II International Conference on Particle-based Methods – Fundamentals and Applications
PARTICLES 2011
E. Oñate and D.R.J. Owen (Eds)

SMOOTHED PARTICLE HYDRODYNAMICS ON GPU COMPUTING

A.J.C. CRESPO[*], J.M. DOMINGUEZ[*], D. VALDEZ-BALDERAS[†], B.D. ROGERS[†] AND M. GOMEZ-GESTEIRA[*]

[*] Environmental Physics Laboratory (EPHYSLAB)
Universidade de Vigo
Campus As Lagoas s/n, 32004, Ourense, Spain
e-mail: alexbexe@uvigo.es, web page: http://ephyslab.uvigo.es

[†] School of Mechanical, Aerospace, & Civil Engineering (MACE)
University of Manchester
Sackville Street, Manchester, M60 1QD, United Kingdom
web page: http://www.mace.manchester.ac.uk/

Key words: CFD, GPU, CUDA, SPH, fluid simulations.

Abstract. Smoothed Particle Hydrodynamics (SPH) is a powerful technique used to simulate complex free-surface flows. However one of the main drawbacks of this method is the expensive computational runtime and the large number of particles needed when 3D simulations are performed. High Performance Computing (HPC) therefore becomes essential to accelerate these codes and perform simulations. In this study, parallelization using Graphics Processing Units (GPU) is applied to the SPHysics code (www.sphysics.org) dedicated to free-surface flows with SPH. Simulations involving several million particles on a single GPU exhibit speedups of up to two orders of magnitude over the same calculations using CPU codes, while parallelization using MPI for multi-GPU leads to further acceleration. This cheap technology allows studying real-life engineering problems at reasonable computational runtimes.

1 INTRODUCTION

Smoothed Particle Hydrodynamics (SPH) is a purely Lagrangian method developed during seventies (Gingold and Monaghan, 1977) in astrophysics. It has since been developed and applied to a range of engineering flows, in particular for-free surface hydrodynamics problems, such as the study of violent flows, wave breaking, wave-structure interactions.

SPHysics is an SPH numerical model developed to study free-surface flows and is the product of a collaborative effort amongst researchers at the Johns Hopkins University (U.S.A.), the University of Vigo (Spain) and the University of Manchester (U.K.). The open-source code, written in FORTRAN, is available to download for public use at www.sphysics.org. Although the SPH method can provide a fine description of the flow, its main drawback is its high computational cost, so that applying over large domains is prohibitive. Graphics Processing Units (GPUs) are a new technology imported from the computer games industry that can be used for scientific computing and is ideal for SPH due to its parallel architecture. As a result, the dual functioning CPU-GPU code *DualSPHysics* has been developed using C++ and Compute Unified Device Architecture (CUDA) for operation on CPUs and GPUs, respectively. More information about the *DualSPHysics* project can be found at www.dual.sphysics.org while different applications and animations can be viewed at www.vimeo.com/dualsphysics.

In the present work, the *DualSPHysics* solver is presented describing the different parallel codes implemented for different cores of CPU and one or more GPUs. The numerical results are firstly validated with experimental data to show the accuracy and reliability of our scheme, and then the achieved speedups comparing CPU and GPU are addressed to prove the efficiency of this new technology in CFD problems.

2 SPH BACKGROUND

SPH is a meshless method that describes a fluid by replacing its continuum properties with locally smoothed quantities at discrete Lagrangian locations and then integrates in time the hydrodynamic equations of motion for each particle in the Lagrangian frame. Relevant physical quantities are computed for each particle as an interpolation of the values of the nearest neighbouring particles, and then particles move according to those values. The conservation laws of continuum fluid dynamics, in the form of differential equations, are transformed into their particle forms by integral equations through the use of an interpolation function that gives the kernel estimate of the field variables at a point. SPH offers distinct advantages including no fixed computational grid being required when calculating spatial derivatives.

The main features of the SPH method, which is based on integral interpolants, are described in detail in Gómez-Gesteira et al., 2010 and Liu and Liu, 2010. Here, only the main points about implementation will be described. Conceptually, an SPH code is an iterative process consisting of three main steps:

a) *neighbour list*: particles only interact with surrounding particles located at a given distance so the domain is divided in cells of the kernel size to reduce the neighbour search to the adjacent cells;

b) *particle interaction*: each particle only looks for neighbours at the adjacent cells, after verifying that the distance between particles lies within the support of the kernel, the conservation laws of continuum fluid dynamics are computed for the pair-wise interaction of particles;

c) *system update*: once the forces between neighbouring particles have been evaluated, all physical magnitudes of the particles are updated at the next time step.

