LABIMA – Laboratory of Maritime Engineering
Florence, 27th September 2018

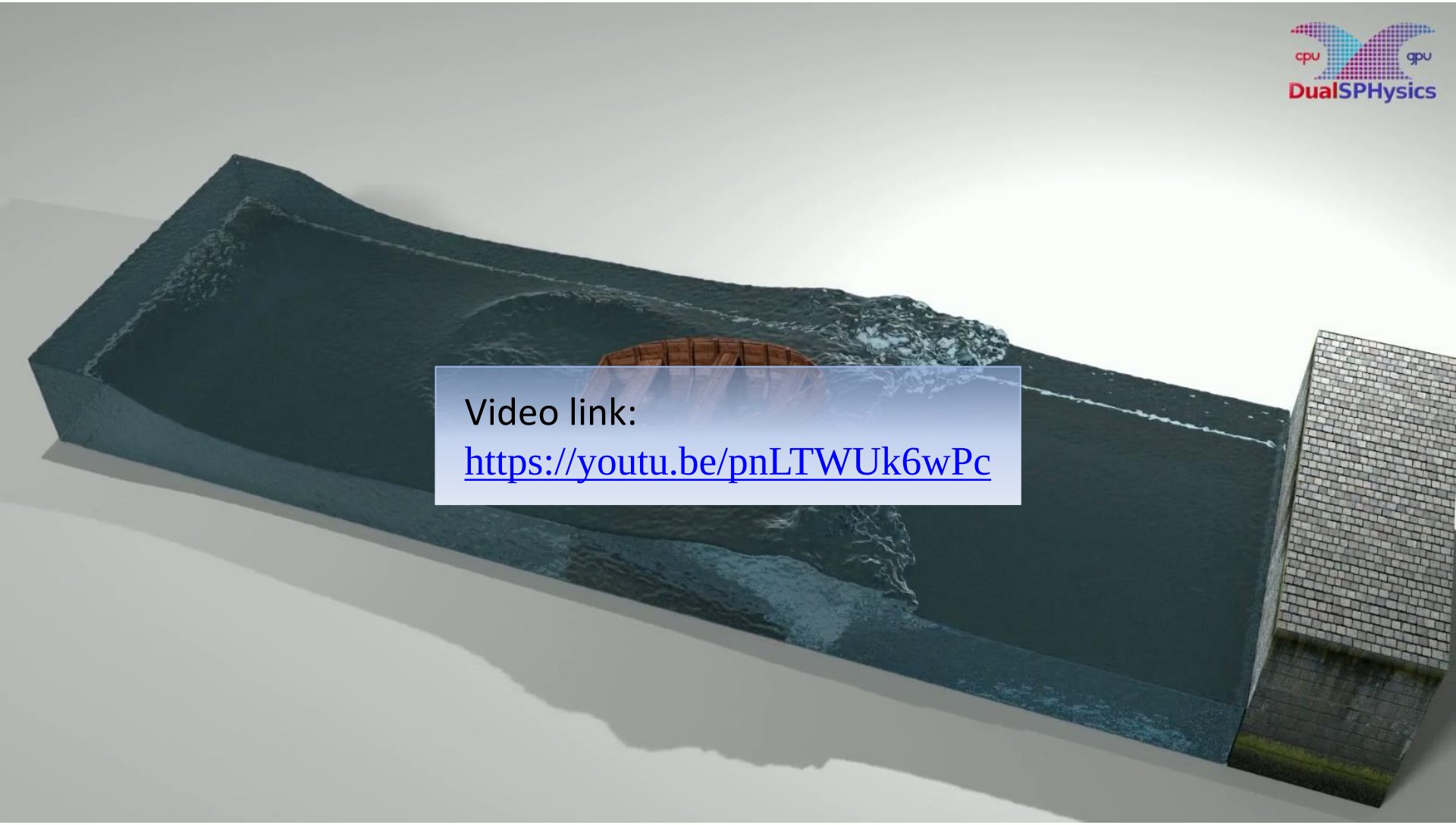
The DualSPHysics code

José Domínguez – Ephyslab - UVigo

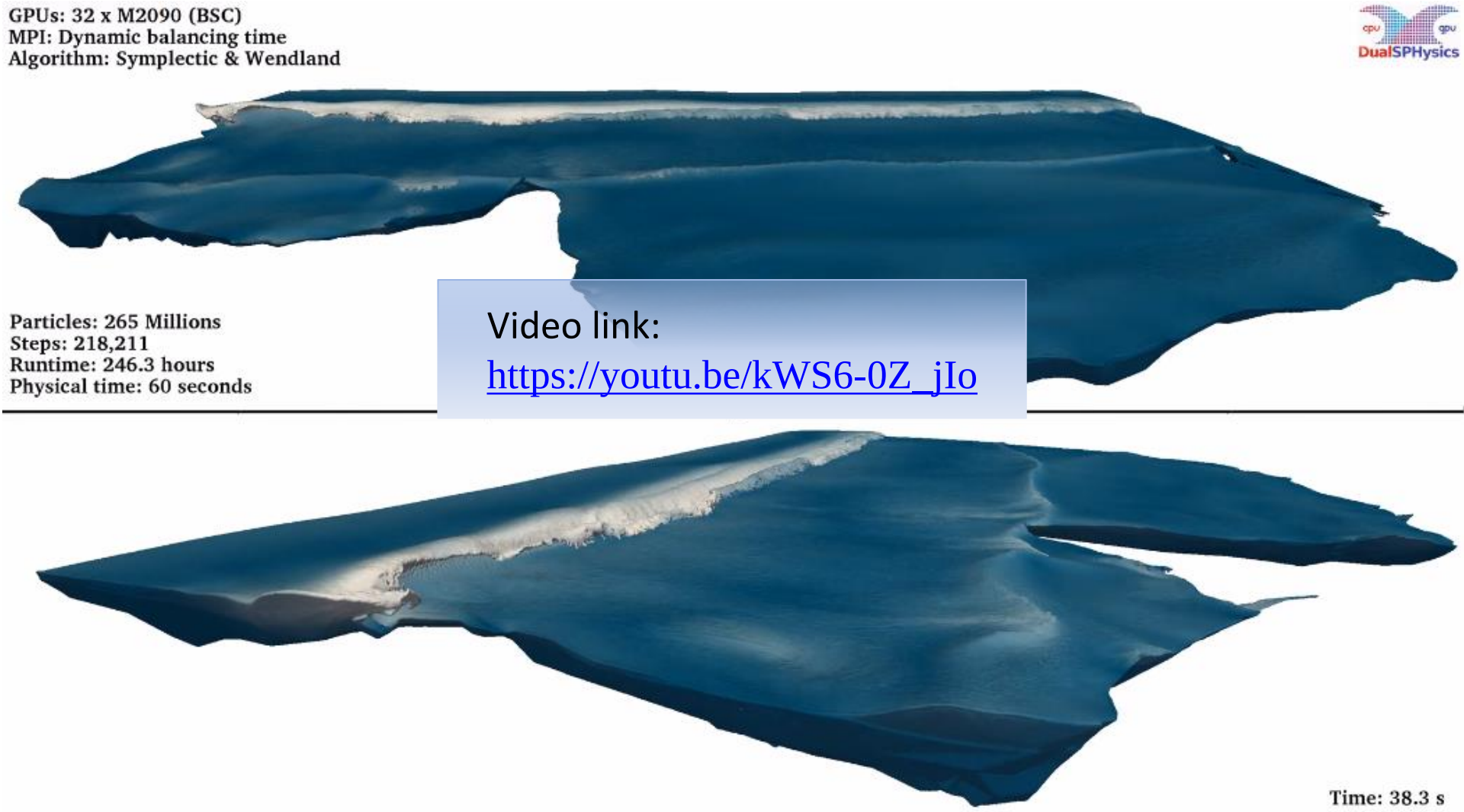
Universidade Vigo

cpu gpu DualSPHysics

DualSPHysics code



DualSPHysics code



Outline

1.1. Smooth Particle Hydrodynamics

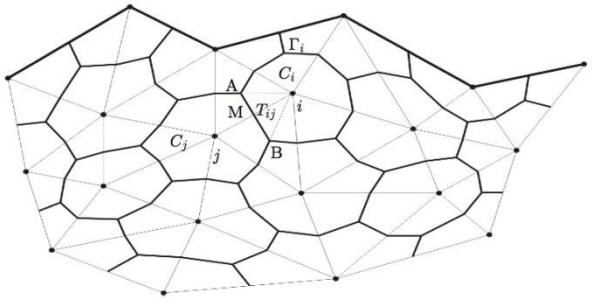
PHYSICAL GOVERNING EQUATIONS

EULERIAN DESCRIPTION
(spatial description)

LAGRANGIAN DESCRIPTION
(material description)

COMPUTATIONAL METHODS

GRID-BASED METHODS



SPH

MESHFREE METHODS

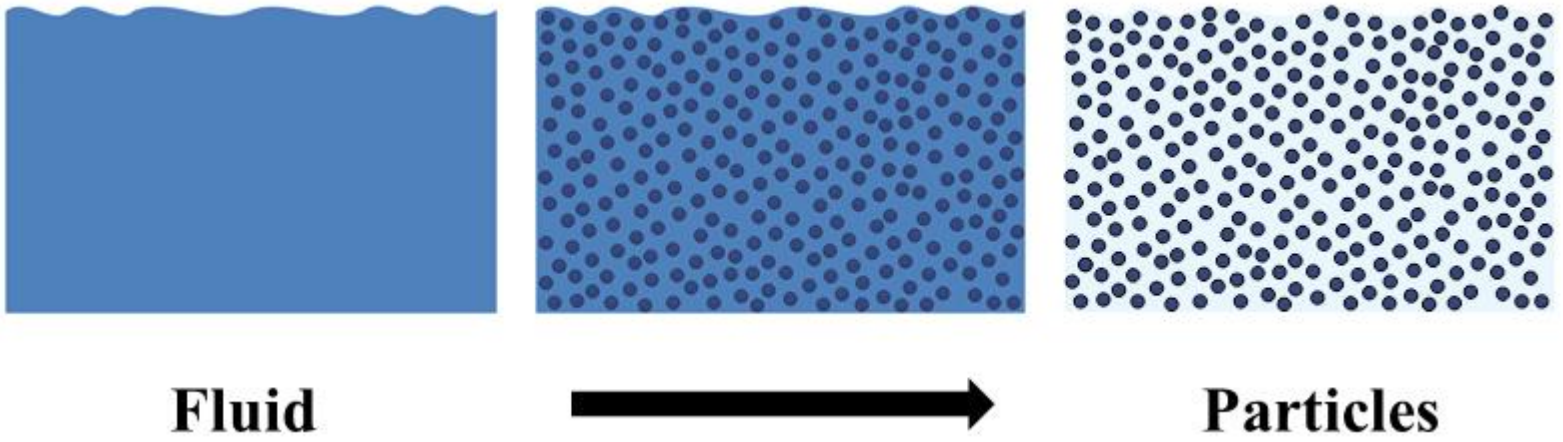
MESHFREE PARTICLE METHODS
(particle represents a part of the continuum domain)

SMOOTHED PARTICLE HYDRODYNAMICS

1.1. Smooth Particle Hydrodynamics

SPH method was invented for astrophysics during the seventies, but now it is applied in many different fields including fluid dynamics and solid mechanics.

Fluid is represented using particles which move according to the governing dynamics.



