Parallélisation d'un code SPH 3D pour des simulations massives sur mémoire distribuée

Guillaume Oger, David Guibert, Matthieu de Leffe, Jean-Guillaume Piccinali

▶ **To cite this version:**

Guillaume Oger, David Guibert, Matthieu de Leffe, Jean-Guillaume Piccinali. Parallélisation d'un code SPH 3D pour des simulations massives sur mémoire distribuée. CFM 2013 - 21ème Congrès Français de Mécanique, Aug 2013, Bordeaux, France. ⟨hal-03439761⟩

HAL Id: hal-03439761

https://hal.science/hal-03439761v1

Submitted on 22 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Parallelization of a 3-D SPH code for massive HPC simulations

G. Oger[a], D. Guibert[b], M. de Leffe[b], J.-G. Piccinali[c]

a. *LHEEA Ecole Centrale de Nantes*
b. *Hydrocean*
c. *CSCS-Swiss National Supercomputing Centre*

Résumé :

La méthode Smoothed Particle Hydrodynamics (SPH) est une méthode particulaire qui a connu une forte émergence au cours des deux dernières décennies. Bien qu'initialement développée pour des applications astrophysiques, cette méthode numérique est aujourd'hui largement appliquée à la mécanique des fluides, à la mécanique des structures et à diverses branches de la physique. Le code SPH-flow est développé conjointement par le LHEEA de l'Ecole Centrale de Nantes et l'entreprise HydrOcean. Cet outil est principalement dédié à la modélisation d'écoulements à surface libre, dans un contexte de dynamique rapide et en présence de solides présentant des géométries complexes en interaction avec le fluide. Dans ce domaine, le principal avantage de cette méthode repose sur sa capacité à simuler les déconnexions/reconnexions de surface libre (déferlements, jets de surface libre etc...) sans nécessiter sa capture. Comme c'est le cas pour la plupart des autres méthodes particulaires, cette méthode est exigeante en termes de ressources de calcul, et sa parallélisation est inévitable dans le cadre d'applications 3D massives, en vue de conserver des temps de restitution raisonnables. L'ordre de grandeur des résolutions adoptées dans nos simulations est de plusieurs centaines de millions de particules, impliquant l'utilisation de plusieurs milliers de processeurs en réseau, et donc une parallélisation performante. Cet article présente la stratégie de parallélisation retenue dans notre code SPH, basée sur une décomposition de domaine et passant par l'utilisation du standard MPI (mémoire distribuée). Les performances parallèles sont présentées en termes d'accélération et d'efficacité, sur des cas allant jusqu'à 3 milliards de particules et utilisant 32768 processeurs en réseau.

Abstract :

Smoothed Particle Hydrodynamics (SPH) is a particle method that experienced a large growth in the last two decades. While it was initially developed for astrophysics, today this method is applied to fluid, structures and other complex multiphysic simulations. The SPH-flow code developed jointly by Ecole Centrale de Nantes and HydrOcean is mainly designed to model complex free surface problems in a context of high dynamic phenomena, together with the presence of complex solid geometries interacting with the fluid. In this field, the main advantage of the SPH method relies on its ability to simulate disconnection and reconnection of the free surface without having to capture it. As for other particle methods, this method is demanding in terms of computational resources, and its parallelization is compulsory for large 3-D applications, in order to maintain some reasonable restitution times. The order of magnitude of the resolution involved in our simulations is several hundred million particles, implying the need for thousands of CPU cores. This paper deals with introducing a parallelization strategy based on a domain decomposition within a purely MPI-based distributed memory framework. The results are discussed through a scalability study involving up to $32,768$ *cores with* 3 *billion particles.*

Mots clefs : SPH ; HPC ; MPI

1 Introduction

The objective of the present work is to improve the performance of the HPC code SPH-flow developed jointly by Ecole Centrale de Nantes and HydrOcean, to make it efficient in research as well as in industrial contexts. Various problems can be solved using this 2-D/3-D parallel SPH code, such as multifuid flows [4], Fluid-Structure Interaction (FSI) [3], and viscous flows [5]. The main difficulties of a SPH model parallelization arise from the interpolation process, based on a kernel function and using possibly variable compact supports. In Section 2, the SPH method is described, together with its kernel-based interpolation feature. In Section 3, the main aspects of the parallelisation performed are introduced. In Section 4, parallel performances are presented and discussed. Finally, industrial test cases involving massively parallel SPH simulations are presented in Section 5.

