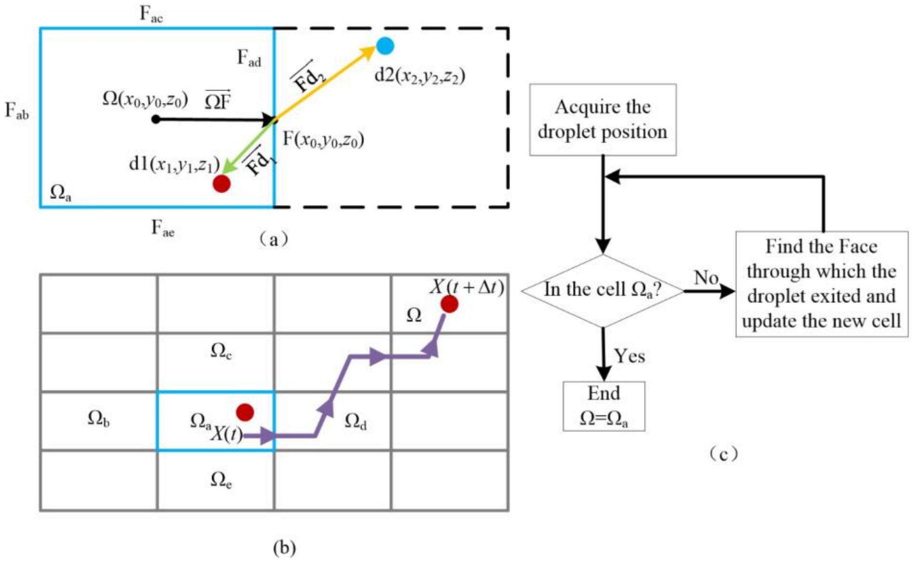

Figure 6. Droplets locating process ((**a**) Judgement criterion; (**b**) Example of a typical path; (**c**) Flow diagram of droplet locating).

Figure 7 summarizes the general two-phase computational procedure, the solid boxes summarize the overall processes on the CPU and GPU, and the dashed box represents the specific solution processes of the two-phase governing equations, which are implemented on the GPU. Of particular note is that, in the event of droplet breakup, a temporary array on GPU and CPU is used to store and transfer the newly produced droplets to the global array on GPU to lessen data transfer. This is also the reason why the droplet breakup takes the most proportion of the time cost.

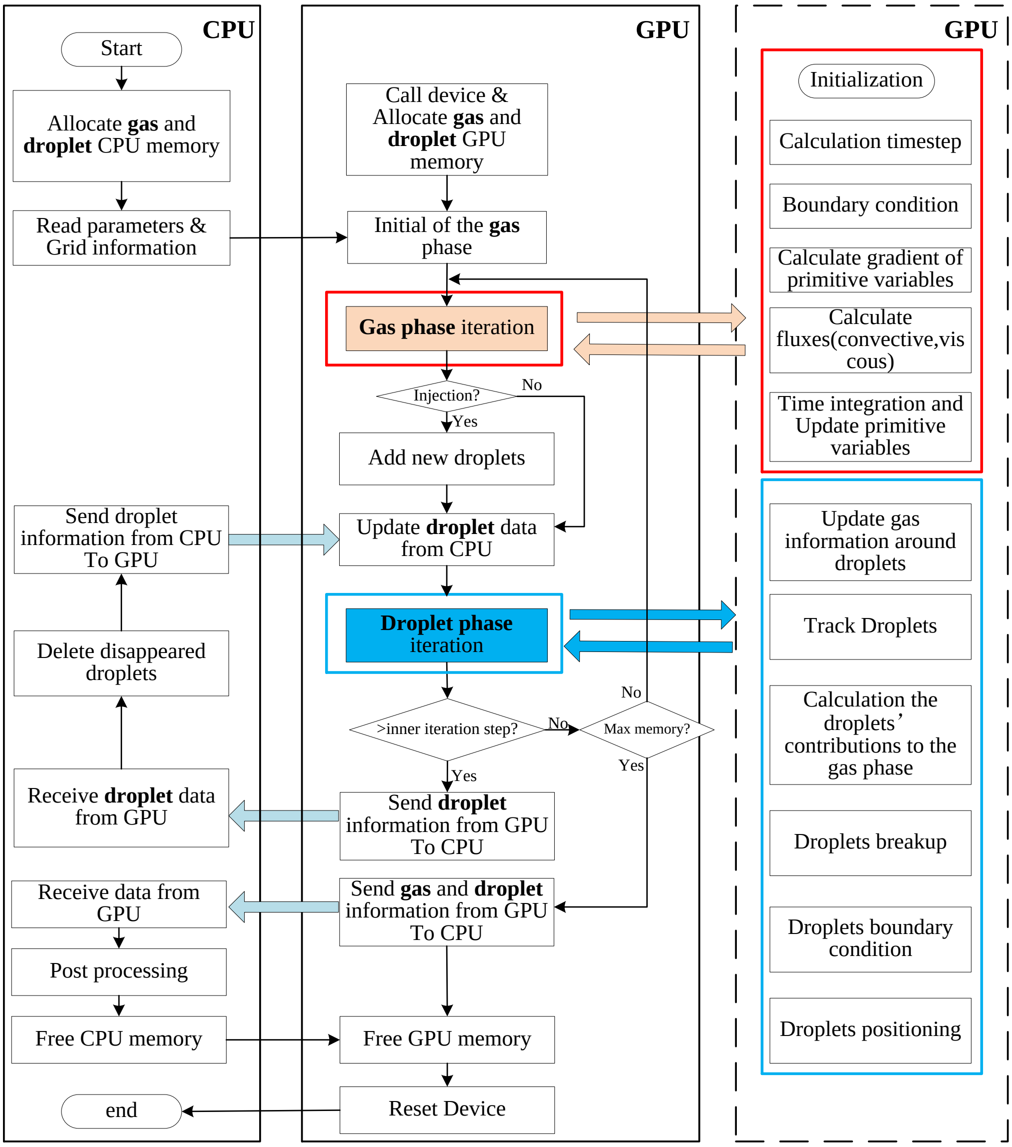
CPU
Start
Allocate gas and droplet CPU memory
Read parameters & Grid information
Send droplet information from CPU To GPU
Delete disappeared droplets
Receive droplet data from GPU
Receive data from GPU
Post processing
Free CPU memory
end
GPU
Call device & Allocate gas and droplet GPU memory
Initial of the gas phase
Gas phase iteration
Injection?
No
Yes
Add new droplets
Update droplet data from CPU
Droplet phase iteration
>inner iteration step?
No
Max memory?
No
Yes
Yes
Send droplet information from GPU To CPU
Send gas and droplet information from GPU To CPU
Free GPU memory
Reset Device
GPU
Initialization
Calculation timestep
Boundary condition
Calculate gradient of primitive variables
Calculate fluxes(convective,vis cous)
Time integration and Update primitive variables
Update gas information around droplets
Track Droplets
Calculation the droplets' contributions to the gas phase
Droplets breakup
Droplets boundary condition
Droplets positioning