Comparing to grid-based methods, SPH interactions are carried out between a given particle and its moving neighbours. Thus, these **neighbours are unknown since they change at each instant.**

1.1. Smooth Particle Hydrodynamics

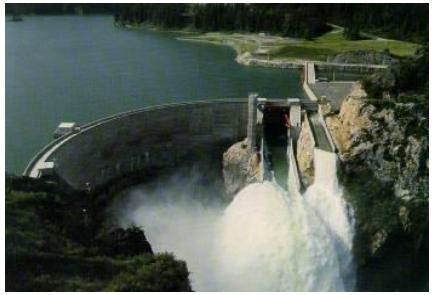
SPH method was invented for astrophysics during the seventies, but now it is applied in many different fields including fluid dynamics and solid mechanics.

Fluid is represented using particles which move according to the governing dynamics.

SPH is particularly suited to describe a variety of **free-surface flows**:

- Wave propagation over a beach.
- Plunging breakers.
- Wave-structure interactions.
- Solid bodies impacting on water surface.
- Dam breaks.



1.2. Why is SPH too slow?

Drawbacks of SPH:

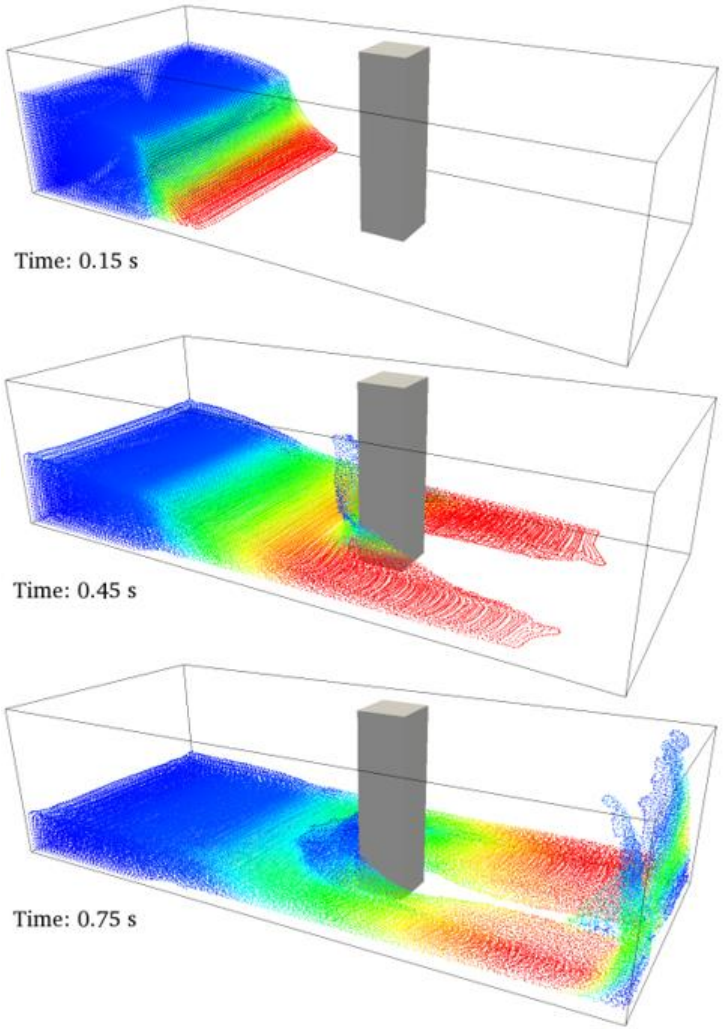
- SPH presents a **high computational cost** that increases when increasing the number of particles.