2 SPH method

The equations to be solved in our field of application are the Euler equations, as classically used to model non viscous fluid flows. One of the main SPH features consists in considering any fluid flow as compressible, resulting in the use of an equation of state to close the system. SPH uses a set of interpolating points (particles) which are initially distributed in the fluid medium. The spatial derivatives present in the Euler equations are then simply interpolated by introducing a kernel function W that is convoluted with the discrete values of the field, as in the following example for the pressure gradient :

$$< \nabla P(\vec{x}) >= \sum_{i=1}^{N} P(\vec{x_i}) \nabla W(\vec{x} - \vec{x_i}, R)\, \omega_i, \tag{1}$$

where i refers to the particles in the vicinity of location $\vec{x}$ and located within the radius R of the compact support of W. Note that this kernel function W implies the same algorithmic needs as the Van der Waals function used in molecular dynamics, requiring the creation of a neighbor list and the computation of particle-to-particle interactions. The main difference resides in the fact that W acts as an actual interpolation function, similarly to the test functions used in Finite Element methods for instance. In their discrete form, the Euler equations become finally

$$\frac{d\vec{x_i}}{dt} = \vec{v_i}, \tag{2}$$

$$\frac{d\rho_i}{dt} = \sum_{j=1}^{N} m_j\, (\vec{v_i} - \vec{v_j})\, \vec{\nabla} W(\vec{r_i} - \vec{r_j}, R), \tag{3}$$

$$\frac{d\vec{v_i}}{dt} = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \vec{\nabla} W(\vec{r_i} - \vec{r_j}, R) + \vec{g}, \tag{4}$$

Note that the above scheme corresponds to one of the various examples of SPH formulations available in the literature. As stated in equation (2), each particle moves according to the computed fluid velocity, emphasizing the Lagrangian feature of SPH. Moreover, each of the above derivatives are advanced in time using an explicit time integration scheme such as Runge-Kutta, preventing from the need of any linear system resolution. This explicit and Lagrangian features of SPH stand for the main properties that impact the parallelization of this method. Furthermore, in practice at least $N = 20$ and $N = 70$ neighbors are needed in 2-D and 3-D respectively for the interpolation of each particle i, so that the main computational costs rely in the computation of the flux terms of equations (3) and (4).

3 Parallel flux computations and time advance

The parallelization performed here is based on a domain decomposition strategy, which consists in splitting the whole particle domain into sub-domains, and to attribute each sub-domain to each process. The whole domain is therefore "retrieved" through point-to-point MPI communications between processors. In [6], we presented parallel results of computations on up to $2,000$ processors. The results were encouraging but exhibited a breakdown, revealing that the communications were not fully overlapped by the computation of the flux terms and/or the time advance. In order to overcome this problem, an overlapped zone is identified. For a given process of interest, this zone is made of three particle groups : the outer particles which are received from the neighbor processes, the inner particles which are sent to the neighbor processes, and the local particles which are neither outer nor inner particles. Local particles are usually more numerous than inner particles. The strategy retained here thus consists in overlapping the communication times of inner particles with the computation times of local particle flux loops. The main difficulty then resides in the creation of such a Local-Inner-Outer zone, which is not straightforward due to the need for an easy identification of the outer particles in the context of variable kernel radii discretisation. To avoid huge communications of particles that do not interact with inner particles, the shape of the overlapped zone must be built piecewise. Finally, time integration of ODEs (3) and (4) can be summarized as in equation (5) :

$$\phi_i^{n+1} = \phi_i^n + \delta t \sum_j F_{ij} \qquad \forall i \text{ particles}, \tag{5}$$

where ϕ_i^n is the variables at time t_n of particle i, and F_{ij} the flux between particles i and j. The parallel scheme is evaluated as described in Algorithm 1. This scheme ensures that a high parallelisation efficiency can be reached provided that the computational cost of the local particle interaction loops is greater than the communication time of the inner particles.

Algorithm 1 LIO scheme with communication overlapping.

Require: the state variables ϕ^n on the local, inner and outer particles should be first correctly addressed for all proceses.

