High performance computing of stiff bubble collapse on CPU-GPU heterogeneous platform

Remy Dubois, Eric Goncalves da Silva, Philippe Parnaudeau

High performance computing of stiff bubble collapse on CPU-GPU heterogeneous platform

Remy Dubois[b], Eric Goncalves da Silva[a,*], Philippe Parnaudeau[a]

[a]*Institut Pprime, CNRS, UPR 3346,*
11 Boulevard Marie et Pierre Curie, 86962 Futuroscope Chasseneuil Cedex, France
[b]*IDRIS, CNRS, UPS 851,*
Campus Universitaire d'Orsay Rue John Von Neumann Bâtiment 506 BP 167 91403 Orsay Cedex, France

Abstract

SCB is an efficient fluid solver developed for computing two-phase compressible flows involving strong shocks and expansion waves. It solves a four-equation diffuse-interface model, which is derived from the five-equation model proposed by Kapila et al. The governing equations are discretized by a finite volume method with explicit time stepping. SCB uses a fully parallel environment via Message Passing Interfaces (MPI). With the fast growing number of heterogeneous computing platforms including disparate hardware architectures, it becomes nowadays necessary to develop hybrid parallelization strategies with a special care to portability. In this context, we present an heterogeneous computing framework based on MPI library and OpenACC. The choice of OpenACC is discussed. Performances, scalability and adaptability are illustrated through a series of tests on an heterogeneous architecture. Validations are proposed on various bubble collapses, in free-field or near a rigid wall. Comparisons are done with existing results and analytical solutions. Furthermore a stiff shock-induced bubble collapse demonstrates the capabilities and the high potential of the code.

Keywords: High performance computing, Compressible multiphase flow, Shock waves, Heterogeneous CPU-GPU computing

*Corresponding author.
Email addresses: remy.dubois@idris.fr (Remy Dubois), eric.goncalves@ensma.fr (Eric Goncalves da Silva), philippe.parnaudeau@univ-poitiers.fr (Philippe Parnaudeau)

1. Introduction

Cavitation erosion occurs when vapor bubbles collapse in the vicinity of solid walls. Erosion then leads to adverse consequences, such as vibrations, material damages and performance loss. Numerous experimental studies of the collapse of cavity in water either of inertial-type or under shock loading aimed to understand the physical mechanisms [34, 5, 6, 39]. For a cavitation bubble near a rigid wall, the high-speed liquid jet toward the boundary and the shock wave emission during the collapse are responsible for cavitation erosion and structural damages. This leading to the reduction of the expected lifespan of hydraulic components.

Shock-induced bubble collapse is an extremely violent event: the pressures generated by the shock wave after jet impact sometimes reach a few GigaPascal and the jet velocity can reach over 1000 m/s. Numerically, the simulation of such configurations is stiff to perform and remains challenging. It has been extensively investigated to better understand the involved physical phenomena [3, 21, 25, 49, 15] and also to test and develop numerical schemes or models [1, 2, 31, 29, 40, 38, 27, 11]. Yet, most of computations have been limited to two dimensions or axisymmetric descriptions. Indeed, such simulations require not only an accurate model to treat the interface region and the complex shock waves interaction correctly, but also highly efficient flow solvers to manage the necessary large resolution near interfaces: computational grids can reach over 1 billion nodes. Up to now, one can only cite few detailed fully three-dimensional simulations, and furthermore, with limited resolution. The works of Hawker and Ventikos [19] deeply investigated the various stages of shock-bubble interaction using a front-tracking technique. They performed 3D simulations on a grid composed by 64 million cells, corresponding to 100 points per bubble radius. In their study about shockwave lithotripsy, Coralic and Colonius [8] used a spatial resolution about 50 cells per bubble radius, for bubble collapse simulations occurring in a vessel. To test their multicomponent model, Beig and Johnsen [4] conducted various 3D simulations of shock-bubble interaction on a uniform grid composed by $500\times400\times400$ nodes (80 million nodes). Yet, the number of points per bubble radius was not given. More recently, high-resolved simulations based on a five-equation model have been proposed by Wermelinger et al. [46] to study a cloud collapse. They performed simulations on a computational grid composed by $1728\times1280\times2560$ nodes (more than 5 billion nodes), corresponding to 250 points per bubble radius. Their efficient solver was parallelized with a hybrid paradigm using the MPI and OpenMP programming models.

Due to the inherent large computational cost of such 3D simulations, the use of supercomputers turns out to be indispensable. The supercomputers's world is evolving in order to offer ever-increasing performance, as reflected by the Top500 list [42]. One aspect on which constructors are focusing on is the computational's part of the supercomputer. In the last 20 years, the enhancement of processor performances has been mainly oriented on increasing two aspects: the number of cores and the size of the registers. This posturing has been followed to obtain better energy efficiency by processors, while preserving Moore's law. Still aiming to achieve greater energy efficiency, hybridization of supercomputers began around ten years ago and today, nothing has been launched without it. This heterogeneous technology is becoming more and more available for a reasonable order of magnitude computers. Thus, little by little, the need to use accelerator devices like Graphics Processing Units (GPU) was imposed to the High Performance Computing (HPC) community, as well as numerous related paradigms to use them, among which the most popular: CUDA, OpenACC, OpenCL and OpenMP. Since various programming models must be considered, leveraging the compute power of hybrid architectures can be really challenging. CUDA is widely used in the HPC community [32, 9], but Xia *et al.*[48] illustrated the benefits of using the OpenACC directives on GPU and, more recently, Li and Shih [26] showed that OpenACC can achieve quite similar performances compared to CUDA.

In this work, we investigate the performance of the multiphase solver SCB devoted to simulate stiff bubble collapses and compressible liquid-gas flows. The code solves a four-equation hyperbolic system [13, 14], which is derivated from the five-equation model proposed by Kapila *et al.*[23] and Murrone and Guillard [30]. In previous studies [16, 17], a programming approach combining MPI and OpenMP libraries have been successfully implemented and the high potential of the four-equation model has been revealed on challenging 3D stiff cases. Here, we extend the solver for GPU-accelerated clusters using OpenACC directives. Moreover, to fully exploit modern heterogeneous supercomputers based on CPU and GPU, a special focus is done on a hybrid parallel strategy combining MPI and OpenACC in order to perform efficient simulations of strong bubble collapses. A particular attention is paid to treat performance and portability with equal importance. To demonstrate the capabilities of this solver, various 3D cases are computed: free-field collapse and shock-induced collapse near a rigid wall. The latter has been compared with the results of Wermelinger *et al.*[46]. Finally, the simulation of a stiff case proposed by Paquette *et al.*[33] where potential wall damages were reported, is performed. Different analyses

are led by comparing cases where the shock intensity increases and where the stand-off distance varies, showing significant discrepancies with the 2D cylindrical cases.

The paper is organized as follows: the governing equations and models are introduced in Section 2. Section 3 presents the discretization of the four-equation system. Section 4 provides an overview of the parallel paradigms implemented in our solver and Section 5 discusses performances. Section 6 presents benchmark examples for validation as well as results from a stiff shock-induced bubble collapse near a wall. Finally Section 7 gives conclusion and perspectives.

2. Four-equation model and governing equations

In the framework of homogeneous mixture model, one can assume that each phase is compressible. Viscous effects are not considered in this study. According to different studies [21, 41], the Reynolds numbers range is high which justify neglecting viscosity in simulations. Surface tension effects are also neglected. Indeed, Weber numbers $W_e = \rho U^2 D/\sigma$ based on bubble diameter and jet velocity (around 800 m/s) are higher than 8.10^6. Thus, inertia is expected to dominate surface tension effects. Mass transfer between phases is not expected to affect the bubble dynamics over the major part of the collapse and is therefore ignored. In the following, subscripts l and g are related respectively to the liquid and gas phases. The void fraction and vapor mass fraction are α and Y, respectively. Equations read as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{1}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + P\mathbf{1}) = \mathbf{0}, \tag{2}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho H \mathbf{u}) = 0, \tag{3}$$

$$\frac{\partial \alpha}{\partial t} + \mathbf{u} \cdot \nabla \alpha = K \, \nabla \cdot \mathbf{u}, \quad \text{where} \quad K = \frac{\rho_l c_l^2 - \rho_g c_g^2}{\frac{\rho_l c_l^2}{1-\alpha} + \frac{\rho_g c_g^2}{\alpha}} \tag{4}$$

This model is built around conservation equations: mass (1), momentum (2) and total energy (3) for the mixture. As well as a transport equation (4) for the void fraction with the source/sink term on the right side.

In equations (1)-(4), ρ is the mixture density, $\mathbf{u}$ the velocity, P the pressure, $\mathbf{1}$ the identity tensor, E the mixture total energy, and H the mixture total enthalpy. The total energy is $E = e + \frac{1}{2}(\mathbf{u} \cdot \mathbf{u})$ with e the internal energy and the total enthalpy is $H = h + \frac{1}{2}(\mathbf{u} \cdot \mathbf{u})$ with h the

enthalpy. Furthermore, ρ_l, ρ_g, c_l and c_g denote respectively the density and sound velocity for each phases. Finally, one can note that $\rho = \alpha\rho_g + (1-\alpha)\rho_l$.