- The simulation of **real problems** requires a high resolution which implies simulating **millions of particles**.

The **time required** to simulate a few seconds is **too large**. One second of physical time can take several days of calculation.

1.2. Why is SPH too slow?

The SPH method is very expensive in terms of computing time.



For example, a simulation of this dam break

300,000 particles

✚

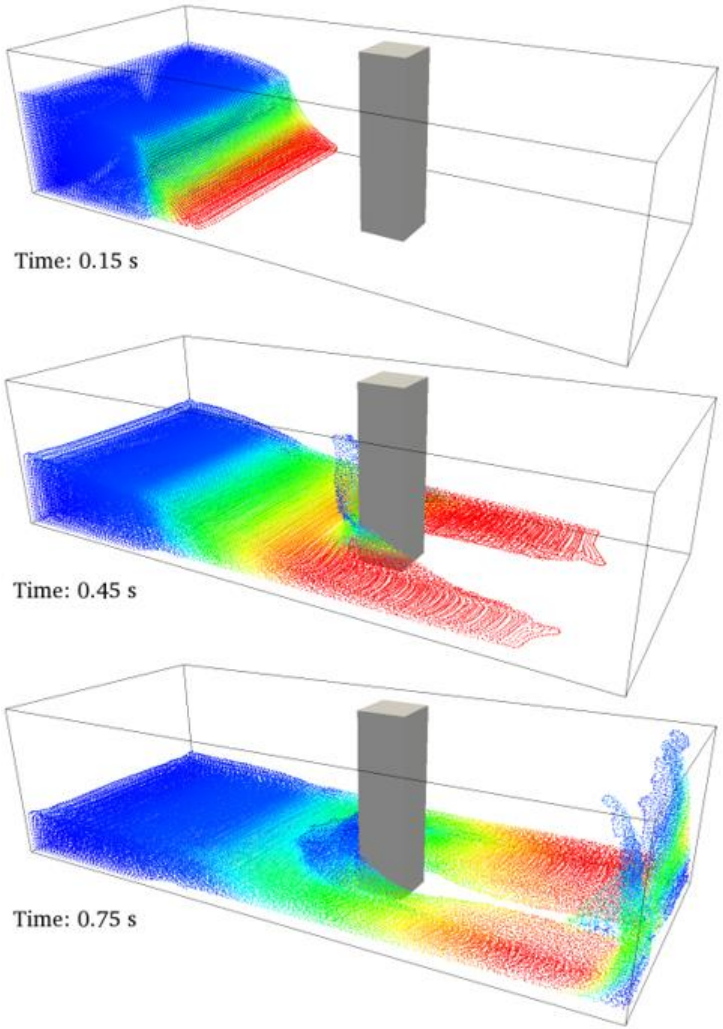
1.5 s (*physical time*)

➡

Takes more than
15 hours
(*execution time*)

1.2. Why is SPH too slow?

The SPH method is very expensive in terms of computing time.



For example, a simulation of this dam break

300,000 particles

✚

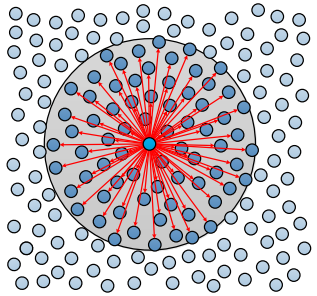
1.5 s (*physical time*)

➡

Takes more than
15 hours
(*execution time*)

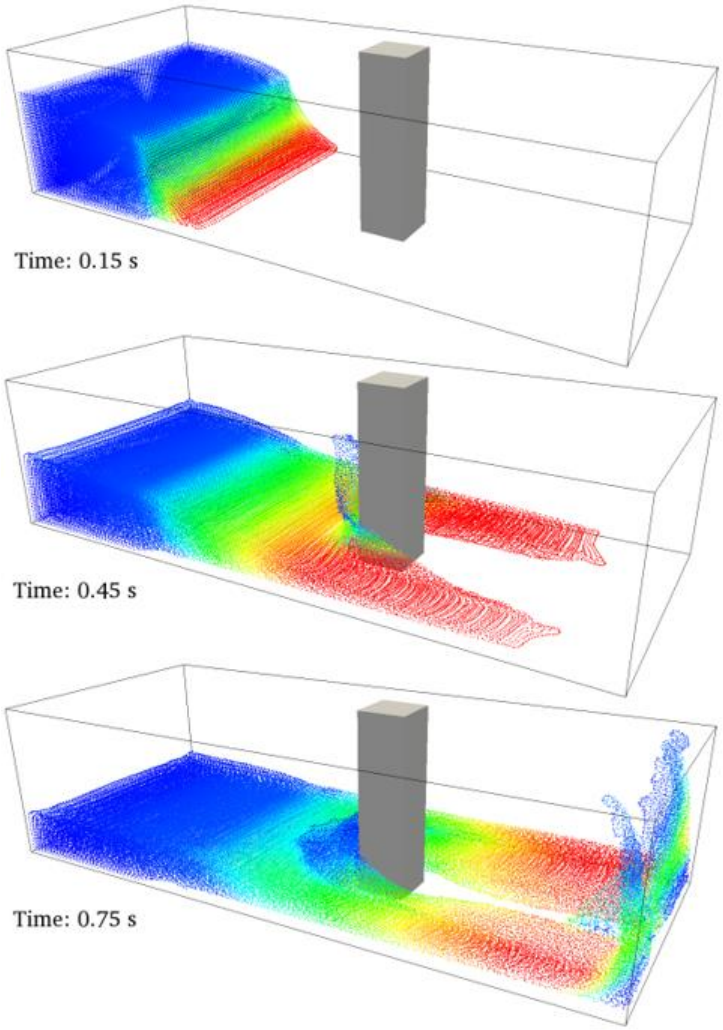
because:

- Each particle interacts with **more than 250 neighbours**.