3 PARALLEL IMPLEMENTATION OF SPH ON CPU AND GPU

As mentioned, the SPH code is an iterative process where force interactions are computed for all particles and all physical quantities are updated at the following time step. All these tasks are very expensive in terms of computation time when the execution is carried out in a single serial machine, so the parallelization of the tasks for large number of particles becomes imperative. In the particular case of the SPHysics code different parallel techniques have been implemented on different devices, threads of CPU and GPU cards.

3.1 Multi-core implementation using OpenMP

OpenMP, a specification for parallel programming is used to implement the multi-core SPH code. Its implementation is straightforward and no significant changes in comparison to the single-core code are required. Most of the sequential tasks and operations that involve a loop over all particles are performed using the different cores of the same CPU. Thus, the time dedicated to communication between different execution threads is reduced since the same shared memory is used. The code is also optimised with the implementation of dynamic load balancing. As is well known, using OpenMP on its own means that this parallelization and potential speedup are limited to a small number of cores (i.e. the number of cores existing on the compute node)

3.2 GPU implementation.

The GPU parallelisation technique uses the CUDA developed by nVidia. An efficient and full use of the capabilities of the GPU architecture is not straightforward. In this case, the sequential tasks over the particles are performed using different execution threads of the GPU architecture. For example, with a GTX480 card a maximum of 23,040 threads can be executed simultaneously (15 multiprocessors and 1,536 threads per multiprocessor as maximum). The most efficient option is to keep all data in the memory of the GPU where the three main processes of SPH are executed in parallel. The neighbour list follows the procedure used on CPU and some tasks in parallel are even improved by using the optimised *radixsort* algorithm provided by CUDA. Updating the physical quantities of the particles is easily parallelized. The particle interaction process, which is the most expensive computationally, is implemented solely on the GPU using one execution thread to compute the particle interaction of only one particle. The thread looks for the neighbours of each particle among the adjacent cells and computes the resulting force from all the interactions. Due to the Lagrangian nature of the method, different problems appear such as lack of balancing, code divergence and no perfect coalescent access to the global memory of the device.

3.3 Multi-GPU implementation

Since the memory requirements are still a limitation for a single GPU, using more than one GPU appears to be the best development to continue accelerating SPH simulations. In order to allow different devices communicating with each other, the Message Passing Interface (MPI) is used jointly with CUDA to implement a multi-GPU version of *DualSPHysics*. MPI presents the advantage of using different compute nodes hosting multiple devices instead of only one as it happens with OpenMP. The multi-GPU implementation consists of assigning different portions of the physical system to different GPUs (Valdez-Balderas et al., 2011). After each computation step, data needs to be transferred between devices such as the information of particles that migrate between GPUs (physical sub-domains) or particles that belong to shared spaces where data is used by several GPUs. With the CUDA v4.0, direct GPU-GPU communication is only very recently supported, so the memory transfer between different GPUs is carried out by GPU-CPU, CPU-CPU and CPU-GPU communications which is proving to be 85% efficient without the use of high-speed Infiniband connections to be used in the future.

4 RESULTS

The *DualSPHysics* code is validated with the same experiment used for the SPHysics validation in Gómez-Gesteira and Dalrymple, (2004) to demonstrate the reliability of the GPU implementation. Thus, model results were compared to experimental data provided by Yeh and Petroff (Chen et al., 1997)) at the University of Washington where a dam break occurred within a rectangular tank, with a volume of water initially contained behind a thin gate at one end of the box and a tall structure was inside the tank. Experimental measurements included the time history of the net force on the structure and the time history of the fluid velocity at a location just in front of the structure. Figure 1 shows different instants of this simulation using one million particles.

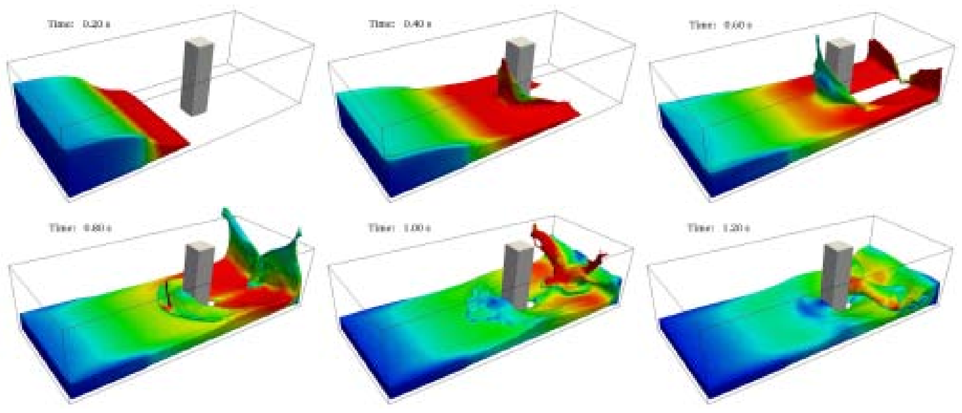

Figure 1: Dam breaking flow impacting on a structure with SPH. Colour represents velocity values.