$K\,\nabla\cdot\mathbf{u}$ in Equation (4) models the compression effects in the mixture. Recently Wermelinger *et al.*[46] and Schmidmayer *et al.*[37] highlighted the importance of this term by comparing a five-equation model without this term [2] and one with it [23]. The major contribution from this term is that it keeps a correct thermodynamic behavior in the mixture. Schmidmayer [37] also reported that the term $K\,\nabla\cdot\mathbf{u}$ induces numerical instabilities that might be particularly amplified. A comparison of the four and five-equation models has been recently proposed for the simulation of shock-induced bubble collapse [17]. This study clearly showed the capability of the four-equation model to reproduce as correctly as the five-equation model the physical phenomena with a much lower computational cost.

Finally, an equation of state (EOS) is needed to close the problem. In the present case, a convex stiffened gas EOS [28] is selected:

$$P(\rho, e, \alpha, Y) = (\gamma(\alpha)-1)\rho(e - q(Y)) - \gamma(\alpha)P_\infty(\alpha), \tag{5}$$

$$T(\rho, h, Y) = \frac{h_l - q_l}{C_{p_l}} = \frac{h_g - q_g}{C_{p_g}} = \frac{h - q(Y)}{C_p(Y)}, \tag{6}$$

$$P_\infty(\alpha) = \frac{\gamma(\alpha)-1}{\gamma(\alpha)}\left[\alpha\frac{\gamma_g}{\gamma_g - 1}P_{\infty_g} + (1-\alpha)\frac{\gamma_l}{\gamma_l - 1}P_{\infty_l}\right], \tag{7}$$

$$\frac{1}{\gamma(\alpha)-1} = \frac{\alpha}{\gamma_g - 1} + \frac{1-\alpha}{\gamma_l - 1}, \tag{8}$$

$$q(Y) = Yq_g + (1-Y)q_l, \tag{9}$$

$$C_p(Y) = YC_{p_g} + (1-Y)C_{p_l}. \tag{10}$$

In Equations (5)-(10), the quantities without index are to be related to the mixture. P_∞ is the constant reference pressure of the EOS, q is the energy at the reference state, γ the heat capacity ratio, C_p and C_v are the thermal capacities.

The four-equation system (1)-(4) is hyperbolic and the speed of sound follows the Wallis formulation [45] *e. g.* :

$$\frac{1}{\rho c^2_{wallis}} = \frac{\alpha}{\rho_g c_g^2} + \frac{1-\alpha}{\rho_l c_l^2} \tag{11}$$

3. Numerical method

Equations (1)-(4) are solved using the cell-centered finite volume method. One can write the discretized equations system in the following generic matrix form:

$$\frac{\partial \mathbf{W}}{\partial t} + \nabla \cdot \mathbf{A} + \mathbf{S}\nabla \cdot \mathbf{u} = \mathbf{0}, \tag{12}$$

with $\mathbf{W} = (\rho, \rho\mathbf{u}, E, \alpha)^\intercal$ being the conservative state vector, $\mathbf{A} = (\rho\mathbf{u}, \rho\mathbf{u} \otimes \mathbf{u} + P\mathbf{1}, \alpha\mathbf{u})^\intercal$ the flux vector and $\mathbf{S} = (0, \mathbf{0}, 0, -(K + \alpha))^\intercal$ the source term vector. By considering a cell (i) with a volume (Ω) delimited by surface (Γ) of normal unit vector $\mathbf{n}$ and knowing that $\mathbf{S}\nabla\cdot\mathbf{u}$ is discretized following the approach of Daude *et al.*[10], the integration of system (12) becomes :

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{W} d\Omega + \oint_\Gamma \mathbf{A} \cdot \mathbf{n} d\Gamma + \tilde{\mathbf{S}} \oint_\Gamma \mathbf{u} \cdot \mathbf{n} d\Gamma = \mathbf{0}, \tag{13}$$

Where $\tilde{\mathbf{S}}$ is some average of $\mathbf{S}$ on cell i. System (13) discretized on Cartesian grid, using an explicit scheme for time evolution becomes:

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n - \Delta t\Big(\sum_{j=1}^{m} \frac{1}{\Delta x_j} \mathbf{A}^*_{\mathbf{ij}} \cdot \mathbf{n}_{\mathbf{ij}} - \tilde{\mathbf{S}} \sum_{j=1}^{m} \frac{1}{\Delta x_j} \mathbf{u}^*_{\mathbf{ij}} \cdot \mathbf{n}_{\mathbf{ij}} \Big) \tag{14}$$

Where $\mathbf{A}^*$ is a numerical flux and $\tilde{\mathbf{S}}$ is the discrete form of the non-conservative term. Finally, the numerical flux and the non-conservative term are computed by using a HLLC scheme [43]. The temporal integration is performed with a fix time-step Δt that respects the Courant-Friedrichs-Lewy condition. In order to obtain the second-order in space and time, the TVD MUSCL-Hancock method [44] is implemented by considering the three following steps.

1. Reconstruction of the solution via piecewise linear functions. As regard to robustness consideration, primitive variables $(\rho, \mathbf{u}, P, \alpha)$ are chosen and the minmod slope limiter is used.
2. Advance of the solution by half time-step $\Delta t/2$ before the derivation of time-centered interface values. With the use of primitive variables, the equations are rewrited in the quasi-linear form by introducing a matrix A_{w_l} for each direction l. The expression can be easily

computed and is given only for the x-direction:

$$A_{w_x} = \begin{bmatrix} \mathbf{u}.\mathbf{n} & \rho & 0 & 0 & 0 & 0 \\ 0 & \mathbf{u}.\mathbf{n} & 0 & 0 & 1/\rho & 0 \\ 0 & 0 & \mathbf{u}.\mathbf{n} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{u}.\mathbf{n} & 0 & 0 \\ 0 & \rho c^2_{wallis} & 0 & 0 & \mathbf{u}.\mathbf{n} & 0 \\ 0 & -K & 0 & 0 & 0 & \mathbf{u}.\mathbf{n} \end{bmatrix}$$

3. Solve the Riemann problem with the HLLC scheme to update the conservative variables.

The numerical treatment of the boundary conditions is based on the use of the characteristic relations of the four-equation system (see [17] for more details). High-resolution numerical methods based on a weighted essentially non-oscillatory (WENO) reconstruction are currently under development (see [18]).

4. Parallel paradigms and implementations

In a previous work [16], a programming approach allying the use of both MPI and OpenMP libraries have been successfully implemented. In the framework of the large deployment of CPU and GPU-based supercomputers, the parallelization of the code by combining MPI and OpenACC has been carried out. In the following, both approaches and the gains obtained are compared.

4.1. Distributed-memory parallelization: MPI

SCB uses a 5-point-stencil per direction, meaning that unknown at one point is computed using 9 and 13 neighbors in 2D and 3D, respectively. For the MPI part, to parallelize HLL-type Riemann solvers, a well-known strategy is to distribute global arrays across the processes. The decomposition is made by using an 3D block partitionning of the matrix. The main idea of this approach is to add ”ghost points” at each subdomain and to exchange data between each neighbor of one subdomain. Each subdomain has a fixed size and is organized on a Cartesian processor topology, two layers of auxiliary cells are defined at each boundary of a subdomain to couple the computation. In the case of hybrid computing with GPUs, each GPU only processes an integer number of subdomains and at least one subdomain. There is no constraint on the shape

of the subdomains, although an identical and square shape of each subdomain results in better performance.

4.2. Accelerator parallelization model: OpenMP

A simplified version of the main part of SCB with the OpenMP implementation is given in Listing 1. A fine grained programming has been implemented, is a matter of using OpenMP's directives to automatically distributes the loops around the threads, with three basic principles. The first principle assesses the size of the considered problem. Indeed, if the problem's size becomes too small, sharing the data between threads becomes irrelevant. The second principle concerns the scheduling. OpenMP loop scheduling is provided to allow better load balancing between threads. There are four kinds: *static*, *dynamic*, *guided*, each with its own advantage, and a last one, *runtime*, which enables to select between the 3 previous ones when executing a run using a variable system environment. Finally, the last principle consists of merging internal loops (*COLLAPSE*) in order to enlarge the iteration space and thus to better distribute the iterations.

4.3. Accelerator parallelization model: OpenACC

OpenACC directives move computation and data from a host device to an accelerator device. OpenACC integration, in this study, is quite similar to the OpenMP, with a notable distinction: attention should be paid when copying data from the host device to the accelerator device to avoid unnecessary round-trips between devices, as illustrated in Listing 2. A data region is firstly created with *ACC DATA COPY* directive and then closed with *ACC END DATA*.

```
1  DO ndt=1,ndtmax
2
3    !$OMP PARALLEL IF(ijmax.gt.256) default(none)
4    !$OMP DO SCHEDULE (runtime) PRIVATE (i,j,k) COLLAPSE(2)
5    DO k=kmin,kmax
6    DO j=jmin,jmax
7    DO i=imin,imax
8       RI1=w1(i,j,k)-w1(i-1,j,k)
9       sl=dmax(0.0,dmin(Ri1,1.0))+dmin(0,dmax(1,Ri1))
10      W1(i,j,k)= W1(i-1,j,k)+1/4*sl*(W1(i-1,j,k)-W1(i-2,j,k))+1/4*sl*(W1(i ,j,k)-
                W1(i-1,j,k))
11   ENDDO
12   ENDDO
13   ENDDO
14   !$OMP END DO
15   CALL BOUNDARY (W1)
16   !$OMP END PARALLEL
17   CALL MPI_SENDRECV(W1, imax*kmax, MPI_DOUBLE_PRECISION,&
            neib_mpi(N),tag, W1, imax*kmax, MPI_DOUBLE_PRECISION,&
            neib_mpi(S) ,tag, comm, status, err_mpi)
18
19 ENDDO
```