1.2. Why is SPH too slow?

The SPH method is very expensive in terms of computing time.



For example, a simulation of this dam break

300,000 particles

✚

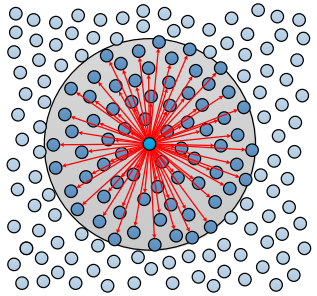
1.5 s (*physical time*)

➡

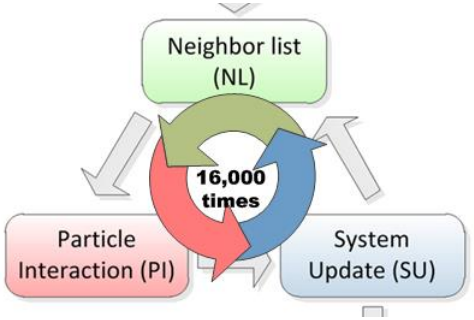
Takes more than **15 hours** (*execution time*)

because:

- Each particle interacts with **more than 250 neighbours**.



- $\Delta t=10^{-5}$-10^{-4} **so more than 16,000 steps** are needed to simulate 1.5 s of physical time.



1.2. Why is SPH too slow?

Drawbacks of SPH:

- SPH presents a **high computational cost** that increases when increasing the number of particles.

- The simulation of **real problems** requires a high resolution which implies simulating **millions of particles**.

The **time required** to simulate a few seconds is **too large**. One second of physical time can take several days of calculation.

IT IS NECESSARY TO USE HPC TECHNIQUES TO REDUCE THESE COMPUTATION TIMES.

1.3. High Performance Computing (HPC)

HPC includes multiple techniques of parallel computing and distributed computing that allow you to execute several operations simultaneously.

The main techniques used to accelerate SPH are:

- **OpenMP** (Open Multi-Processing)
  - Model of parallel programming for systems of shared memory.
  - Portable and flexible programming interface using directives.
  - Its implementation does not involve major changes in the code.
  - The improvement is limited by the number of cores.

Multi-core processor

OPENMP IS THE BEST OPTION TO OPTIMIZE THE PERFORMANCE OF THE MULTIPLE CORES OF THE CURRENT CPUS.

1.3. High Performance Computing (HPC)

HPC includes multiple techniques of parallel computing and distributed computing that allow you to execute several operations simultaneously.

The main techniques used to accelerate SPH are:

- **MPI** (Message Passing Interface)
  - Message-passing library specification for systems of distributed memory: parallel computers and clusters.
  - Several processes are communicated by calling routines to send and receive messages.
  - The use of MPI is typically combined with OpenMP in clusters by using a hybrid communication model.
  - Very expensive for a small research group.

MPI cluster

MPI IS THE BEST OPTION TO COMBINE THE RESOURCES OF MULTIPLE MACHINES CONNECTED VIA NETWORK.

1.3. High Performance Computing (HPC)

HPC includes multiple techniques of parallel computing and distributed computing that allow you to execute several operations simultaneously.

The main techniques used to accelerate SPH are:

- **GPGPU** (General-Purpose Computing on Graphics Processing Units)
  - It involves the study and use of parallel computing ability of a GPU to perform general purpose programs.
  - New general purpose programming languages and APIs (such as Brook and **CUDA**) provide an easier access to the computing power of GPUs.
  - New implementation of the algorithms used in CPU is necessary for an efficient use in GPU.

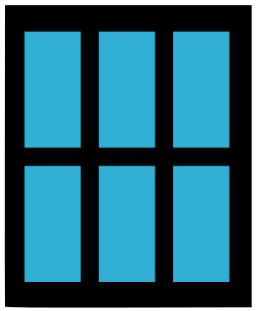
GPU

1.3. High Performance Computing (HPC)

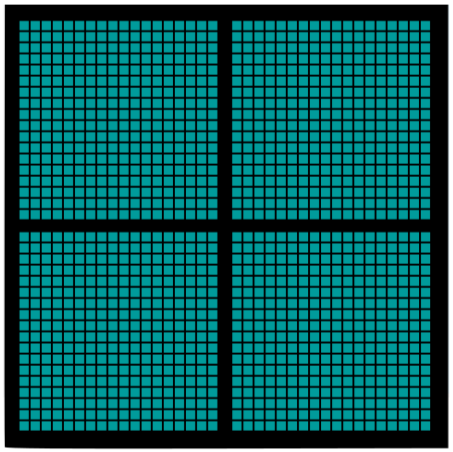
Central Processing Units (CPUs) vs. Graphics Processing Units (GPUs)

- general purpose processor (any complex calculations)
- designed for serial data processing
- complex and large cores, so limited number cores in one CPU

- designed for graphics rendering
- designed for simple calculations that require high parallelism
- simple cores, but large number of cores in one GPU



CPU
Multiple Cores

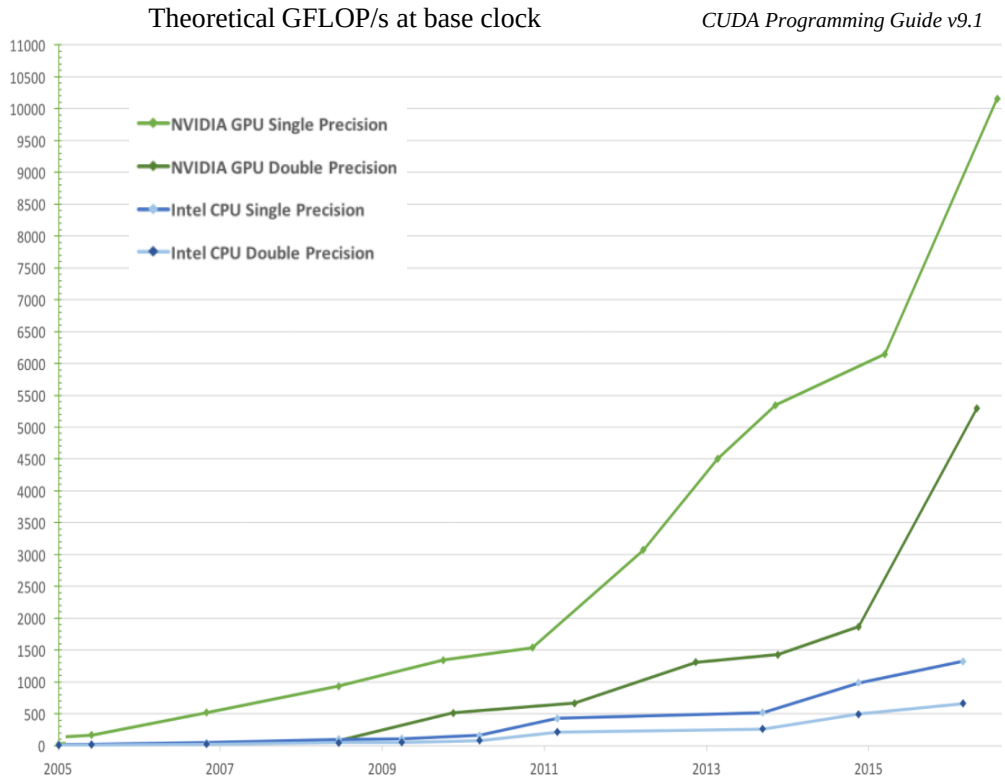


GPU
Thousands of Cores

1.3. High Performance Computing (HPC)

Graphics Processing Units (GPUs)

- video game market boosted its improvement
- their computing power has increased much faster than CPUs.
- powerful parallel processors



Advantages: GPUs provide a high calculation power with very low cost and without expensive infrastructures.