Figure 7. Code structure of the CPU/GPU two-phase computational mode.

3. Results and Discussion

3.1. Code Validation

Simulations of a liquid jet in supersonic flow are performed to numerically replicate the experiments of Lin et al. [20]. Figure 8 shows a schematic diagram of the computational domain and experimental conditions. The Cartesian coordinate system employed is also demonstrated in Figure 8; x, y, and z denote the streamwise, transverse, and spanwise directions, respectively. The width of the computational domain is 40 mm, while the dimensions of the streamwise and transverse directions are 200 mm and 40 mm, respectively. The nozzle with a diameter of 0.5 mm is installed in the center of the bottom floor and 50 mm downstream of the leading edge. The number of the grid points is 401 × 21 × 41 in x, y, and z directions, respectively, and the grid near the nozzle and the wall are refined. The effect of the grid has been investigated in our previous work, and the grid used in this paper fulfills the requirement of independence [42–44]. Water is used as the test substance, whose physical properties are illustrated in Table 1, the initial droplet diameter is 100 µm. The freestream crossflow air has a Mach number of 1.94, a static pressure of 29 kPa, and a static temperature of 304.1 K. Details of the simulation are listed in Table 2.

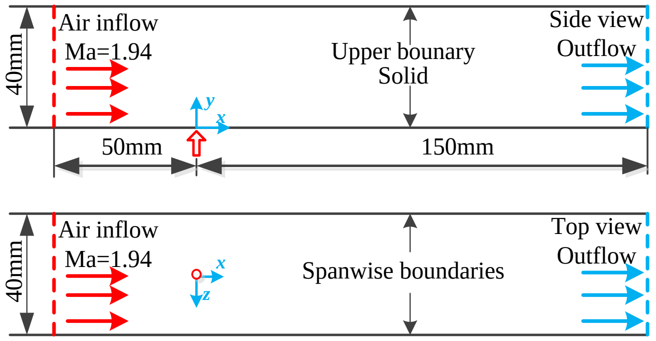

Figure 8. Schematic of the computation domains and conditions.

Table 1. Physical properties of water.

Density	Viscosity	Surface Tension
988 kg/m^3	2.67×10^{-3} kg/(m·s)	0.075 N/m

Table 2. Simulation conditions and parameters.

Supersonic Crossflow (Air)		Jet-Exit Flow (Water)	
Mach number	1.94	Gas-liquid momentum ratio	7
Static temperature	304.1 K	Injector nozzle diameter	0.5 mm
Static pressure	29 KPa	Water temperature	298 K
Velocity	678.13 m/s	Injection velocity	32.73 m/s

For the gas phase, a supersonic inflow condition is used at the inlet, and the 1/7th power-law velocity profile is adopted. The extrapolation method is used at the outlet. Symmetry boundaries are employed at the spanwise boundaries. All walls are in non-slip, non-penetration, and adiabatic condition. For the droplet–wall collision, the particle rebound correlation proposed by Grant [45] is adopted.