Listing 1: OpenMP implementation

```
1  !$ACC DATA COPY (W1)
2  DO ndt=1,ndtmax
3    !$ACC KERNELS DATA PRESENT(W1)
4    DO k=kmin,kmax
5    DO j=jmin,jmax
6    DO i=imin,imax
7       RI1=w1(i,j,k)-w1(i-1,j,k)
8       sl=dmax(0.0,dmin(Ri1,1.0))+dmin(0,dmax(1,Ri1))
9       W1(i,j,k)= W1(i-1,j,k)+1/4*sl*(W1(i-1,j,k)-W1(i-2,j,k))+1/4*sl*(W1(i ,j,k)-
                W1(i-1,j,k))
10   ENDDO
11   ENDDO
12   ENDDO
13   !$ACC END KERNELS
14   CALL BOUNDARY (W1)
15   !$ACC UPDATE HOST(W1)
16   CALL MPI_SENDRECV(W1, imax*kmax, MPI_DOUBLE_PRECISION,&
            neib_mpi(N),tag,W1, imax*kmax, MPI_DOUBLE_PRECISION,&
            neib_mpi(S) ,tag, comm, statut, err_mpi)
17   !$ACC UPDATE DEVICE(W1)
18 ENDDO
19 !$ACC END DATA
```

Listing 2: OpenACC implementation

Due to the time loop, *ACC DATA PRESENT* notifies that the table $W1$ is already on the device from another data region. The choice has been made to use the *ACC KERNEL* instead of *ACC LOOPS*, preferring to leave the compiler optimizing by itself as much as possible. Indeed, Diaz *et al.*[12] have shown similar performances for both approaches.

Calls to the MPI library are made from the host. Thus, the arrays used in the communications must firstly be sent to the host using the host update directive: *ACC UPDATE HOST*. Once the communication is completed, the result should be sent to the device using the update directive: *ACC UPDATE DEVICE*. These transfers must therefore be avoided as much as possible because they significantly affect performance and above all scalability.

5. Performance and scalability

The considered physical problems are characterized by high-speed dynamics and very small spatio-temporal scales leading to an intensive computing usage. The physical phenomena require the use of a very fine mesh, at least 10^6 and 10^9 cells in 2D and 3D, respectively. The physical times of these phenomena are very short (order of microseconds). Combined with the difficulty of the system (14) to be integrated by an implicit method, this implies the use of explicit time integration schemes.

For these reasons, all the performances and scalability tests have been made with the physical parameters of the problem presented in section 6.3 with a varying number of cells: 10^6 and 4×10^6 (called **1M** and **4M**, respectively) in 2D and 10^9 to 4×10^9 (called **1B** and **4B**, respectively) in 3D. It can be noted that some compressible multicomponent and multiphase flow solvers, allowing studies of collapsing bubble, offer a possibility of using an Adaptive Mesh Refinement (AMR) [46, 36, 7]. But due to the very high pressure peaks induced by the collapse of the bubble, the compression rate between regions with coarse and fine mesh may not be so far from 1. Meanwhile the computation of the refinement takes so much time that, for this case, there is no advantage. The fact that AMR does not always give gain has been reported by related studies [35].

All the tests were carried out on supercomputer Jean Zay [20], which is a HPE SGI 8600 supercomputer composed with 1528 scalar compute nodes and 261 accelerated compute nodes. Each scalar compute node is composed by 2 Intel Xeon Gold 6248 processors, and each accelerated compute node by 2 Intel Xeon Gold 6248 processors and by 4 Nvidia Tesla V100. The total peak performance is around 15.9 PFlop/s and Intel Omni-Path interconnections, with one link for nodes without GPU, and four links for the others. All tests have been performed with disabled hyperthreading option on processor. This section is organized into two parts. The first one is devoted to analysing performances on a single device[1] in order to draw the relevance of choosing OpenACC paradigm on GPU as well as CPU. The second part is focused on the performances obtained on multiple devices.

5.1. Single-device performance

SCB performance has been investigated on one Intel core (clock signal frequency of 2.5 GHz) and intel-advisor to extract roofline [47]. The executable has been generated with Intel Ifort Compiler (version 19.0.4) using options:
-O3-fp-model strict -xHost -fma -align array64byte.
The case **1M** has been selected to perform analysis, which focuses on two aspects:

- The computational limits of the solver.
- The performances extracted from OpenMP and OpenACC paradigms on a single CPU and the performances obtained with OpenACC on the GPU.

[1] Device mean: one CPU core or a CPU with all its cores or a GPU card.

Prior to the analysis one can define arithmetic intensity, which is the ratio between floating-point operations realized by application, relative to the amount of memory accesses that are required to support them. The following metrics summarize the analysis:

- Arithmetic intensity value is around 0.3. This kind of arithmetic intensity shows that SCB is driven by memory and bound to bandwidth.
- 7% of the theoretical peak of a single core is reached.
- A vectorization rate of around 85% (for this kind of architecture) is obtained.
- A memory bandwidth of around 21 GB/s (which is not so far from Level 3 cache) is observed.

This range of value for arithmetic intensity is often observed for stencil codes [47, 22]. Optimizations could be implemented to minimize this effect by managing cache levels of processors. However, such a kind of optimization requires a deep rewriting of the compute kernel that is more challenging. For this reason, we favor optimizations through programming paradigms that do not depend on the hardware and do not require important rewriting.

Before starting the comparison between performances obtained with executables built with OpenMP or OpenACC, the influence of the results using OpenMP is being checked when executables are generated by two compilers: PGI and Intel. Performances obtained do not show any significant differences. In the worst case the executable produced by Intel compiler is 10% slower than the one obtained with PGI (see Table 1). This performance gap is caused by *-fp-model strict* optimization with the Intel compiler.

Moreover, we note that overall, the scaling across several threads is very similar. This leads to think that the obtained performances do not depend on the compiler and therefore the analysis of the performance between the two accelerator parallelization models can be driven as objectively as possible.

Table 1: Comparison of calculation times using the OpenMP library.

	case 1M		**case 4M**	
	Time		**Time**	
Threads	**Intel-v**19.0.4	**PGI-v**19.7	**Intel-v**19.0.4	**PGI-v**19.7
1	763.04	691.00	3052.60	2770.12
2	385.58	348.77	1541.07	1399.92
4	207.86	183.43	830.75	733.03
8	119.49	102.73	473.50	408.13
16	81.90	69.67	318.96	268.62
20	74.23	62.10	286.18	244.35

The relevance of the use of OpenACC is now analyzed as regard to the portability and efficiency on heterogeneous supercomputing platforms. The executable has been generated with PGI Fortran compiler (version 19.7) with options:

-Mpreprocess -fast -m64 -acc -Mvect=nocond -ta=multicore.

The parallel speedup and efficiency (see Figure 1) show that by using less than half of the whole CPU, performances with OpenMP or OpenACC are nearly the same. Using more than half cores of a CPU, discrepancies become notable. Thus, regardless of the paradigm, the performances obtained from 1 to 10 cores have a parallel efficiency of more than 80%. Then, by using more than half cores of a CPU, parallel efficiency decreases notably with OpenMP implementation, while it remains above 70% with OpenACC.

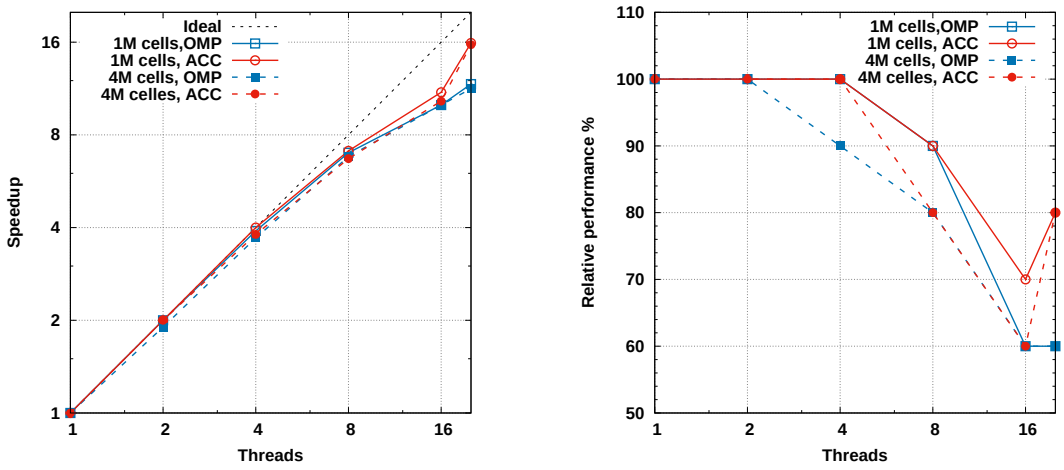


Figure 1: Strong scaling on single Intel Xeon Gold 6248. With **OMP**: OpenMP, **ACC**: OpenACC.

Yet, it is also necessary to consider the computation time provided by both implementations. The comparison of the calculation time is given in Table 2 for cases **1M** and **4M**. With 1 core the OpenACC executable is two times slower than the OpenMP one, while with 20 cores, OpenACC requires the same execution time as OpenMP. It is also noticeable that the time execution spread between OpenMP and OpenACC also decreases as the problem size increases. OpenMP scaling is mainly driven by the choice of fine-grained programming, which does not allow optimal performances.

Using this version of PGI, these results show that the choice of accelerator parallelization model for multicore CPUs can be either OpenACC or OpenMP, provided that the number of cores used is well adjusted.