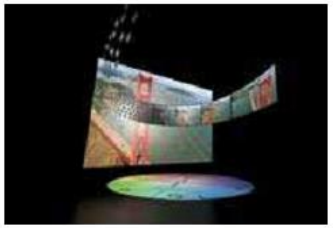
Drawbacks: An efficient and full use of the capabilities of the GPUs is not straightforward.

1.3. High Performance Computing (HPC)

Why GPUs?

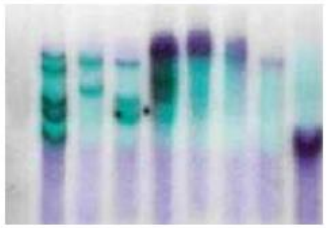
GPUs are an accessible tool to accelerate SPH,
all numerical methods in CFD and any computational method

5X
Digital Content Creation
Adobe

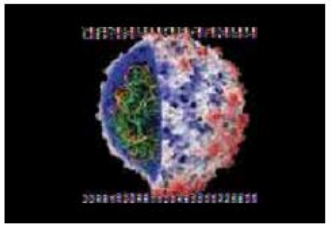


18X
Video Transcoding
Elemental Technologies

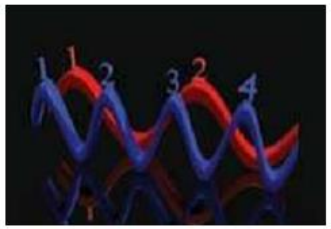
20X
3D Ultrasound
TechniScan



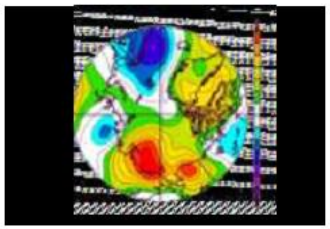
30X
Gene Sequencing
U of Maryland



36X
Molecular Dynamics
U of Illinois, Urbana-Champaign

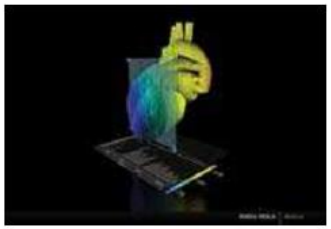


50X
MATLAB Computing
AccelerEyes

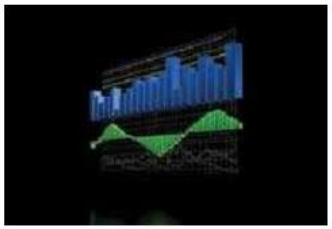


80X
Weather Modeling
Tokyo Institute of Technology

100X
Astrophysics
RIKEN



146X
Medical Imaging
U of Utah



149X
Financial Simulation
Oxford University

http://www.nvidia.com

1.3. High Performance Computing (HPC)

Why GPUs?

TOP500 LIST – JUNE 2018

https://www.top500.org/lists/2018/06/

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, **NVIDIA Volta GV100**, Dual-rail Mellanox EDR Infiniband , IBM, DOE/SC/Oak Ridge National Laboratory United States	2,282,544	122,300.0	187,659.3	8,806
2	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC, National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
3	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, **NVIDIA Volta GV100**, Dual-rail Mellanox EDR Infiniband , IBM, DOE/NNSA/LLNL United States	1,572,480	71,610.0	119,193.6	
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT, National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2550 M4, Xeon Gold 6148 20C 2.4GHz, **NVIDIA Tesla V100 SXM2**, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	1,649
6	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries intercon., **NVIDIA Tesla P100**, Cray Inc., Swiss National Supercomputing Centre (CSCS) Switzerland	361,760	19,590.0	25,326.3	2,272
7	**Titan** - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, **NVIDIA K20x** , Cray Inc., DOE/SC/Oak Ridge National Laboratory United States	560,640	17,590.0	27,112.5	8,209

1.3. High Performance Computing (HPC)

Why GPUs?