Figure 9 demonstrates the pressure contour at the bottom wall and center plane perpendicular to the *z*-axis, the streamwise velocity contour at the plane of $x = 0.08$ m, and the simulated droplets, whose color represents their streamwise velocity. When injected into the supersonic crossflow, the water jet acts as an obstruction to the supersonic flow. As a result, a bow shock is formed in front of the jet, where the pressure changes sharply, manifesting as a high-pressure zone upstream of the jet. Near the boundary layer, the reverse pressure gradient transports air upstream and forms a separation zone. The droplets are rapidly broken down into finer sizes by interacting with the crossflow. As the droplets are swept down in the direction of the crossflow, the span of their distribution increases in width. The droplet streamwise velocity in the periphery of the atomization zone is relatively high, while in the core and the near-wall region, the velocity is lower. Moreover, due to the spray, the streamwise velocity of air of the core region is lowered even down to subsonic levels. These qualitative phenomena are consistent with the experimental and numerical results given in reference [13].

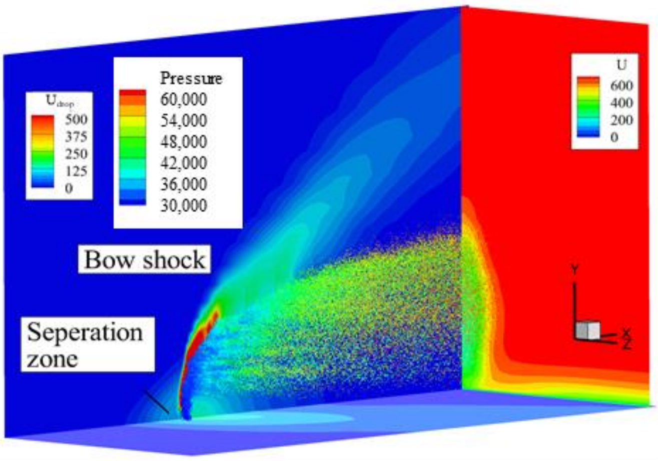

Figure 9. Superimposition of the contours of pressure, streamwise velocity, and droplets.

Figure 10 compares the penetration height of our model with that observed in the experiment of Lin et al. [20]. The black dots represent the instantaneous water droplets. The solid and dashed lines represent the average jet boundary of numerical results based on the CPU/GPU model and that of the experiment, respectively. It can be seen that the droplet distribution obtained is in good agreement with that of the experiment [20]. In addition, the penetration height obtained by Im et al. [46] using conservation-element and solution-element (CE/SR) is also shown in Figure 10. These results prove that our model can simulate a liquid jet in supersonic crossflow well.

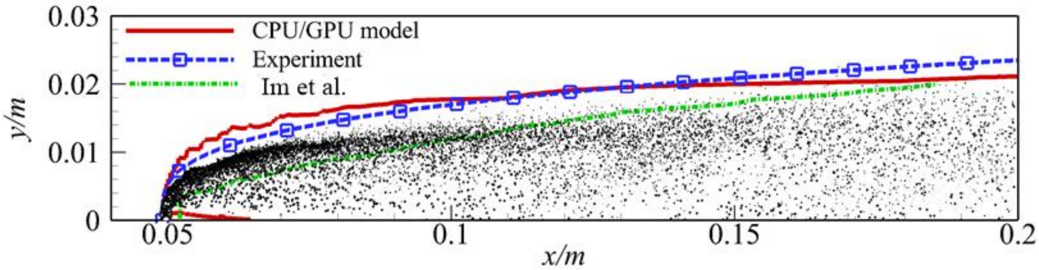

Figure 10. Comparison of simulated spray penetration with experiments [12] in central plane.

3.2. Performance Analysis

To test the proposed CPU/GPU mode, five simulations of the jet spray based on the experiments of Lin et al. [20] are performed using two devices. The first device consists of two Intel Xeon Gold 5218 CPUs (Intel Corporation, Palo Alto, CA, USA) and runs CPU-based code. The other device consists of a Tesla V100 GPU (NVIDIA Corporation, Santa Clara, CA, USA) and uses the numerical scheme described in this paper. The specifications of CPU and GPU computing environments are listed in Table 3. For the CPU/GPU mode, a single CPU core coupled with one GPU is used, whereas for the CPU mode, 32 cores of the CPUs are used. The five simulations are prescribed with different numbers of computational droplets, namely, about 0.06 million, 0.12 million, 0.24 million, 0.48 million, and 0.96 million, resulting in the droplet-to-grid ratios of 0.09375, 0.1875, 0.375, 0.75, and 1.5, respectively. Details are shown in Table 4.

Table 3. Specifications of CPU and GPU computing environments.

Indices		Intel Xeon Gold 5218	Tesla V100
No. of cores/shading units		16	5120
DP theoretical performance		1.18 TFlop/s	14 TFlop/s
Base/boost frequency		2300/3900 MHz	1230/1380 MHz
Cache	L1	1 MB	128 KB (per SM)
	L2	16 MB	6 MB
	L3	22 MB	-
Max Memory Bandwidth		897 GB/s	900 GB/s
TDP		250 W	125 W

Table 4. Number of grids and droplets for five simulations.