Table 2: Comparison of computational time between OpenMP (**OMP**) and OpenACC (**ACC**) on CPU.

	case 1M			**case 4M**		
	Time		**Speedup**	**Time**		**Speedup**
Threads	**OMP**	**ACC**		**OMP**	**ACC**	
1	691.00	1243.85	1.8	2770.12	4047.05	1.46
2	348.77	603.18	1.73	1399.92	2016.77	1.44
4	183.43	309.99	1.69	733.03	1047.72	1.43
8	102.73	176.96	1.72	408.13	600.99	1.47
16	69.67	113.77	1.63	268.62	389.09	1.44
20	62.10	78.60	1.26	244.35	255.85	1.04

Finally, we consider the parallel performance and portability of the code on a heterogeneous supercomputer composed of CPUs and GPUs. In order to compare CPU and GPU performances separately, two tests have been set up on one device: 1 Intel Xeon Gold 6248 and 1 GPU NVidia Tesla V100 card, respectively. The executable has been generated with PGI Fortran compiler. One can see on Table 3 that GPU performances are from 2 to 27 times faster than CPU, depending on the number of cores used on CPU.

Table 3: Comparison of computational time between OpenMP on CPU (**CPU**) and OpenACC on GPU (**GPU**).

	case 1M			**case 4M**		
	Time		**Speedup**	**Time**		**Speedup**
Threads	**CPU**	**GPU**		**CPU**	**GPU**	
1	691.00	34.60	19.97	2770.12	102.79	26.94
20	62.10		1.79	244.35		2.37

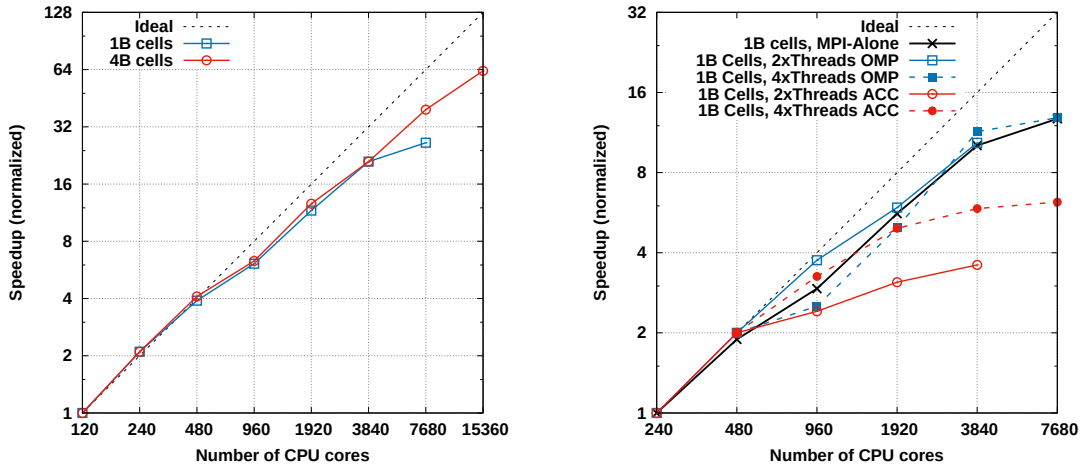
All these results reflect the ability of the OpenACC paradigm to efficiently exploit the cores of a CPU as well as GPU accelerator cards and give credits to test it on multi-accelerator-devices, *e. g.* multi-CPU versus multi-GPU.

5.2. Multi-device performance

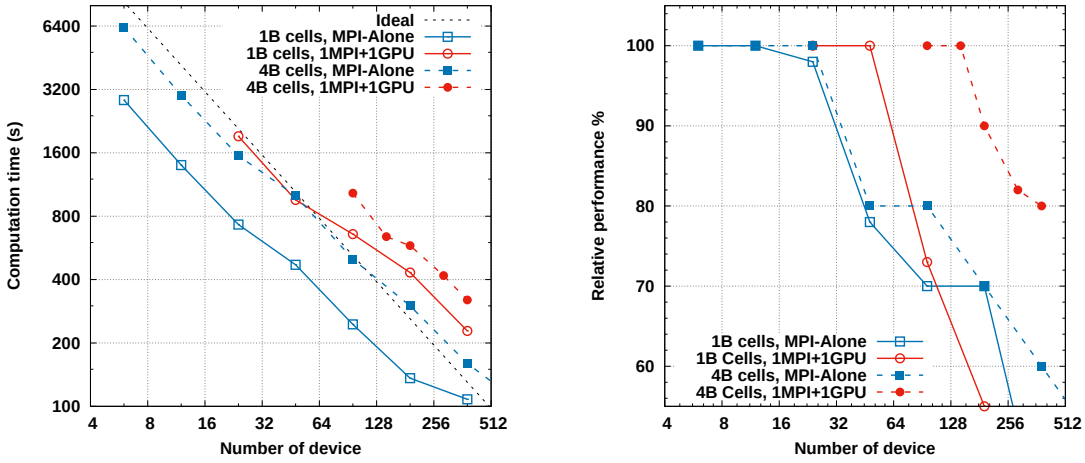
In this part, the performances of the code are studied in the context of distributed parallel computing. To begin, performances are extracted by using a single parallelization paradigm that supports data distribution on several devices, namely MPI. Then, various tests are conducted using the *hybrid* computation concept, *e. g.* in the present case, by mixing MPI with OpenMP (MPI-OpenMP) or OpenACC (MPI-OpenACC). Analyses are conducted on both a homogeneous platform (CPU only) and a heterogeneous platform (CPU and GPU).

The strong scaling tests are shown in Figure 2, using grids composed of: 1 billion cells (**1B**) and 4 billion cells (**4B**). MPI implementation alone (MPI-Alone) is tested and a parallel efficiency of around 50% relative to $15,360$ cores is observed, if the problem size is large enough. To maintain an efficiency greater than 80%, a minimum of around 10^6 cells per MPI subdomain is an optimum. This rather high value is explained by the fact that the arithmetic intensity is rather low, around 0.3, implying that the latency-dominated scenario is observed early in the speedup curve. Thus, when using less than 1 million cells per sub-domain the lower efficiency is not related to an increase in communication time, but to a decrease in computing time.

Figure 2 also shows that on a homogeneous platform, 1 MPI + 2 OpenMP threads give at least the same performances as the MPI-Alone, as long as the subdomains' size criterion provided previously, is respected. The MPI-OpenMP version provides a performance of around 75% of $3,800$ cores, which is a gain of around 5% in comparison to the MPI performance alone. Thus, a performance of around 7% of the theoretical peak performance can be sustained, with an efficiency of 75% relative to $3,800$ cores for a given problem with a size of 1 billion cells. Nevertheless, MPI-OpenACC hybrid mode on CPU cannot reproduce 2D's performances obtained in section 5.1 wih OpenACC paradigm.



(a) Homogeneous platform: CPU/CPU. Normalized speedups with MPI-Alone (left) and MPI-OpenMP or MPI-OpenACC (right).



(b) Heterogeneous platform: CPU/GPU. Device represents either 1 CPU (e.g. 20 MPI processes) or 1 GPU card attached to 1 MPI process. Scaling curve (left) and efficiency curve (right).

Figure 2: Strong scaling, speedups and performances obtained on: a) homogeneous platform composed of CPUs and b) heterogeneous platform composed of CPUs and GPUs.

Here, the performances obtained on the homogeneous platform are compared to those obtained on the heterogeneous platform. These performances have been measured by increasing the number of devices. A device corresponds to all the cores of a CPU (20 cores) on the ho-

mogeneous platform and to 1 MPI process (*e. g.* 1 core) attached to one GPU card on the heterogeneous platform case. MPI-Alone and MPI-OpenACC (GPU) scalability are quite similar (see Figure 2) as long as the size of the problem remains moderate (1B), with higher efficiency when using multi-GPU. When the problem becomes larger (4B), the multi-GPU version still maintains an efficiency of more than 80%, while it can decrease drastically with MPI-Alone or MPI-OpenMP mode. The performance curve reveals that in order to maintain an acceptable performance of 80%, the sub-domain decomposition must be with at least 10 million cells per GPU card, whereas 1 million cells per sub-domain are sufficient for CPU, like it is reported below. This explains the lower performance observed on the CPU part (Figure 2).

Lastly, while the performances in terms of scalability and efficiency are highlighted on the heterogeneous architecture, it is also necessary to consider the acceleration provided by the use of multi-GPU compared to multi-CPU, as shown on Table 4.

It can be noted that performances observed with one GPU card are not reproduced in the context of MPI-OpenACC. This lower multi-GPU performance is mostly due to the additional cost of communications that GPU cards imply. To address this drawback, some preliminary tests are currently conducted with MPI Cuda-aware.

Table 4: Comparison of 3D case (**1B**, **4B**) computational time between MPI on CPU and MPI-OpenACC on GPU.

case 1B				**case 4B**			
CPU		**GPU**		**CPU**		**GPU**	
Cores	**Times**	**Cards**	**Times**	**Cores**	**Times**	**Cards**	**Times**
240	1400	24	1917	960	1001	96	1030
480	730	48	955	1920	530	192	580
960	470	96	658	3840	318	384	320
1920	245	192	432				
3840	136	384	228				

Briefly, these performance results illustrate a very good scaling of the code from the resources offered by GPU computing. Given that the OpenACC version provides quite encouraging results

on any type of device, it becomes obvious that the OpenACC choice is interesting at the moment.