GREEN500 LIST – JUNE 2018

https://www.top500.org/green500/lists/2018/06/

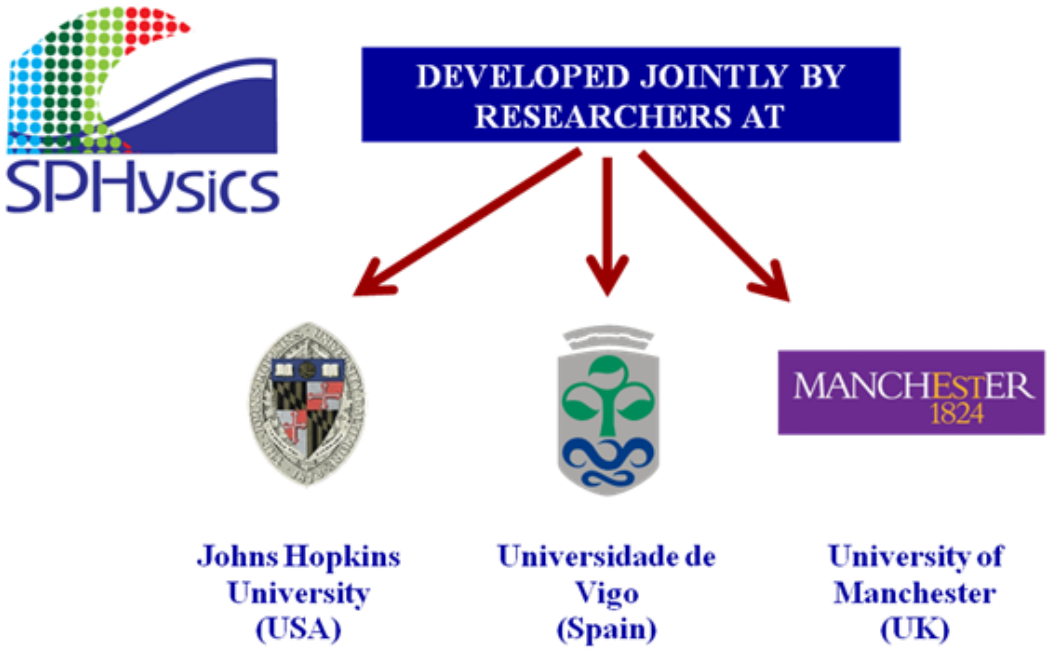
Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	Power (kW)	Power Efficiency (GFlops/watts)
1	359	**Shoubu system B** - ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 , PEZY Computing / Exascaler Inc., Advanced Center for Computing and Communication, RIKEN Japan	794,400	857.6	47	18.404
2	419	**Suiren2** - ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 , PEZY Computing / Exascaler Inc., High Energy Accelerator Research Organization /KEK Japan	762,624	798.0	47	16.835
3	385	**Sakura** - ZettaScaler-2.2, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband EDR, PEZY-SC2 , PEZY Computing / Exascaler Inc., PEZY Computing K.K. Japan	794,400	824.7	50	16.657
4	227	**DGX SaturnV Volta** - **NVIDIA DGX-1 Volta36**, Xeon E5-2698v4 20C 2.2GHz, Infiniband EDR, NVIDIA Tesla V100 , Nvidia, NVIDIA Corporation United States	22,440	1,070.0	97	15.113
5	1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, **NVIDIA Volta GV100**, Dual-rail Mellanox EDR Infiniband , IBM, DOE/SC/Oak Ridge National Laboratory United States	2,282,544	122,300.0	8,806	13.889
6	19	**TSUBAME3.0** - SGI ICE XA, IP139-SXM2, Xeon E5-2680v4 14C 2.4GHz, Intel Omni-Path, **NVIDIA Tesla P100** SXM2 , HPE, GSIC Center, Tokyo Institute of Technology Japan	135,828	8,125.0	792	13.704
7	287	**AIST AI Cloud** - NEC 4U-8GPU Server, Xeon E5-2630Lv4 10C 1.8GHz, Infiniband EDR, **NVIDIA Tesla P100** SXM2 , NEC, National Institute of Advanced Industrial Science and Technology Japan	23,400	961.0	76	12.681
8	5	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2550 M4, Xeon Gold 6148 20C 2.4GHz, **NVIDIA Tesla V100** SXM2, Infiniband EDR , Fujitsu, National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	1,649	12.054
9	255	**MareNostrum P9 CTE** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, Dual-rail Mellanox EDR Infiniband, **NVIDIA Tesla V100** , IBM, Barcelona Supercomputing Center Spain	19,440	1,018.0	86	11.865
10	171	**RAIDEN GPU subsystem** - NVIDIA DGX-1 Volta36, Xeon E5-2698v4 20C 2.2GHz, Infiniband EDR, **NVIDIA Tesla V100** , Fujitsu, Center for Advanced Intelligence Project, RIKEN Japan	35,360	1,213.0	107	11.363

Outline

1. Introduction
    1.1. Smooth Particle Hydrodynamics
    1.2. Why is SPH too slow?
    1.3. High Performance Computing (HPC)
2. **DualSPHysics project**
3. SPH formulation
4. DualSPHysics code
    4.1. Source files
    4.2. Object-Oriented Programming
    4.3. Execution diagram
5. DualSPHysics implementation
    5.1. CPU acceleration
    5.2. GPU acceleration
    5.3. Multi-GPU acceleration

2. DualSPHysics project

The DualSPHysics code was created starting from SPHysics.



SPHysics is a numerical model SPH developed for the study of free-surface problems.

It is a code written in Fortran90 with numerous options (different kernels, several boundary conditions,…), which had already demonstrated high accuracy in several validations with experimental results… but it is too slow to apply to large domains.

2. DualSPHysics project

First version in late 2009.

It includes **two implementations**:

- **CPU**: C++ and OpenMP.
- **GPU**: CUDA.

Both options optimized for the best performance of each architecture.

Why two implementations?

This code can be used on machines with GPU and without GPU.

It allows us to make a fair and realistic comparison between CPU and GPU.

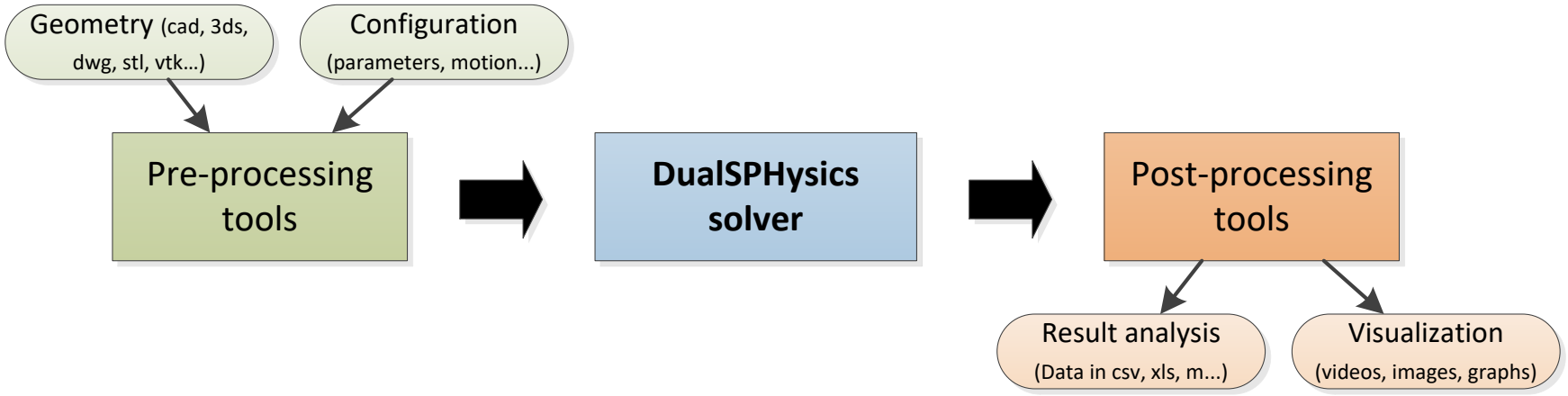
Some algorithms are complex and it is easy to make errors difficult to detect. So they are implemented twice and we can compare results.

It is easier to understand the code in CUDA when you can see the same code in C++.

Drawback: It is necessary to implement and to maintain two different codes.

2. DualSPHysics project

DSPH project includes:



Pre-processing tools:

- Converts geometry into particles.
- Provides configuration for simulation.

DualSPHysics solver:

- Runs simulation using SPH particles.
- Obtains data simulation for time intervals.

Post-processing tools:

- Calculates magnitudes using particle data.
- Generates images and videos starting form SPH particles.

2. DualSPHysics project

DualSPHysics

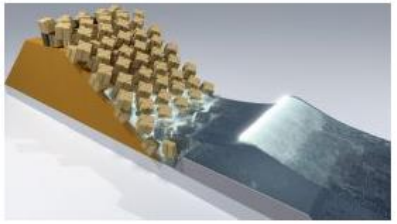
FAQ References Downloads Validation Animations SPHysics GPU Computing
Developers Contact News Forums

Universidade Vigo

MANCHESTER 1824
The University of Manchester

DualSPHysics is based on the Smoothed Particle Hydrodynamics model named SPHysics (www.sphysics.org).