Case	Number of Grids (Million)	Number of Droplets (Million)	Droplet-to-Grid Ratio
Case1	0.64	0.06	0.09375
Case2	0.64	0.12	0.1875
Case3	0.64	0.24	0.375
Case4	0.64	0.48	0.75
Case5	0.64	0.96	1.50

Speedup (*SP*), an important parameter that can be used to assess the performance of a hybrid algorithm, is defined as:

$$SP = \frac{t_{\text{cpu}}}{t_{\text{gpu}}} \tag{34}$$

where t_{cpu} and t_{gpu} are the runtimes of one iteration for the CPU mode and that for the GPU/CPU mode, respectively. In this paper, the runtime of one iteration step is achieved by averaging the execution time of 1000 time steps.

Figure 11 shows the time cost and composition for the two models. The green bar is the time cost for the gas phase, and the orange bar is that for the droplet phase. In the five simulations, the grid remains unchanged. Thus, the time cost for the gas phase remains the same. However, the CPU model has a higher time cost, while the droplet time consumption for the two models grows with increasing droplet numbers. However, the increase of the CPU model is not obvious, which is related to the management method of droplets on the CPU program. The CPU program associates each droplet with a control cell and establishes a linked list structure for the control cell to manage them dynamically. This droplet management method can provide great convenience in updating the gas phase information around the droplet and calculating the droplets' contribution to the gas phase. In addition, this droplet management method keeps the linked list structure from being too long. However, the inconvenience of this method is that all control cells need to be traversed when solving the liquid phase kernel function. In other words, even if there is no droplet in the computational domain, the calculation using the CPU code still costs time. Figure 12 illustrates the gas, droplet, and total computation speedups of the CPU/GPU mode for the different numbers of droplets simulated. It can be seen that, as the droplet-to-grid ratio varies from 0.09375 to 1.5, the gas phase speedup remains around 40, whereas the acceleration rate of the droplet phase slows down as the number of droplets increases. The latter trend can be attributed to the time required for data transfer between the array on GPU and CPU in the droplet's breakup and subtraction.

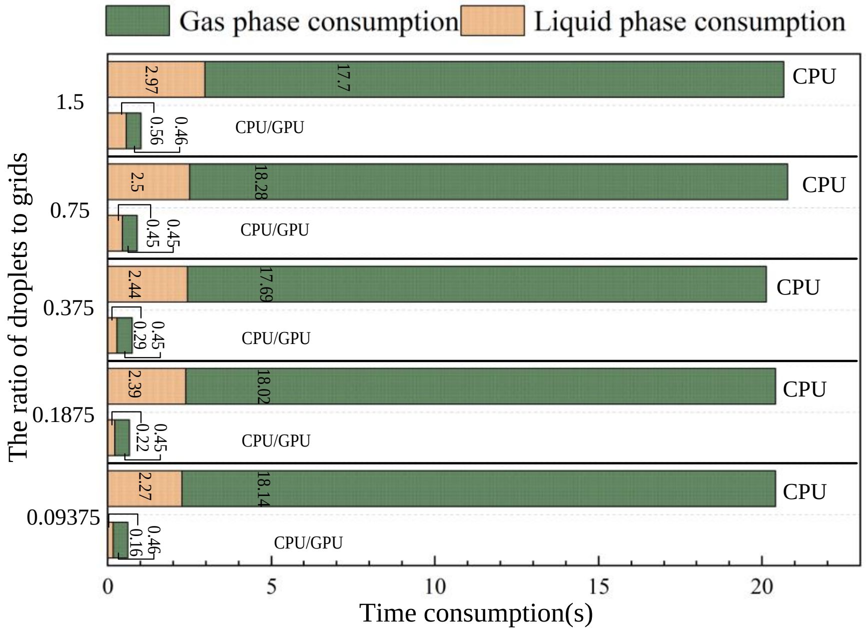

Figure 11. Time consumption and composition for simulating various ratios of droplets to grids, using the CPU/GPU and CPU model.

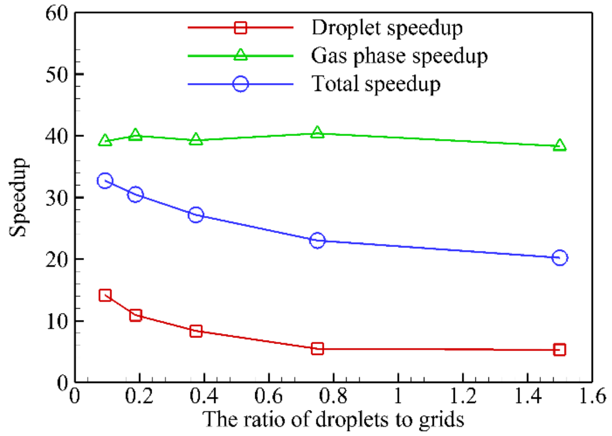

Figure 12. Speedup of one core of CPU with one GPU for different ratios of droplets to grids.