1: post the MPI reception of the newly updated state variables ϕ^{n+1} of outer particles.
2: **for all** interaction of the inner particles with their local, inner and outer neighbor particles **do**
3: compute the flux terms F_{ij}
4: **end for**
5: **for all** inner particles i **do**
6: update ϕ_i^{n+1} as in equation (5).
7: **end for**
8: MPI send of the newly computed state variables of inner particles to neighbor processes
9: **for all** interactions of the local particles with their local neighbor particles **do**
10: compute the flux terms F_{ij}
11: **end for**
12: **for all** local particles i **do**
13: update ϕ_i^{n+1} as in equation (5).
14: **end for**
15: wait the communication ending of the outer particle state variables.

4 Scalability study in massive HPC context

To study the parallel performances of the two approaches combined, two distinct studies can be performed. The first one consider a fixed problem size while the number of processes increases. Ideally the reduction factor of the CPU time on p processes should be $\frac{1}{p}$, but we generally introduce the speedup which is the inverse of this factor. Such a study is called "strong scalability" study, and is presented in Subsection 4.1. Another study may consider a fixed problem size per process. In this case, the communications increase as the number of processes increase. This study named "weak scalability"

study enables to identify some MPI communications bottlenecks. The results are exposed in Subsection 4.2. The study presented hereafter have been performed on the facilities provided by CSCS. The code has been executed on Monte Rosa, fitted with AMD Interlagos 2 x 16-core 64-bit CPUs, 32 GB per compute node and high performance networking with Gemini 3D torus interconnect. It features a total of $1,496$ nodes, corresponding to $47,872$ cores, with a theoretical peak performance of 402 TFlops.

4.1 Strong scalability

Figure 1 shows the results obtained for a total of 10^7 particles from 8 to 256 cores, for 10^8 particles from 256 to $16,384$ cores and 10^9 particles from $4,096$ to $32,768$ cores.

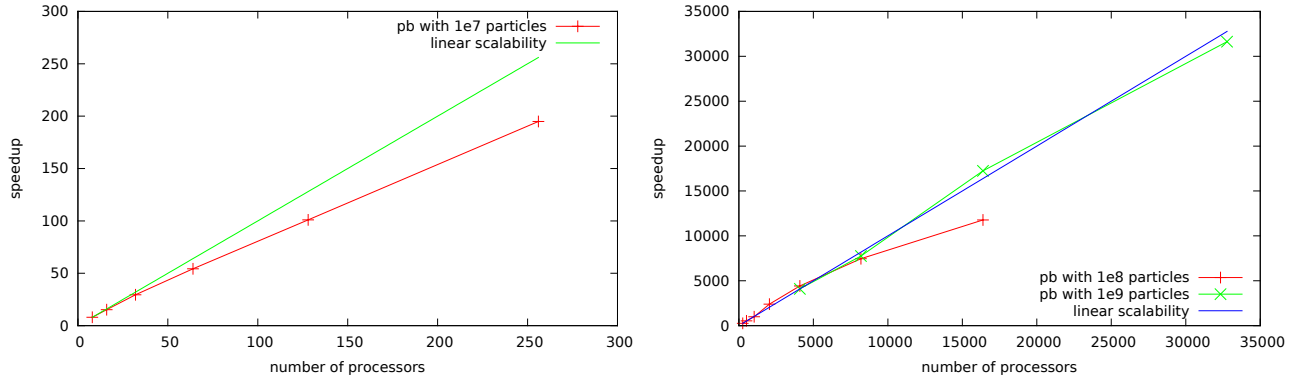

FIGURE 1 – Speedup obtained on a dam break test case with 10^7, 10^8 and 10^9 particles

Figure 1 shows that the code scales linearly but with a slope that remains lower than the ideal speedup. Due to the nature of our implementation, the boundaries are taken into account with some specific treatments, which implies that some extra interactions must be computed. These treatments suffer from some defects in terms of parallelization. As a result, as the proportion of near-boundary particles decreases while the number of particles increases, this defect tends to present a lower relative importance for very large simulations. For 10^8 particles, the speedup tends to start saturating from $16,384$ processors. For the simulation involving 1 billion particles, a linear speedup is obtained, even when $32,768$ cores are used.

4.2 Weak scalability

The weak scalability study is a way to define as how the time to solution varies with the number of processors for a fixed problem size per processor. Note that a fixed number of numerical time steps is applied for each case.