6. Numerical results

In this section, numerical tests with various 3D bubble collapses are provided. First, with the simulation of a spherical bubble collapsing in a free-field, then investigating the collapse of a bubble impacting by a normal shock wave with the same set of physical parameters, and finally showing the ability of the solver to compute a very stiff case.

6.1. Spherical bubble collapse in a free-field

The study consists of simulating the collapse of a gas bubble trapped in a liquid. This collapse is driven by the effect of a pressure jump between the internal pressure of the bubble P_b and the liquid P_{l_∞}. At the initial time, the physical parameters of the study are:

- an air bubble, with a radius $R_0 = 1$ mm and an uniform internal pressure $P_{(b,0)}$,
- the fluid is water at rest,
- pressure in the water are given by: $P_{(l,0)}(R) = \dfrac{R_0}{R}(P_{(b,0)} - P_{l_\infty})$, and the ratio of pressure is $\dfrac{P_{l_\infty}}{P_{(b,0)}} = 353$,
- Densities of air and water are initially 1 kg/m^3 and 1000 kg/m^3, respectively.

This kind of configuration, which is slightly compressible, allows to compare the evolution of the effective bubble radius R with the solution resulting from the Keller-Miksis equation [24]. The effective bubble radius is defined as:

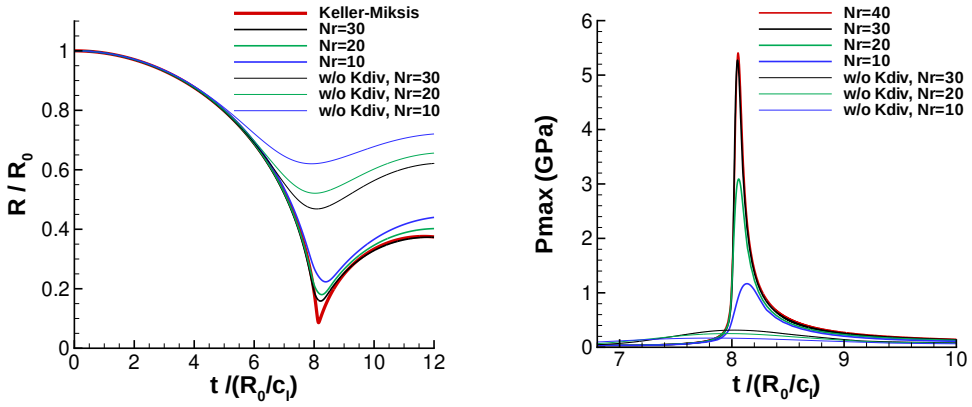
$$\mathrm{R} = \left(\frac{3V_b}{4\pi}\right)^{1/3}, \text{ with } V_b = \int \alpha \mathrm{d}\Omega$$

where V_b is the total volume of gas.

Using the symmetry of the problem, a rectangular computational domain is considered for a quarter of the bubble and taken large enough to minimize the impact of the boundary conditions. The computational domain, relatively to R_0, is $[L_x \times L_y \times L_z] = [100R_0 \times 50R_0 \times 50R_0]$. The position of the center of the bubble is $(x_b, y_b, z_b) = (50R_0, 0, 0)$. Symmetry conditions are applied in both planes $(x-y)$ and $(x-z)$, while non-reflecting conditions are applied on the other faces of the domain. Let introduce $N_{r_{(b,0)}}$, the number of points per initial bubble radius and $t^* = t/(R_0/c_l)$

a dimensionless time. The grid refinement influence is studied by considering 3 meshes with $N_{r_{(b,0)}} = [10, 20, 30]$, respectively.

The time evolution of the dimensionless bubble radius R/R_0 is presented in Figure 3 (left part) for the different meshes, with and without the Kdiv term in the void ratio equation. When using the finest grid, a very good agreement is obtained with the Keller-Miksis solution. Moreover, the effect of the Kdiv term is clearly highlighted: this term in the four-equation model produces a significantly more accurate result, especially at the minimum radius around time $t^* = 8$. Such a behaviour has been commented in recent papers using a five-equation model [46, 37].



(a) (b)

Figure 3: Time evolution for the collapse with $P_{l_\infty}/P_{(b,0)} = 353$ of: a) the effective bubble radius R/R_0 and b) the maximum pressure inside the liquid (in Gpa).

The maximum pressure evolution inside the fluid is plotted in the right part of the figure. A finer grid with $N_{r_{(b,0)}} = 40$ is added to clearly attest the mesh convergence for the peak intensity. The effect of the Kdiv term is spectacular. The pressure peak observed at the minimum radius at around time $t^* = 8$ is largely lower without this term (by a factor 12). Besides, the high magnitude obtained with the complete model and using the finest grid (more than 5 GPa) illustrates the violence of the collapse and the stiffness of the numerical problem being dealt with. On the other hand, we observe that the spatial resolution has a great influence on the pressure intensity.

6.2. Shock-induced bubble collapse near a solid wall, P_{sh} = 353 bar

The study consists of simulating the collapse of a single gas bubble immersed in water near a solid wall and impacted by a normal shock wave for which the downstream pressure is P_{sh} = 353 bar. This problem has been investigated numerically in 2D by Johnsen and Colonius [21] and more recently in 3D by Wermelinger *et al.*[46]. A spherical air bubble with an initial radius R_0, located at coordinates $(x_{(b,0)}, y_{(b,0)}, z_{(b,0)})$, is immersed in water at temperature 293 K under the following initial conditions:

$$(\rho, \mathbf{u}, p) = \begin{cases} (998 \text{ kg/m}^3, \mathbf{0} \text{ m/s}, 10^5 \text{ Pa}) & \text{inside the liquid} \\ (1 \text{ kg/m}^3, \mathbf{0} \text{ m/s}, 10^5 \text{ Pa}) & \text{inside the gas} \end{cases} \tag{15}$$

To close the problem, material parameters for the stiffened gas EOS are set to:

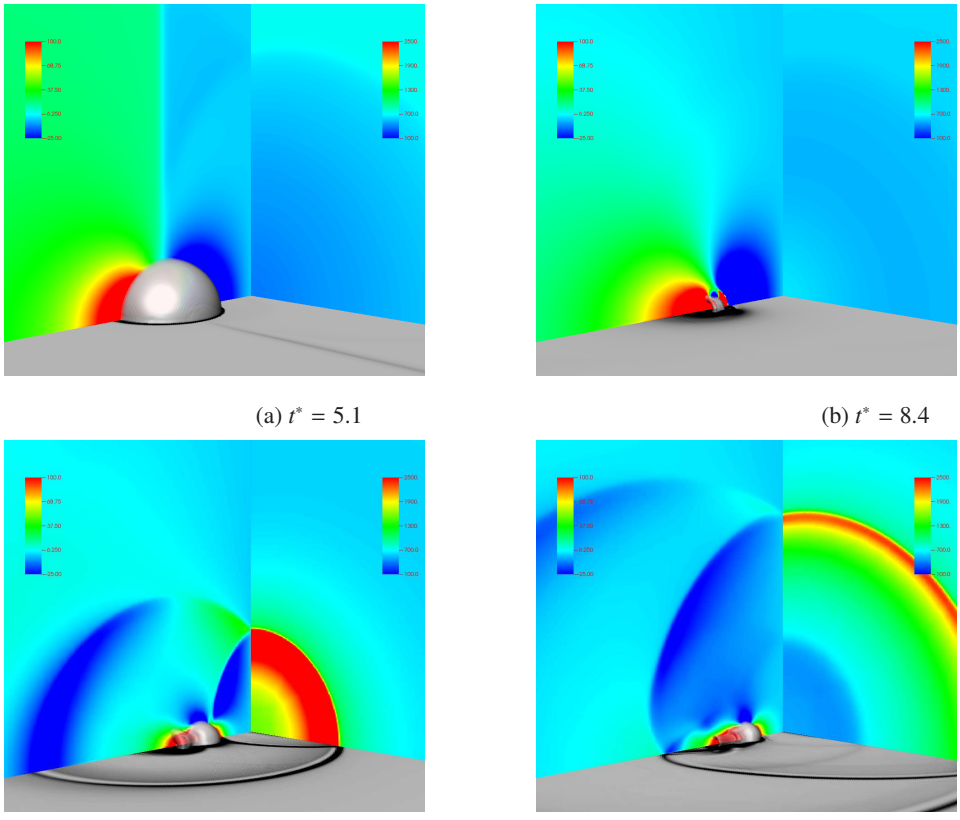
$$(\gamma_l, p_{\infty_l}, C_{p_l}) = (6.68, 4.103 \cdot 10^8 \text{ Pa}, 1650 \text{ J/kg.K}), \text{ for the liquid phase} \tag{16}$$

$$(\gamma_g, p_{\infty_g}, C_{p_g}) = (1.4, 0 \text{ Pa}, 1487 \text{ J/kg.K}), \text{ for the gas phase.} \tag{17}$$

All spatial dimensions are defined relatively to the initial radius R_0. The computational domain is $[L_x \times L_y \times L_z] = [8R_0 \times 12R_0 \times 12R_0]$ and the position of the bubble is located at $(x_{(b,0)}, y_{(b,0)}, z_{(b,0)}) = (6R_0, 0, 0)$. The incident shock wave normal to x-direction is initialized at position $x_{sh} = 4.85R_0$. The stand-off distance between the bubble center and the wall, which is a major parameter that governs the bubble collapse dynamics, is set to $H/R_0 = 2$. Due to the symmetry of the problem, only a quarter of the bubble is simulated. A slip condition is used for the wall. As previously, $N_{r_{(b,0)}}$ represents the number of nodes per initial bubble radius. Two meshes have been simulated with $N_{r_{(b,0)}} = [55, 110]$.