The code is developed to study free-surface flow phenomena where Eulerian methods can be difficult to apply, such as waves or impact of dam-breaks on off-shore structures. **DualSPHysics** is a set of C++, CUDA and Java codes designed to deal with real-life engineering problems.



Contact E-Mail: dualsphysics@gmail.com

Youtube Channel: www.youtube.com/user/DualSPHysics

Twitter Account: @DualSPHysics

www.dual.sphysics.org

2. DualSPHysics project

DEVELOPED JOINTLY BY RESEARCHERS AT

Universidade de Vigo (Spain)

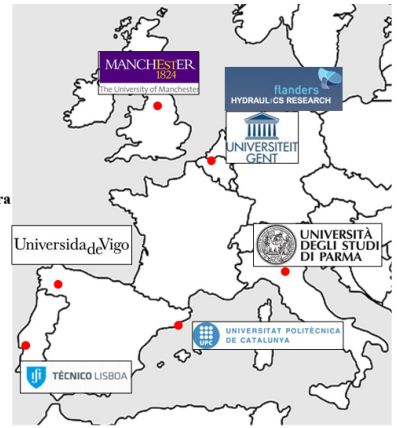
University of Manchester (UK)

and many contributors

Dr Benedict D. Rogers
Dr Athanasios Mokos
Dr Georgios Fourtakas
Prof. Peter Stansby
Alex Chow
Annelie Baines

Prof. Moncho Gómez Gesteira
Dr Alejandro J.C. Crespo
Dr José M. Domínguez
Dr José González Cao
Orlando G. Feal
Andrés Vieira

Prof. Rui Ferreira
Dr Ricardo Canelas
Moisés Brito



Dr Corrado Altomare
Dr Tomohiro Suzuki
Tim Verbrugghe

Dr Renato Vacondio
Prof. Paolo Mignosa

Dr Xavier Gironella
Dr Andrea Marzeddu

www.dual.sphysics.org

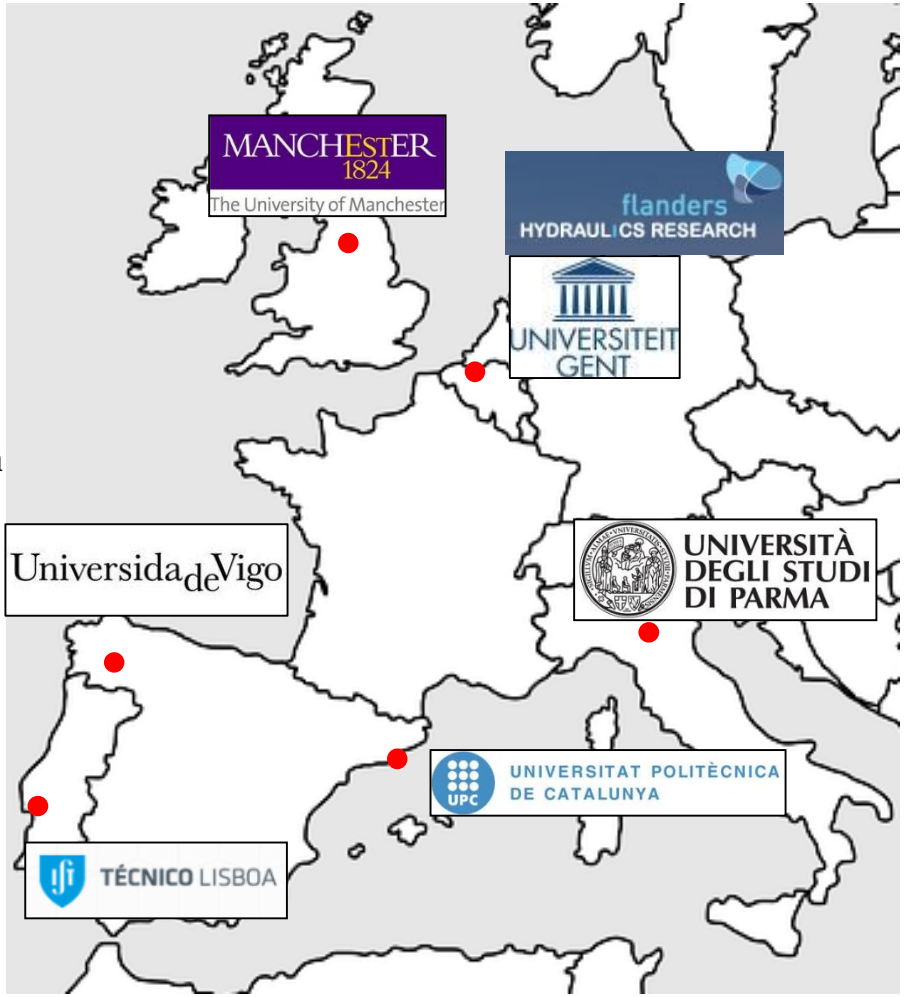
2. DualSPHysics project

People working on DualSPHysics project:

Dr Benedict D. Rogers
Dr Athanasios Mokos
Dr Georgios Fourtakas
Prof. Peter Stansby
Alex Chow
Annelie Baines

Prof. Moncho Gómez Gesteira
Dr Alejandro J.C. Crespo
Dr José M. Domínguez
Dr José González Cao
Orlando G. Feal
Andrés Vieira

Prof. Rui Ferreira
Dr Ricardo Canelas
Moisés Brito



Dr Corrado Altomare
Dr Tomohiro Suzuki
Tim Verbrugghe

Dr Renato Vacondio
Prof. Paolo Mignosa

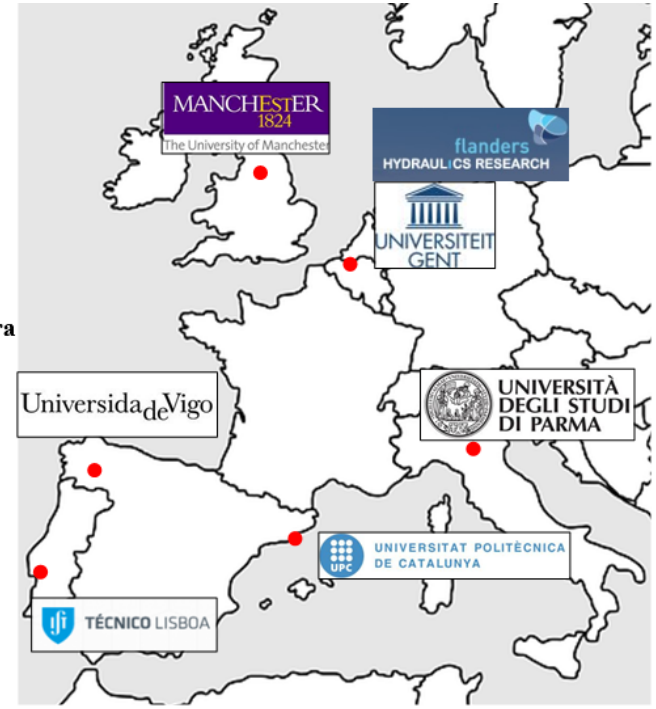
Dr Xavier Gironella
Dr Andrea Marzeddu

2. DualSPHysics project

Dr Benedict D. Rogers
Dr Athanasios Mokos
Dr Georgios Fourtakas
Prof. Peter Stansby
Alex Chow
Annelie Baines