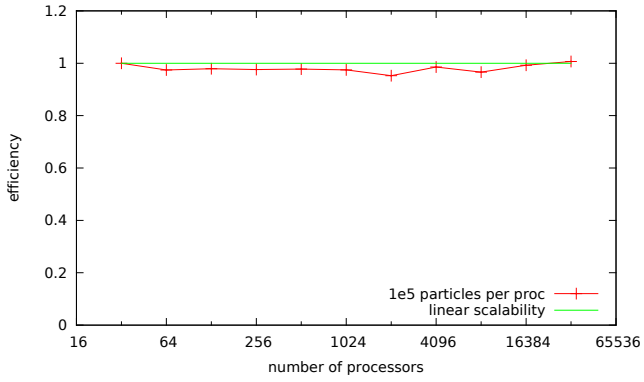

FIGURE 2 – Efficiency on the dam break test case with 10^5 particles per processes.

Monte Rosa is fitted with 32 cores per node. As a result, in order to avoid bus bandwidth bottlenecks, the weak scalability is started from a reference of 32 processors. Figure 2 shows that the efficiency obtained is nearby 95%. There is no performance decrease, showing that the communications are therefore fully overlapped by the computations.

5 Industrial test cases

Two distinct applications of the SPH-flow code are presented in this part with the improvements described above. These 3-D applications require a large number of points and therefore an efficient HPC code.

5.1 Sphere impact

This test case consists of the 3-D impact of a sphere at the free surface. The sphere of radius $R = 1m$ is moving at a constant imposed vertical velocity $U = -1m/s$. The slamming coefficient is defined as $C_S = \frac{F_Z}{\frac{1}{2}\rho U^2 \pi R^2}$. A convergence study is performed here for three different resolutions. The particle sizes in the impact area for the different meshes are $2 \times 10^{-2}m$, $1 \times 10^{-2}m$ and $5 \times 10^{-3}m$.

Δx (m)	N particles	N CPU	CPU time (h)
2×10^{-2}	1×10^6	32	3.2
1×10^{-2}	4.4×10^6	64	15.1
5×10^{-3}	24.1×10^6	256	45

TABLE 1 – CPU datas of the sphere impact.

A snapshot of the free surface deformations is given in figure 3, and CPU data are summarized in table 1. Figure 3 shows that the results obtained converge towards the solution given by Baldwin & Steves [1] and Battistin & Iafrati [2].

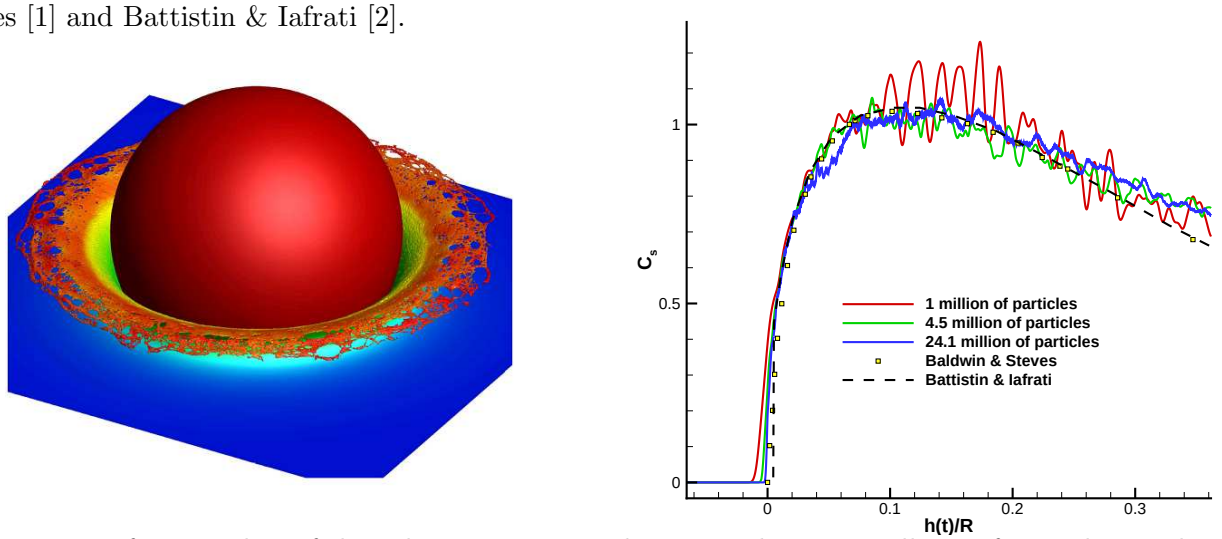

FIGURE 3 – Left : snapshot of the sphere impact simulation involving 25 millions of particles. Right : convergence study related to the sphere impact.