Figure 13 shows the partition of time taken for the main steps of droplet phase calculation, and the time consumption percentage of the sub-process in the droplet's breakup is also displayed in Figure 13. It can be seen that the droplets' breakup and subtraction account for the most share of the time cost due to the data transfer. In addition, in the droplet's breakup, the processes of random number generation for the distribution of the new small droplets and the droplets screening also occupy a certain proportion. The next bottleneck is the droplet's positioning. This is because the method we use to address the droplet is a loop, it starts from the cell containing the droplet at the last moment, and its purpose is to locate the neighboring cell one at a time until the cell containing the droplet is found in the present moment. This method will show its strengths in a refined and large-scale grid. Overall, the total speedup is considerable. When the ratio of droplets to grids is 1.5, the speedup is as high as 20.2. However, the lag caused by data transfer between entities should be the main issue to be addressed in the future.

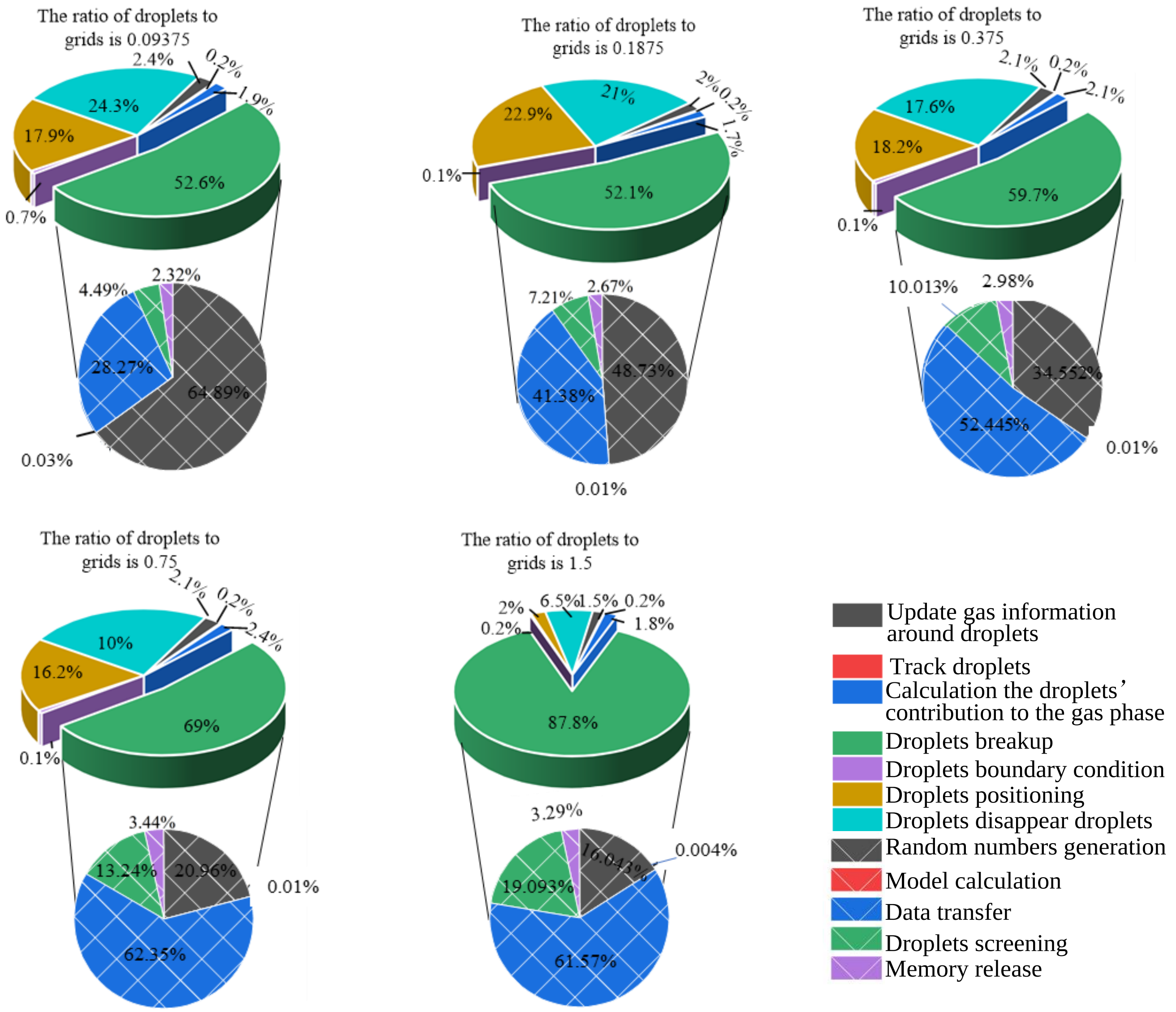

Figure 13. The partition of time cost for the main calculation steps of the droplet phase calculation.

In addition, the issue of the CPUs' load imbalance is caused by the fact that the number of simulated droplets in each core of the CPU partition varies greatly. However, for the CPU/GPU mode, each simulated droplet is treated as a computing unit. Therefore, the computational load is evenly distributed across the GPU cores for fine-grained parallel computing.