Figure 2 shows the close agreement between numerical velocity (red line), numerical force (green line) and the experimental values (blue points). The SPH model is able to reproduce the experimental velocity field and the forces generated by the collision between the incoming wave and the structure although the maximum impact force is underpredicted which is most likely due to either the SPH formulation or boundary conditions.

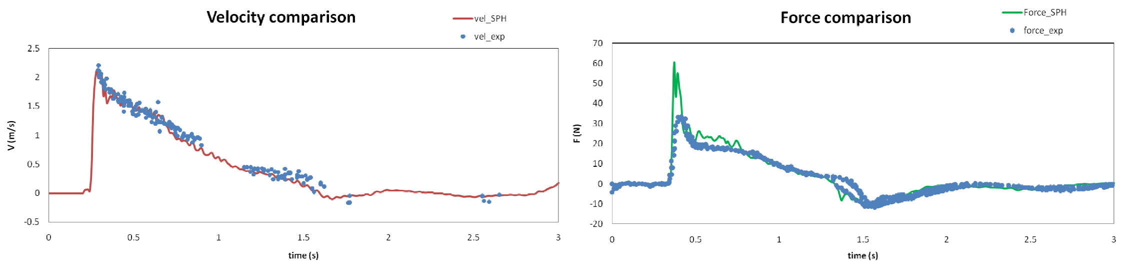

Figure 2: Comparison between SPH results and experimental data.

A performance analysis has been carried out for this test case varying the number of particles (N_p) and using schemes introduced in Section 3. The codes are executed on the CPU Intel® Core ™ i7 940 at 2.93GHz and on the GPU GTX 480 at 1.40GHz. Figure 3 shows the computational runtimes and the number of steps computed per second for the different simulations. Note that *N*GPU means using *N* GPU cards while *N*CPU means running on *N* cores of the CPU.

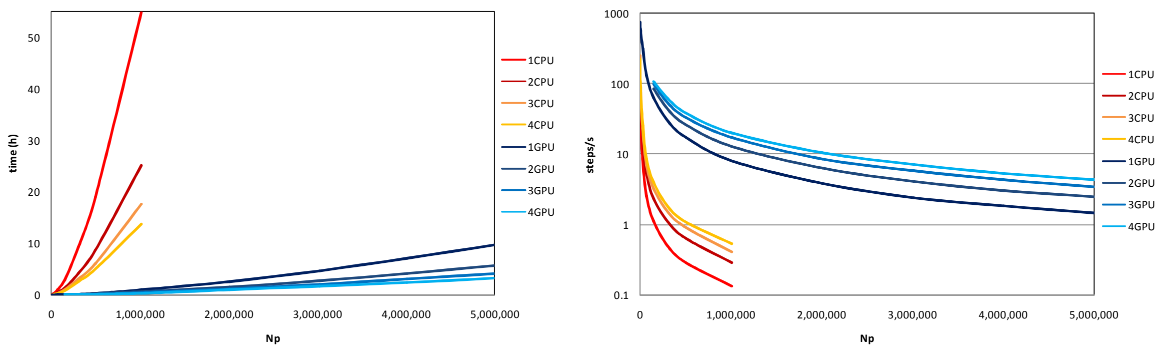

Figure 3: Runtime (left) and number of steps per second (right) for different number of particles using the different parallel codes CPU/GPU.

The computational runtime increases dramatically with the number of particles when simulations are performed on the CPU device while the GPU technology allows accelerating the code easily even with only one GPU card. In addition, the performance of the GPU, measured here as the number of time steps computed per second is always more than one order of magnitude higher than the performance shown by the CPU codes.

The speedups of using *NCPU* and *NGPU* against only one device are shown in Figure 4 to analyse the improvement achieved by using the multi-core and multi-GPU codes respectively. Smaller speedup factors are obtained in the multi-GPU case than in the multi-core approach due to the fact that multi-GPU involves data transferring between different computational devices (GPU to CPU and CPU to CPU), whereas in the multi-core OpenMP approach all data resides in the same compute node, namely, the CPU. Additionally, the multi-GPU scheme does not yet use a high-speed Infiniband connection or feature a load balancing algorithm but the multi-core OpenMP approach incorporates this inherently.