5.2 Lifeboat water entry

This case consists in the free fall of a lifeboat in calm water. A correct assessment of loads and slamming pressures on a lifeboat during its entry in the sea is essential for both structural and human safety. The present simulation has been performed for three resolutions : 1, 10 and 100 millions of particles. CPU data are summarized in table 2.

N particles	N CPU	CPU time (h)
1×10^6	64	3
10×10^6	512	5
100×10^6	2048	30

TABLE 2 – CPU datas of the lifeboat free fall.

Figure 4 shows the comparison for the three resolutions. As expected, the free surface jets created by the lifeboat water entry are captured more and more accurately.

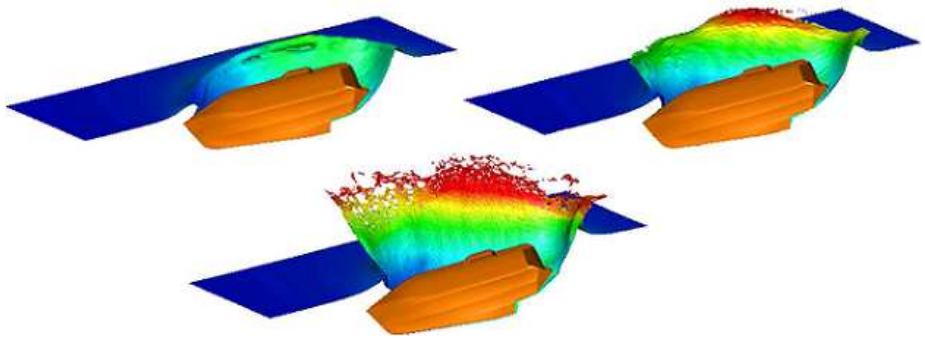

FIGURE 4 – Lifeboat impact with 1, 10 and 100 millions of particles.

6 Conclusion

A strategy to carry out 2-D and 3-D massively parallel SPH simulations has been presented. This has been achieved by introducing a domain decomposition framework with effective non-blocking communications. Very good scalability has been obtained for billion particle problems on several thousands of processors. In order to further improve the parallel performances, a particular attention sould be paid to the implementation. All the duplicated instructions between processes should be avoided, especially the boundary handling. A future way to reduce the simulation run-times would be to combine our approach with GPGPU approaches, in order to handle flux computations on graphic processors.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement #225967 "NextMuSE". This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID s290 and the authors would like to thanks the CSCS Staff, especially Jerome Soumagne and John Biddiscombe. Hydrocean and ECN acknowledge the computing resources offered by CCIPL, IFREMER and CRIHAN, their help was very much appreciated.

Références

[1] L. Baldwin, H.X. Steves *Vertical water entry of spheres*, NSWC/WOL/TR 75-49, White Oax Laboratory, Silver Spring, MD, USA, 1975

[2] D. Battistin, A. Iafrati *Water impact of 2D and axisymmetric bodies of arbitrary section*, INSEAN Technical Report No. 06/01, Roma, 2001

[3] G. Fourey, D. Le Touzé, B. Alessandrini *Three-dimensional validation of a SPH-FEM coupling method*, 6^{th} SPHERIC workshop Proceedings, 2011

[4] P.-M. Guilcher, G. Oger, L. Brosset, E. Jacquin, N. Grenier, D. Le Touzé *Simulations of liquid impacts with a two-phase parallel SPH model*, Proceedings of ISOPE, 2010

[5] M. de Leffe, D. Le Touzé, B. Alessandrini *A modified no-slip condition in weakly-compressible SPH*, 6^{th} SPHERIC workshop Proceedings, 2011

[6] P. Maruzewski, G. Oger, D. Le Touzé, J. Biddisccombe *High performance computing 3D SPH model : Sphere impacting the free-surface of water*, 3nd International SPHERIC Workshop, 2008