The main phenomena involved in this shock-bubble interaction have been described by various authors (see for example [21, 19]). Due to the pressure difference between both sides, the bubble is asymmetrically contracted and becomes toroid shaped during the process. It induces a jet of water along the axis of flow symmetry. When this water jet impacts the opposite bubble interface, an intense blast wave or water-hammer shock is formed generating a high-pressure zone. The bubble is then cut into pieces. The impact of the blast wave on the wall produces a strong pressure peak, which can cause material damages leading to erosion. To illustrate this bubble collapse, visualizations at different times $t^* = t/(R_0/c_l)$ are plotted in Figure 4. Different

quantities are presented on borders: the longitudinal velocity component (in m/s) on the the vertical symmetry plane, the pressure (in bar) on the wall and the modulus of the density gradient (Schlieren-like representation) on the horizontal symmetry plane. Inside the volume, the isosurface of void ratio $\alpha = 0.15$ is plotted. At time $t^* = 5.1$, the incident shock wave has impacted the wall and the reflected wave is located on the bubble. At time $t^* = 8.4$, one can clearly observe the toroidal shape of the bubble, the high-speed jet penetrating inside the bubble and the generation of the blast wave. At time $t^* = 10.1$, the leftward front of the blast wave has impacted the wall causing an intense pressure peak and the reflected wave propagates toward the bubble pieces. Finally at time $t^* = 12.1$, the reflected wave has impacted the bubble fragments leading to a recollapse and continues its propagation.

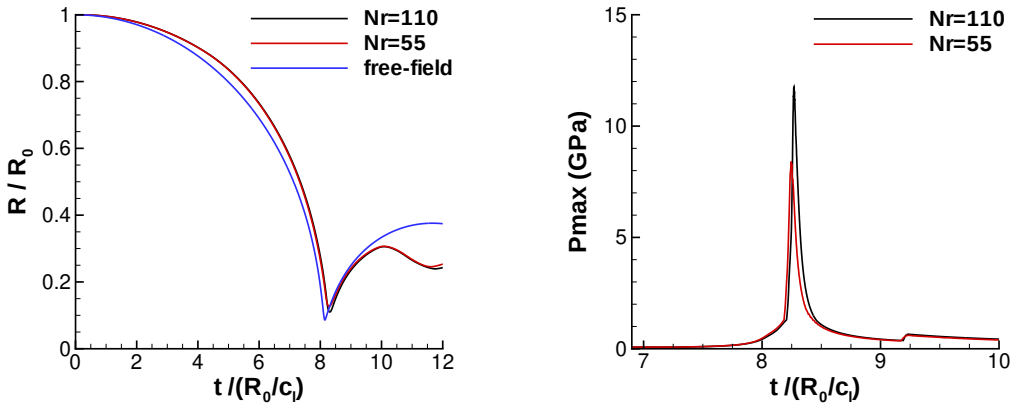


(a) $t^* = 5.1$ (b) $t^* = 8.4$

(c) $t^* = 10.1$ (d) $t^* = 12.1$

Figure 4: Visualization of the shock-induced collapse with $P_{sh}/P_{(b,0)} = 353$ at different times t^*. Longitudinal velocity component, wall pressure, Schlieren-like representation and isosurface of void ratio.

The time evolution of the dimensionless effective bubble radius R/R_0 is plotted in Figure 5 (left part) and compared with the Keller-Miksis solution obtained with the ratio $P_{l_\infty}/P_{(b,0)} = 353$. Of course, the analytical profile is given for illustrative purposes. The radius evolution is close to the inertial free-field case up to time $t^* = 10$. At this time, the blast wave impacted the wall and the reflected wave recollapses the bubble pieces.



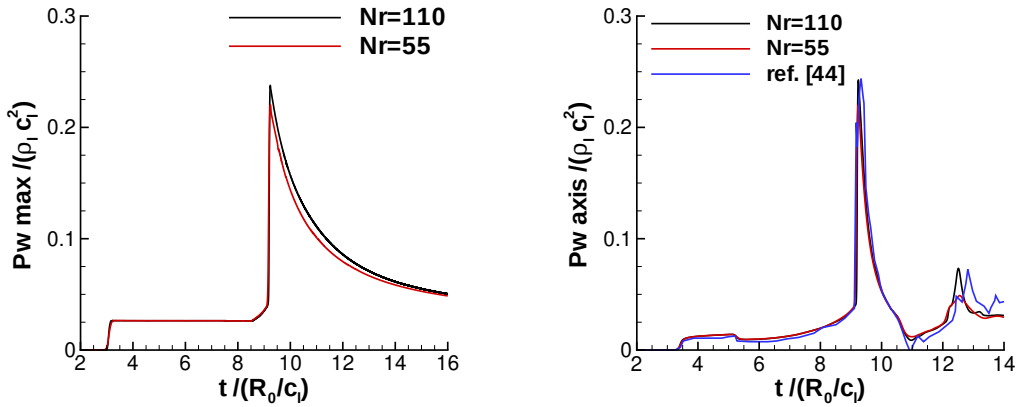
(a) (b)

Figure 5: Time evolution for the shock-induced collapse with $P_{sh}/P_{(b,0)} = 353$ of: a) the effective bubble radius R/R_0 compared with the free-field theory and b) the maximum pressure inside the liquid (in GPa).

The maximum pressure evolution, plotted on the right part of Figure 5, illustrates the high-pressure peak obtained when the water jet impacts the bubble interface generating the blast wave. The maximum pressure intensity is around 12 GPa, corresponding to 340 times the initial shock pressure P_{sh}.

The dimensionless maximum pressure along the wall is considered in Figure 6 (left part) as a function of time t^* for both meshes. The influence of the spatial distribution is weak for the wall pressure evolution. The first peak at time $t^* = 3$ is due to the impact of the incident shock wave on the wall. The most important peak, of about 0.65 GPa, is induced by the impact of the blast wave on the wall at time $t^* = 9.24$. In the study of Wermelinger et al., a peak intensity of 0.67 GPa obtained at time $t^* = 9.34$ was monitored. The present results are therefore in very good agreement with those of Wermelinger et al. As regard to the 2D results obtained by Johnsen and

Colonius, they showed a pressure peak $P/(\rho_l c_l^2)$ around 0.2. For the stand-off distance $H/R_0 = 2$, it highlights the fact that the wall pressure peak for the cylindrical collapse is close to the wall pressure value of the spherical case.



(a) (b)

Figure 6: Time evolution for the shock-induced collapse with $P_{sh}/P_{(b,0)} = 353$ of: a) the maximum wall pressure and b) the wall pressure along the axis.

The wall pressure evolution monitored on the bubble axis is plotted in Figure 6 (right part) and compared with the solution of ETH Zurich [46]. One can see that the results match very well the solution obtained in [46]. Small discrepancies are noticeable for the secondary waves impact after time $t^* = 12$.

6.3. Shock-induced bubble collapse near a solid wall, $P_{sh} = 1200$ bar

This subsection considers a stiffer case with a more intense incident shock wave. The test is similar to the one presented in [33]. It consists of simulating the collapse of a single gas bubble immersed in water near a solid wall and impacted by a normal shock wave for which the downstream pressure is $P_{sh} = 1200$ bar. The bubble with an initial bubble radius $R_0 = 0.05$ mm is immersed in water at temperature 298 K under the following initial conditions:

$$(\rho, \mathbf{u}, p) = \begin{cases} (1000\ \text{kg/m}^3, \mathbf{0}\ \text{m/s}, 10^5\ \text{Pa}) & \text{inside the liquid} \\ (1.176\ \text{kg/m}^3, \mathbf{0}\ \text{m/s}, 10^5\ \text{Pa}) & \text{inside the gas} \end{cases} \tag{18}$$

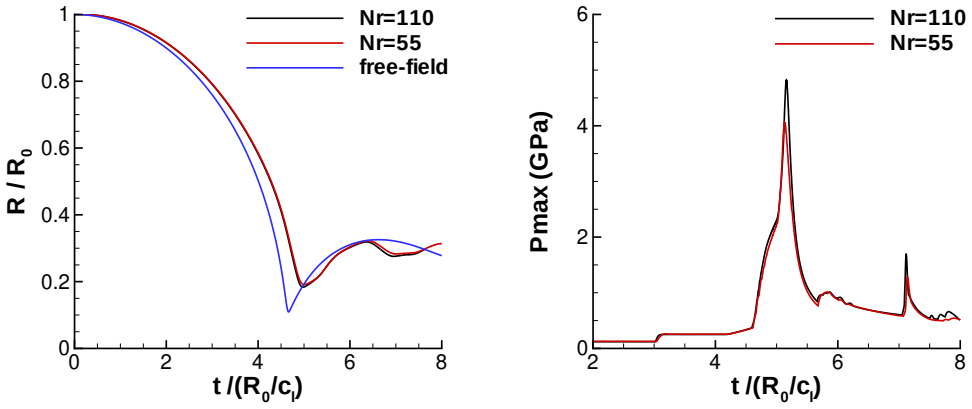
To close the problem, material parameters for the stiffened gas EOS are set to:

$$(\gamma_l, p_{\infty_l}, C_{p_l}) = (2.35, 10^9 \text{ Pa}, 5844 \text{ J/kg.K}), \text{ for the liquid phase} \tag{19}$$

$$(\gamma_g, p_{\infty_g}, C_{p_g}) = (1.4, 0 \text{ Pa}, 1009 \text{ J/kg.K}), \text{ for the gas phase.} \tag{20}$$

As previously, only a quarter of the bubble is being considered. The computational domain is $[L_x \times L_y \times L_z] = [8R_0 \times 12R_0 \times 12R_0]$ and the initial position of the bubble is $(x_{(b,0)}, y_{(b,0)}, z_{(b,0)}) = (6R_0, 0, 0)$. The incident shock wave normal to x-direction is initialized at position $x_{sh} = 4.85R_0$. As previously, $N_{r_{(b,0)}}$ represents the number of nodes per initial bubble radius. Two meshes have been simulated with $N_{r_{(b,0)}} = 55$ and 110, respectively. This study considers two values of the stand-off distance: $H/R_0 = 2$ and $H/R_0 = 1.4$.