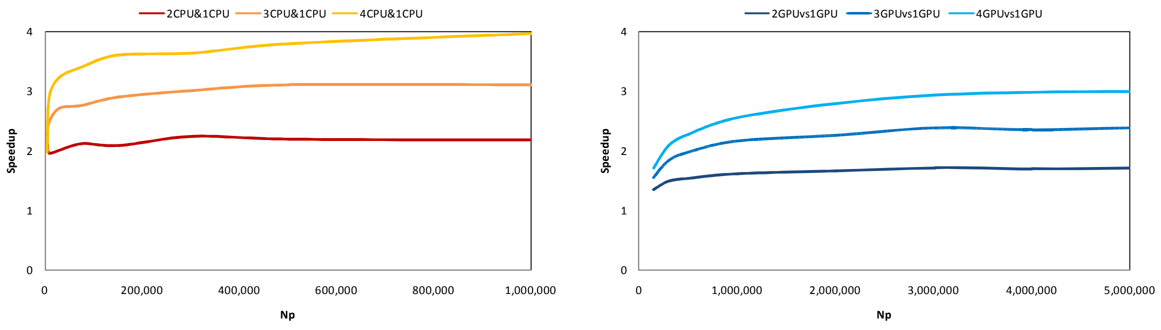

Figure 4: Speedups for each parallel code in comparison to the serial code for CPU (left) and for the GPU device (right) for different number of particles.

The simulation of one million particles takes more than 2 days on CPU and takes only 55mins on a single GPU, thus, a speedup of 60 is achieved (Figure 5). The speedups of using *NGPU* against *NCPU* are also shown in Figure 5 and they are still promising (38-44).

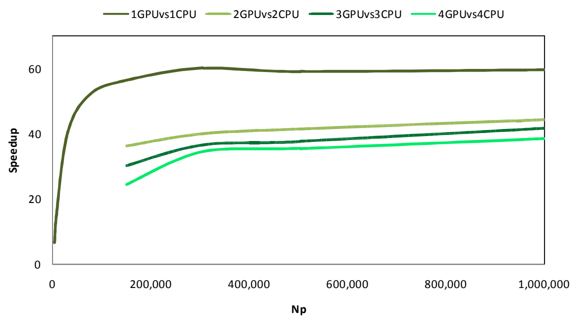

Figure 5: Speedups of the parallel GPU codes in comparison with the parallel CPU ones.

5 APPLICATIONS

In order to create a real complex geometry to reproduce an industrial problem the first main issue is the resolution with which the objects are represented. To obtain realistic results with SPH it is appropriate that the initial geometry is as close as possible to a real industrial problem. This drawback can be solved when several million particles are used in the simulation. Figure 6 shows the example of an SPH simulation for a pump mechanism.

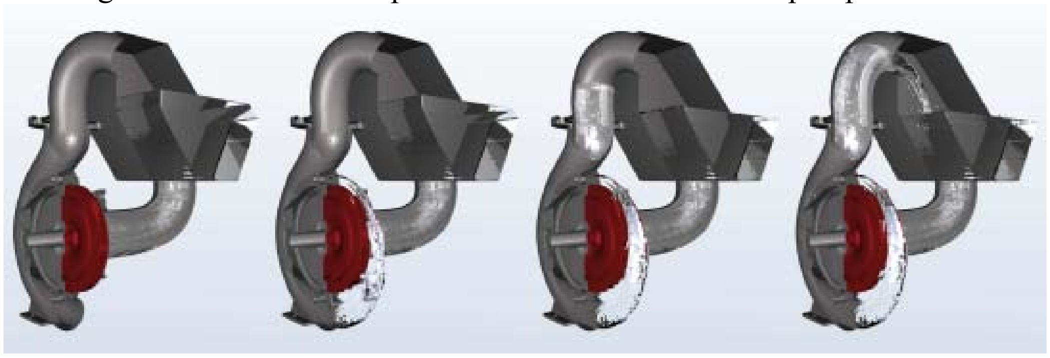

Figure 6: Pump mechanism with SPH using 2.5 million particles.

However the main field of application of our software is the design of coastal protection schemes. Simulating million particles in a few hours allows us to investigate a real scenario where the damage due to extreme waves can be analysed and mitigation structures can be designed. Figure 7 shows different snapshots of a simulation where a large wave interacts with an idealised seafront consisting of a beach, a seawalk, pavement, the street, trees and buildings. The model reproduces realistically an overtopping.