4. Conclusions

In the present work, a two-phase CPU/GPU heterogeneous parallel computation mode based on the Euler–Lagrange approach is established to simulate gas–liquid supersonic flows. The main conclusions are as follows.

1. Taking full advantage of CPU and GPU, an efficient parallel model for simulation of a liquid jet in supersonic flow is developed, in which the data-intensive and highly parallel tasks such as the kernel computation of gas phase and liquid phase are implemented on GPU, whereas the logical and data dependent tasks like the data transfer and general control are executed on CPU.
2. An effective method for droplet dynamic management and efficient calculation on the CPU/GPU model is proposed, in which a redundantly allocated array on the GPU

and the CPU is used to manage and calculate the simulated droplets, and a method based on the two-pointer method is applied to subtract the disappeared droplets.

3. A droplet-locating algorithm is developed, in which a determination criterion based on scalar product and a search approach by traveling through the neighboring cell is applied to address the cell.
4. Simulation of a liquid jet in supersonic crossflow is implemented to verify the reliability of the CPU/GPU mode. The result agrees well with the experiment.
5. A simulation of a jet spray in supersonic flow is executed using CPU mode and CPU/GPU mode, respectively, to analyze the method's efficiency and limitations. Although the speedup diminishes with increasing droplet number, the benefit is still very substantial even for the worst-case scenario studied, i.e., when the ratio of droplets to grids is 1.5, the overall speedup is 20.2.

Although the two-phase CPU/GPU heterogeneous parallel computation mode based on the Euler–Lagrange approach proposed in this paper is used for liquid in supersonic flow, it is also applicable to other Euler–Lagrange simulations.

Author Contributions: Conceptualization, X.L., M.S. and H.W.; methodology, X.L. and P.L.; software, X.L., M.S., H.W., C.W., D.X. and P.L.; validation, X.L., C.W., G.Z. and Y.Y.; formal analysis, X.L., H.W. and P.L.; investigation, X.L., H.W. and P.L.; resources, H.W., P.L., M.S. and Y.Y.; data curation, X.L. and H.W.; writing—original draft preparation, X.L.; writing—review and editing, H.W., P.L., D.X. and G.Z.; visualization, X.L.; supervision, M.S., H.W. and P.L.; project administration, M.S. and H.W.; funding acquisition, H.W. and P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grant number T2221002, 11925207, 12102472.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The study did not report any data.

Conflicts of Interest: The authors report there are no competing interest to declare.

References

1. Chang, J.; Zhang, J.; Bao, W.; Yu, D. Research progress on strut-equipped supersonic combustors for scramjet application. *Prog. Aerosp. Sci.* **2018**, *103*, 1–30. [CrossRef]
2. Duan, Y.; Yang, P.; Xia, Z.; Feng, Y.; Li, C.; Zhao, L.; Ma, L. Experimental Study of the Formation and Evolution of Gas Jets in Supersonic Combustion Chambers. *Appl. Sci.* **2023**, *13*, 2202. [CrossRef]
3. Zhang, J.; Yang, D.; Wang, Y.; Zhang, D. A Mixing Process Influenced by Wall Jet-Induced Shock Waves in Supersonic Flow. *Appl. Sci.* **2022**, *12*, 8384. [CrossRef]
4. Tian, Y.; Yang, S.; Le, J.; Su, T.; Yue, M.; Zhong, F.; Tian, X. Investigation of combustion and flame stabilization modes in a hydrogen fueled scramjet combustor. *Int. J. Hydrog. Energy* **2016**, *41*, 19218–19230. [CrossRef]
5. Tian, Y.; Shi, W.; Zhong, F.; Le, J. Pilot hydrogen enhanced combustion in an ethylene-fueled scramjet combustor at Mach 4. *Phys. Fluids* **2021**, *33*, 015105. [CrossRef]
6. Waltrup, P.J. Upper bounds on the flight speed of hydrocarbon-fueled scramjet-powered vehicles. *J. Propuls. Power* **2001**, *17*, 1199–1204. [CrossRef]
7. Liu, C.Y.; Wang, Z.G.; Wang, H.B.; Sun, M.B. Mixing characteristics of a transverse jet injection into supersonic crossflows through an expansion wall. *Acta Astronaut.* **2016**, *129*, 161–173. [CrossRef]
8. Tian, Y.; Yang, S.; Le, J.; Zhong, F.; Tian, X. Investigation of combustion process of a kerosene fueled combustor with air throttling. *Combust. Flame* **2017**, *179*, 74–85. [CrossRef]
9. Tian, Y.; Xiao, B.G.; Zhang, S.P.; Xing, J.W. Experimental and computational study on combustion performance of a kerosene fueled dual-mode scramjet engine. *Aerosp. Sci. Technol.* **2015**, *46*, 451–458. [CrossRef]
10. Liu, X.; Li, P.; Li, F.; Wang, H.; Sun, M.; Wang, C.; Yang, Y.; Xiong, D.; Wang, Y. Effect of kerosene injection states on mixing and combustion characteristics in a cavity-based supersonic combustor. *Chin. J. Aeronaut.* **2023**. [CrossRef]
11. Tian, Y.; Le, J.; Yang, S.; Zhong, F. Investigation of Combustion Characteristics in a Kerosene-Fueled Supersonic Combustor with Air Throttling. *AIAA J.* **2020**, *58*, 5379–5388. [CrossRef]
12. Li, F.; Wang, Z.G.; Li, P.B.; Sun, M.B.; Wang, H.B. The spray distribution of a liquid jet in supersonic crossflow in the near-wall region. *Phys. Fluids* **2022**, *34*, 063301. [CrossRef]