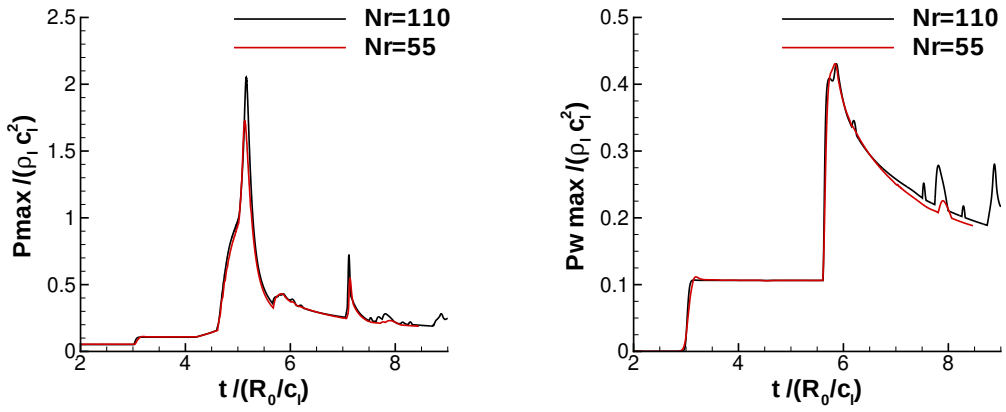
The time evolution of the effective bubble radius R/R_0 is plotted in Figure 7 (left part) for the stand-off distance $H/R_0 = 2$ and compared with the Keller-Miksis solution obtained with $P_{l_\infty}/P_{(b,0)} = 1200$. One can see that the minimum radius, close to 0.185, is higher than the previous case where it was close to 0.1. It is therefore expected a less intense pressure peak during the collapse due to the less important volume reduction. The evolution of the maximum pressure in the liquid, plotted in the right part of the figure, confirms this point. The peak intensity reaches 4.8 GPa, corresponding to 40 times the shock pressure P_{sh}, whereas it was around 12 GPa in the previous case. Interestingly, the water-hammer pressure decreases when the incident shock intensity increases. The inverse behaviour was depicted for the cylindrical collapse: it was observed an increase of the the water-hammer pressure with increasing the incident shock. Yet, proportionally with the incident shock intensity, the water-hammer shock is weakest at higher shock intensities, as commented by different authors [21, 19, 15].



(a) (b)

Figure 7: Time evolution for the shock-induced collapse with $P_{sh}/P_{(b,0)} = 1200$ and $H/R_0 = 2$ of: a) the effective bubble radius R/R_0 compared with the free-field theory and b) the maximum pressure inside the liquid (in GPa).

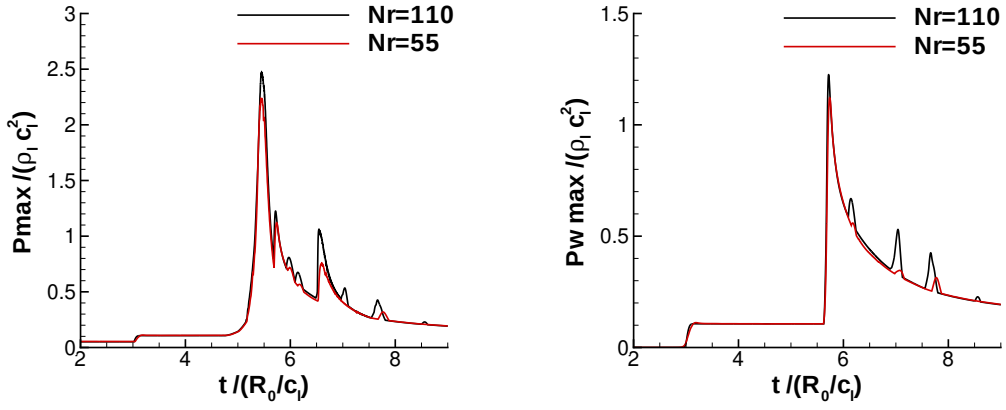
Figure 8 illustrates, for both meshes and for the ratio $H/R_0 = 2$ the time evolution of the maximum pressure in the liquid (on the left) and the maximum pressure along the wall (on the right). The main phenomena involved in the collapse are similar to the previous case. The most important peak at around time $t^* = 5.15$ corresponds to the blast wave formation when the liquid jet impacts the opposite bubble interface. The second peak at around time $t^* = 7$ is due to the recollapse of the bubble pieces by the reflected wave. At the wall, the first peak is due to the incident shock wave and the most intense peak is due to the impact of the generated blast wave. At a later time, other peaks emerge due to secondary waves generated by the recollapse of the bubble pieces. The influence of the mesh is clearly noticeable for the capture of these secondary waves.



(a) (b)

Figure 8: Time evolution for the shock-induced collapse with $P_{sh}/P_{(b,0)} = 1200$ and $H/R_0 = 2$ of: a) the dimensionless maximum pressure and b) the dimensionless maximum wall pressure.

The same quantities are plotted in Figure 9 for the stand-off distance $H/R_0 = 1.4$. A similar behaviour is observed.



(a) (b)

Figure 9: Time evolution for the shock-induced collapse with $P_{sh}/P_{(b,0)} = 1200$ and $H/R_0 = 1.4$ of: a) the dimensionless maximum pressure and b) the dimensionless maximum wall pressure.

As expected, closer is the bubble, the more intense is the pressure peak due to the blast wave impact. The magnitude of the most important pressure peak on the wall is reported in Table 5

for both values of the stand-off distance. Moreover, 2D results are added, obtained with the same 4-equation model and with a number of nodes per bubble radius equal to 150. The pressure is divided by the incident shock pressure P_{sh}. As previously commented, the maximum wall pressure peak obtained with a 2D simulation is fairly close to the 3D solution (the gap is about 7%) when the bubble is located at a distance $2R_0$ not too close to the wall. It is no more the case when the bubble becomes closer to the wall. For the situation where $H/R_0 = 1.4$, the pressure peak provided by the 3D simulation (which is 24 times the shock intensity) approximatively doubles compared to the 2D result.

Table 5: Maximum wall pressure during the shock-induced collapse with $P_{sh}/P_{(b,0)} = 1200$. Comparison between the 3D simulations and the 2D ones for $H/R_0 = 2$ and 1.4.

	stand-off distance $H/R_0 = 2$		$H/R_0 = 1.4$	
	3D simulation	2D simulation	3D simulation	2D simulation
maximum value of P_w/P_{sh}	8.43	7.83	24.0	11.96

7. Conclusion

This study presents a compressible two-phase flow solver devoted to compute strong shock-interface interactions. In the framework of diffuse interface methods, a four-equation model was used. The computational approach is based on a HLL-type Riemann solver discretized through a finite volume method. A particular attention is paid on the parallel strategy and the performance portability to fully exploit heterogeneous architectures. An hybrid well-suited MPI+OpenACC implementation was developped and a detailed performance study assessed the high potential of the proposed approaches. Results indicated that OpenACC paradigm was mature and provided a good efficiency on any kind of devices and therefore seems to be a relevant choice for hybrid implementation with MPI library. The present study validates the implementations on the benchmark of the bubble collapse in a free-field by comparison with the analytical solution of the Keller-Miksis equation. Then a shock-induced bubble collapse was computed and a very close agreement with existing results was illustrated. Finally, a strong collapse near a rigid wall was investigated for two positions of the bubble. A high pressure peak at the wall was predicted for the case where the bubble is closest to the wall. Moreover, results underlined the importance of performing 3D simulations for the prediction of the wall damage in comparison with 2D results.

These results highlighted the robustness of the solver and the ability of the proposed methods to compute efficiently challenging two-phase flows. Ongoing and future works are in progress to investigate multi-bubble collapse.

Acknowledgements

This research was supported by the French National Research Agency ANR (project 18-CE46-009). Computations have been performed on the supercomputer facilities of the the Mesocentre CRIANN (project 2018004) and the HPC resources of GENCI under allocations A0072A10981.

Author contributions

All of the authors were involved in the preparation of the manuscript and have read and approved the final manuscript version.

[1] R. Abgrall, *How to prevent pressure oscillations in multicomponent flow calculations: a quasi conservative approach*, J. Computational Physics, (1996), **125**, 150-160.

[2] G. Allaire, S. Clerc and S. Kokh *A five-equation model for the simulation of interfaces between compressible fluids*, J. Computational Physics, (2002), **181**, 577-616.

[3] G.J. Ball, B.P. Howell, T.G. Leighton and M.J. Schofield, *Shock-induced collapse of a cylindrical air cavity in water: a free-Lagrange simulation*, Shock Waves, (2000), **10**, 265-276.