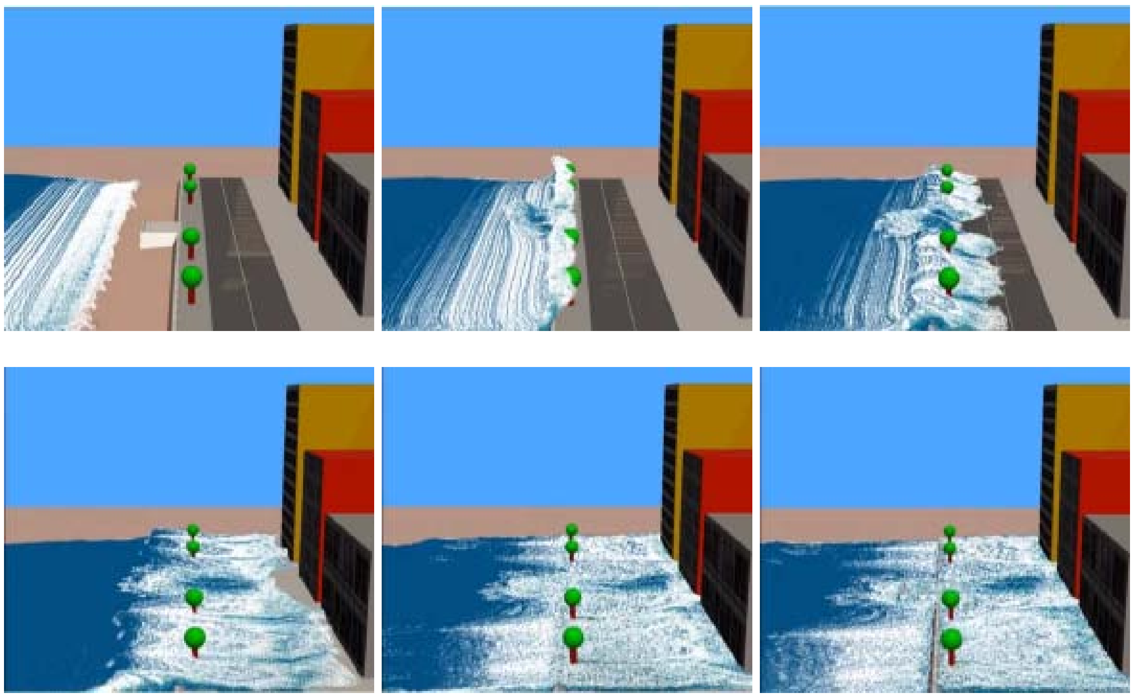
Figure 7: Promenade-wave interaction with SPH using 20 million particles.

6 CONCLUSIONS AND FUTURE LINES

- It is demonstrated that the achieved performance of SPH simulations with a small number of GPUs can be compared to that of large cluster of CPUs, both in terms of speed and in the number of particles employed (Maruzewski et al., 2010) Furthermore, other important advantage is the cost and ease-of-maintenance of GPUs in comparison with those clusters.
- Once the GPU implementation allows studying simulations of large domains with a reasonable computational runtime, real-life engineering problems will be studied for industrial purposes using SPH models.

REFERENCES

[1] Gingold, R.A. and Monaghan, J.J. Smoothed particle hydrodynamics: theory and application to non- spherical stars. *Mon Not R Astr Soc* (1977) 181: 375-389.

[2] Gómez-Gesteira, M., Rogers, B.D., Dalrymple, R.A. and Crespo, A.J.C. State-of-the-art of classical SPH for free-surface flows. *Journal of Hydraulic Research*, (2010) 48: 6–27.

[3] Liu, M. B. and Liu, G. R. Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments. *Arch Comput Methods Eng* (2010) 17: 25-76.

[4] Valdez-Balderas, D., Dominguez, J.M., Crespo, A.J.C. and Rogers, B.D. Massively parallel SPH simulations on multi-GPU cluster for free-surface flows. The fourth International Many-core and Reconfigurable Supercomputing Conference, Bristol 2011.

[5] Gómez-Gesteira, M. and Dalrymple, R. Using a 3D SPH method for wave impact on a tall structure. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, (2004) 130(2): 63-69.

[6] Chen, S., Johnson, D. B., Raad, P. E. and Fadda, D.. The surface marker and microcell method. *Int. J. Numer. Methods Fluids,* (1997) 25: 749–778.

[7] Maruzewski, P., Touzé, D. Le, Oger, G. and Avellan, F. SPH high-performance computing simulations of rigid solids impacting the free-surface of water. *Journal of Hydraulic Research*, (2010) 48: 126–134.