13. Li, P.; Wang, Z.; Bai, X.-S.; Wang, H.; Sun, M.; Wu, L.; Liu, C. Three-dimensional flow structures and droplet-gas mixing process of a liquid jet in supersonic crossflow. *Aerosp. Sci. Technol.* **2019**, *90*, 140–156. [CrossRef]
14. Li, P.; Li, C.; Wang, H.; Sun, M.; Liu, C.; Wang, Z.; Huang, Y. Distribution characteristics and mixing mechanism of a liquid jet injected into a cavity-based supersonic combustor. *Aerosp. Sci. Technol.* **2019**, *94*, 105401. [CrossRef]
15. Li, C.; Li, C.; Xiao, F.; Li, Q.; Zhu, Y. Experimental study of spray characteristics of liquid jets in supersonic crossflow. *Aerosp. Sci. Technol.* **2019**, *95*, 105426. [CrossRef]
16. Li, C.Y.; Li, P.B.; Li, C.; Li, Q.L.; Zhou, Y.Z. Experimental and numerical investigation of cross-sectional structures of liquid jets in supersonic crossflow. *Aerosp. Sci. Technol.* **2020**, *103*, 105926. [CrossRef]
17. Yoo, Y.-L.; Han, D.-H.; Hong, J.-S.; Sung, H.-G. A large eddy simulation of the breakup and atomization of a liquid jet into a cross turbulent flow at various spray conditions. *Int. J. Heat Mass Transf.* **2017**, *112*, 97–112. [CrossRef]
18. Dai, Q.; Luo, K.; Jin, T.; Fan, J. Direct numerical simulation of turbulence modulation by particles in compressible isotropic turbulence. *J. Fluid Mech.* **2017**, *832*, 438–482. [CrossRef]
19. Dai, Q.; Jin, T.; Luo, K.; Fan, J. Direct numerical simulation of particle dispersion in a three-dimensional spatially developing compressible mixing layer. *Phys. Fluids* **2018**, *30*, 113301. [CrossRef]
20. Lin, K.-C.; Kennedy, P.J.; Jackson, T.A. Structures of water jets in a Mach 1.94 supersonic crossflow. In Proceedings of the 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 5–8 January 2004; pp. AIAA 2004-971.
21. Wu, L.Y.; Chang, Y.; Zhang, K.L.; Li, Q.L.; Li, C.Y. Model for three-dimensional distribution of liquid fuel in supersonic crossflows. In Proceedings of the 21st AIAA International Space Planes and Hypersonics Technologies Conference, Xiamen, China, 6–9 March 2017; pp. AIAA 2017-2419.
22. Li, P.; Wang, Z.; Sun, M.; Wang, H. Numerical simulation of the gas-liquid interaction of a liquid jet in supersonic crossflow. *Acta Astronaut.* **2017**, *134*, 333–344. [CrossRef]
23. Li, X.; Liu, W.; Pan, Y.; Yang, L.; An, B.; Zhu, J. Characterization of kerosene distribution around the ignition cavity in a scramjet combustor. *Acta Astronaut.* **2017**, *134*, 11–16. [CrossRef]
24. Li, P.B.; Wang, H.B.; Sun, M.B.; Liu, C.Y.; Li, F. Numerical study on the mixing and evaporation process of a liquid kerosene jet in a scramjet combustor. *Aerosp. Sci. Technol.* **2021**, *119*, 107095. [CrossRef]
25. Song, J.; Jeong, H.; Jeong, J. Performance Optimization of Object Tracking Algorithms in OpenCV on GPUs. *Appl. Sci.* **2022**, *12*, 7801. [CrossRef]
26. Mo, T.; Li, G. Parallel Accelerated Fifth-Order WENO Scheme-Based Pipeline Transient Flow Solution Model. *Appl. Sci.* **2022**, *12*, 7350. [CrossRef]
27. Guo, M.; Dong, Z.; Keutzer, K. SANA: Sensitivity-Aware Neural Architecture Adaptation for Uniform Quantization. *Appl. Sci.* **2023**, *13*, 10329. [CrossRef]
28. Liu, R.K.-S.; Wu, C.-T.; Kao, N.S.-C.; Sheu, T.W.-H. An improved mixed Lagrangian–Eulerian (IMLE) method for modelling incompressible Navier–Stokes flows with CUDA programming on multi-GPUs. *Comput. Fluids* **2019**, *184*, 99–106. [CrossRef]
29. Salvadore, F.; Bernardini, M.; Botti, M. GPU accelerated flow solver for direct numerical simulation of turbulent flows. *J. Comput. Phys.* **2013**, *235*, 129–142. [CrossRef]
30. Jonker, H.J.J.; Schalkwijk, J.; Siebesma, A.P.; Van Meijgaard, E. Weather Forecasting Using GPU-Based Large-Eddy Simulations. *Bull. Am. Meteorol. Soc.* **2015**, *96*, 715–723. [CrossRef]
31. Lai, J.; Li, H.; Tian, Z. CPU/GPU Heterogeneous Parallel CFD Solver and Optimizations. In Proceedings of the 2018 International Conference on Service Robotics Technologies-ICSRT '18—ICSRT '18, Chengdu China, 16–19 March 2018; pp. 88–92.
32. Sweet, J.; Richter, D.H.; Thain, D. GPU acceleration of Eulerian–Lagrangian particle-laden turbulent flow simulations. *Int. J. Multiph. Flow* **2018**, *99*, 437–445. [CrossRef]
33. Ge, W.; Sankaran, R.; Chen, J.H. Development of a CPU/GPU portable software library for Lagrangian–Eulerian simulations of liquid sprays. *Int. J. Multiph. Flow* **2020**, *128*, 103293. [CrossRef]
34. Xua, J.; Qi, H.B.; Fang, X.J.; Lu, L.Q.; Ge, W.; Wang, X.W.; Xu, M.; Chen, F.G.; He, X.F.; Li, J.H. Quasi-real-time simulation of rotating drum using discrete element method with parallel GPU computing. *Particuology* **2011**, *9*, 446–450. [CrossRef]
35. Xu, M.; Chen, F.; Liu, X.; Ge, W.; Li, J. Discrete particle simulation of gas–solid two-phase flows with multi-scale CPU–GPU hybrid computation. *Chem. Eng. J.* **2012**, *207–208*, 746–757. [CrossRef]
36. Ikebata, A.; Xiao, F. GPU-accelerated large-scale simulations of interfacial multiphase fluids for real-case applications. *Comput. Fluids* **2016**, *141*, 235–249. [CrossRef]
37. Lai, J.; Tian, Z.; Yu, H.; Li, H. Numerical investigation of supersonic transverse jet interaction on CPU/GPU system. *J. Braz. Soc. Mech. Sci. Eng.* **2020**, *42*, 81. [CrossRef]
38. Lai, J.; Yu, H.; Tian, Z.; Li, H. Hybrid MPI and CUDA Parallelization for CFD Applications on Multi-GPU HPC Clusters. *Sci. Program.* **2020**, *2020*, 8862123. [CrossRef]
39. Wright, M.J.; Candler, G.V.; Prampolini, M. Data-parallel lower-upper relaxation method for the Navier-Stokes equations. *AIAA J.* **1996**, *34*, 1371–1377. [CrossRef]
40. Tofighian, H.; Amani, E.; Saffar-Avval, M. Parcel-number-density control algorithms for the efficient simulation of particle-laden two-phase flows. *J. Comput. Phys.* **2019**, *387*, 569–588. [CrossRef]
41. Chorda, R.; Blasco, J.A.; Fueyo, N. An efficient particle-locating algorithm for application in arbitrary 2D and 3D grids. *Int. J. Multiph. Flow* **2002**, *28*, 1565–1580. [CrossRef]