[4] S. Beig and E. Johnsen, *Maintaining interface equilibrium conditions in compressible multiphase flows using interface capturing*, J. Computational Physics, (2015), **302**, 548-566.

[5] N.K. Bourne and J.E. Field, *Shock-induced collapse and luminescence by cavities*, Phil. Trans. R. Soc. Lond. A, (1999), **357**, 295-311.

[6] A. Brujan, G. S. Keen, A. Vogel and J. R. Blake, *The final stage of the collapse of a cavitation bubble close to a rigid boundary*, Phys. Fluids, (2002), **14**, 85.

[7] S.H. Bryngelson, K. Schmidmayer, V. Coralic, K. Maeda, J. Meng and T. Colonius, *MFC: An open-source high-order multi-component, multi-phase, and multi-scale compressible flow solver*, Computer Physics Communication, (2021), **266**, 107396.

[8] V. Coralic and T. Colonius, *Shock-induced collapse of a bubble inside a deformable vessel*, European Journal of Mechanics B/Fluids, (2013), **40**, 64-74.

[9] P. Costa, E. Phillips, L. Brandt and M. Fatica, *GPU acceleration of CaNS for massively-parallel direct numerical simulations of canonical fluid flows*, Computers and Mathematics with Applications, (2021), **81**, 502-511.

[10] F. Daude,P. Galon,Z. Gao and E. Blaud, *Numerical experiments using a HLLC- type scheme with ALE formulation for compressible two-phase flows five-equation models with phase transition*, Computers and Fluids, (2014), **94**, 112-138.

[11] F. Denner and B. van Wachem, *Numerical modelling of shock-bubble interactions using a pressure-based algorithm without Riemann solvers*, Experimental and Computational Multiphase Flow, (2019), **1**(4), 271-285.

[12] J.M. Diaz, G. Jost, S. Chandrasekaran and S. Pino, *Is OpenMP 4.5 target offload ready for real life ?*, SuperComputing OpenMP booth talks, SC18 Dallas, (2018).

[13] E. Goncalves, *Numerical study of expansion tube problems: toward the simulation of cavitation*, Computers and Fluids, (2013), **72**, 1-19.

[14] E. Goncalves and B. Charriere, *Modelling for isothermal cavitation with a four-equation model*, Int. J. Multiphase Flow, (2014), **59**, 54-72.

[15] E. Goncalves , Y. Hoarau and D. Zeidan, *Simulation of shock-induced bubble collapse using a four-equation model*, Shock Waves, (2019), **29**, 221-234.

[16] E. Goncalves and P. Parnaudeau, SCB: An efficient and simple parallel code to simulate a 3D shock-induced bubble collapse. IUTAM Symposium on Computational Modelling of Instabilities and Turbulence in Separated Two-Phase Flows, June 10-12, Dublin, 2019.

[17] E. Goncalves and P. Parnaudeau, *Comparison of multiphase models for computing shock-induced bubble collapse*, Int. J. Numerical Methods for Heat and Fluid Flow, (2020), **22**, 3845-3877.

[18] K. Kozhanova, E. Goncalves and Y. Hoarau, *High-order numerical methods for compressible two-phase flows*, Finite Volumes for Complex Applications, FCVA-9, June 15-19, Bergen, Norway, 2020. Springer Proceedings in Mathematics and Statistics, **323**, 685-693.

[19] N. Hawker and Y. Ventikos, *Interaction of a strong shockwave with a gas bubble in a liquid mediam: a numerical study*, J. Fluid Mechanics, (2012), **701**, 59-97.

[20] *HPE SGI 8600 (Jean Zay)*, (2019), http://www.idris.fr/eng/jean-zay/cpu/jean-zay-cpu-hw-eng.html

[21] E. Johnsen and T. Colonius, *Numerical simulations of non-spherical bubble collapse*, J. Fluid Mechanics, (2009), **629**, 231-262.

[22] S. Kamil, C. Chan, L. Oliker, J. Shalf and S. Williams, *An auto-tuning framework for parallel multicore stencil computations*, IEEE International Symposium on Parallel & Distributed Processing, (2010).

[23] A. Kapila, R. Menikoff, J. Bdzil, S. Son and D. Stewart, *Two-phase modeling of deflagration to detonation transition in granular materials: reduced equations*, Physics of Fluids, (2001), **13**, 3002-3024.

[24] J.B. Keller and M. Miksis, *Bubble oscillations of a large amplitude*, J. Acous. Soc. Am., (1980), **68**, 628-633.

[25] E. Lauer, X.Y. Hu, S. Hickel and N.A. Adams, *Numerical investigation of collapsing cavity arrays*, Physics of Fluids, (2012), **24**, 052104.

[26] X. Li and P-C. Shih, *An early performance comparison of Cuda and OpenACC*, MATEC Web of Conferences, ICMIE 2018, (2018), **208**.

[27] OS. Majidi and A. Afshari *An adaptive interface sharpening methodology for compressible multiphase flows*, Comput. Math. Applications, (2016), **72**, 2660-2684.

[28] O.L. Metayer, J. Massoni and R. Saurel, *Elaborating equations of state of a liquid and its vapor for two-phase flow models*, Int. J. Thermal Sciences, (2004), **43**, 265-276.

[29] S. Muller, P. Helluy and J. Ballmann, *Numerical simulation of a single bubble by compressible two-phase fluids*, Int. J. Numerical Methods Fluids, (2010), **62**(6), 591-631.

[30] A. Murrone and H. Guillard, *A five equation reduced model for compressible two phase flows problems*, J. Computational Physics, (2005), **202**, 664-698.

[31] R. Nourgaliev, T. Dinh and T. Theofanous, *Adaptive characteristics-based matching for compressible multifluid*

dynamics, J. Computational Physics, (2006), **213**, 500-529.

[32] C. Obrecht, F. Kuznik, B. Tourancheau and J-J. Roux, *A new approach to the lattice Boltzmann method for graphics processing units*, Computers & Mathematics with Applications, (2011), **61**, 3628-3638.

[33] Y. Paquette, M. Fivel, G. Ghigliotti, E. Johnsen and J-P. Franc, *Fluid-Structure Interaction in Cavitation Erosion*, (2018), 10th International Symposium on Cavitation CAV2018, Baltimore, USA.

[34] A. Philipp and W. Lauterborn, *Cavitation erosion by single laser-produced bubbles*, J. Fluid Mechanics, (1998), **361**, 75-116.

[35] D. Rossinelli, B. Hejazialhosseini, P. Hadjidoukas, C. Bekas, A. Curioni, A. Bertsch, S. Futral, S.J. Schmidt,N.A. Adams and P. Koumoutsakos, *11 PFLOP/s simulations of cloud cavitation collapse*, SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Denver, CO, (2013), 1-13.

[36] K. Schmidmayer, F. Petitpas, S. Lemartelot and E. Daniel, *ECOGEN: An open-source tool for multiphase, compressible, multiphysics flows*, Computer Physics Communication, (2019), **251**, 107093.

[37] K. Schmidmayer, S.H. Bryngelson and T. Colonius, *An assessment of multicomponent flow models and interface capturing schemes for spherical bubble dynamics*, J. Computational Physics, (2020), **402**, 109080.

[38] R.K. Shukla, C. Pantano and J.B. Freund, *An interface capturing method for the simulation of multi-phase compressible flows*, J. Computational Physics, (2010), **229**, 7411-7439.

[39] O. Supponen, D. Obreschkow, P. Kobel, M. Tinguely, N. Dorsaz and M. Farhat, *Shock waves from nonspherical cavitation bubbles*, Physical Review Fluids, (2017), **2**, 093601.

[40] H. Terashima and G. Tryggvason, *A front-tracking/ghost-fluid method for fluid interfaces in compressible flows*, J. Computational Physics, (2009), **228**, 4012-4037.

[41] A. Tiwari, C. Pantano and J.B. Freund, *Growth-and-collapse dynamics of small bubble clusters near a wall*, J. Fluid Mechanics, (2015), **775**, 1-23.

[42] TOP 500 The list. http://top500.org/

[43] E. Toro, M. Spruce and W. Speares, *Restoration of the contact surface in the HLL-Riemann solver*, Shock Waves, (1994), **4**, 25-34.

[44] B. van Leer, *On the relation between the upwind-differencing schemes of Godunov, Engquist–Osher and Roe*, SIAM J. Sci. Stat. Comput., (1984), **5**(1), 1-20.

[45] G. Wallis, *One-Dimensional Two-Phase Flow*, McGraw-Hill, (1967), New York, NY.

[46] F. Wermelinger, U. Rasthofer, P.E. Hadjidoukas and P. Koumoutsakos, *Petascale simulations of compressible flows with interfaces*, J. Computational Science, (2018), **26**, 217-225.

[47] S. Williams, A. Waterman and D. Patterson, *Roofline: An insightful visual performance model for floating-point programs and multicore architectures*, Communications of ACM, (2009), **52**, issue 4.

[48] Xia Y, Lou J, Luo H, Edwards J, Mueller F. OpenACC acceleration of an unstructured CFD solver based on a reconstructed discontinuous Galerkin method for compressible flows. *Int. J. Numerical Methods Fluids*, (2015), **78**, 123-139.

[49] G. Xiang and B. Wang, *Numerical study of a planar shock interacting with a cylindrical water column embedded with an air cavity*. J. Fluid Mechanics, (2017), **825**, 825-852.