42. Liu, M.J.; Sun, M.B.; Zhao, G.Y.; Meng, Y.; Huang, Y.H.; Ma, G.W.; Wang, H.B. Effect of combustion mode on thrust performance in a symmetrical tandem-cavity scramjet combustor. *Aerosp. Sci. Technol.* **2022**, *130*, 107904. [CrossRef]
43. Xiong, D.P.; Sun, M.B.; Yu, J.F.; Hu, Z.W.; Yang, Y.X.; Wang, H.B.; Wang, Z.G. Effects of confinement and curvature on a jet in a supersonic cross-flow. *Proc. Inst. Mech. Eng. Part G* **2022**, *236*, 3518–3530. [CrossRef]
44. Ma, G.W.; Sun, M.B.; Zhao, G.Y.; Liang, C.H.; Wang, H.B.; Yu, J.F. Effect of injection scheme on asymmetric phenomenon in rectangular and circular scramjets. *Chin. J. Aeronaut.* **2023**, *36*, 216–230. [CrossRef]
45. Grant, G.; Tabakoff, W. Erosion Prediction in Turbomachinery Resulting from Environmental Solid Particles. *J. Aircr.* **1975**, *12*, 471–478. [CrossRef]
46. Im, K.-S.; Zhang, Z.-C.; Cook, G., Jr.; Lai, M.-C.; Chon, M.S. Simulation of Liquid and Gas Phase Characteristics of Aerated-Liquid Jets in Quiescent and Cross Flow Conditions. *Int. J. Automot. Technol.* **2019**, *20*, 207–213. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.