Prof. Moncho Gómez Gesteira
Dr Alejandro J.C. Crespo
Dr José M. Domínguez
Dr José González Cao
Orlando G. Feal
Andrés Vieira

Prof. Rui Ferreira
Dr Ricardo Canelas
Moisés Brito



Dr Corrado Altomare
Dr Tomohiro Suzuki
Tim Verbrugghe

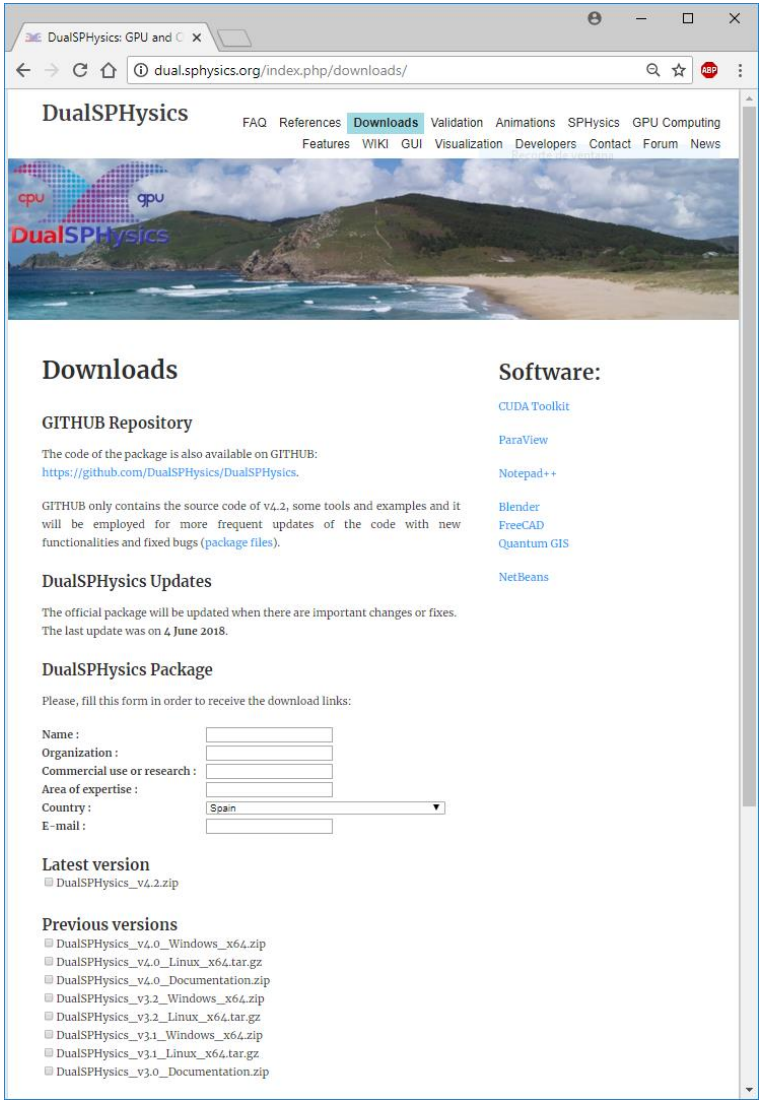
Dr Renato Vacondio
Prof. Paolo Mignosa

Dr Xavier Gironella
Dr Andrea Marzeddu

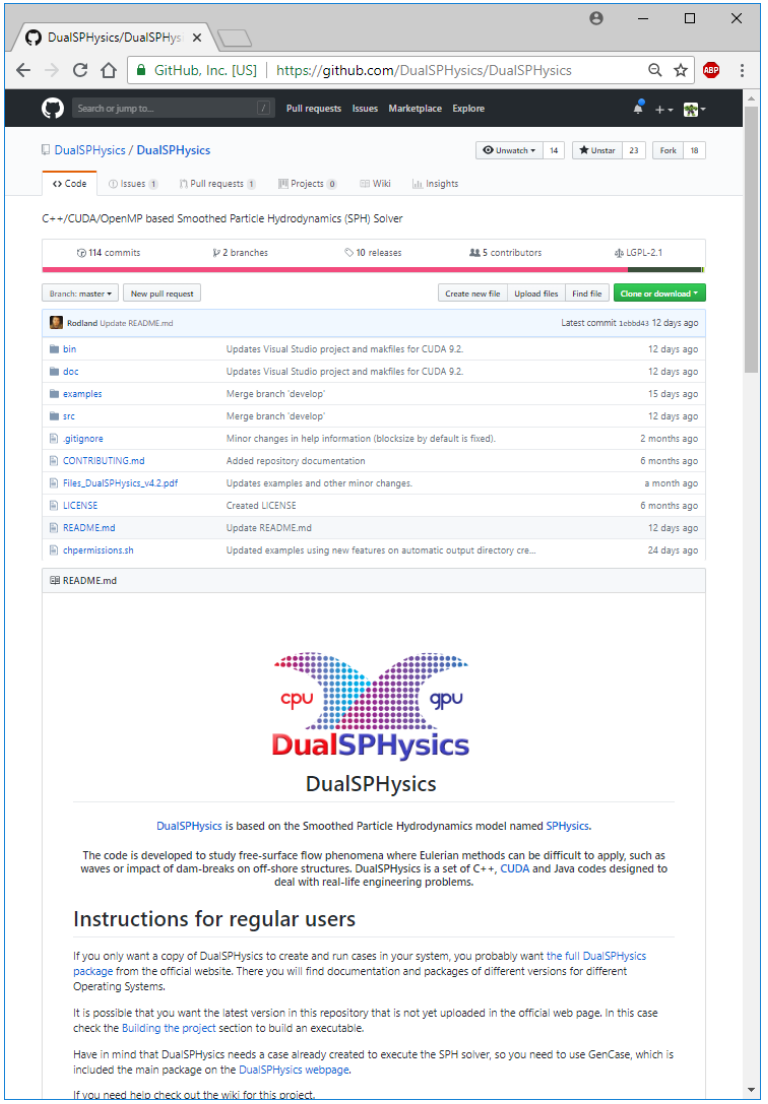
It has been downloaded and used by researchers but also by companies:
NASA JSC, BAE Systems, Volkswagen AG, McLaren Racing Ltd, Forum NOKIA, NVIDIA, AECOM, HDR Engineering, ABPmer, DLR, Maine Marine Composites, CFD-NUMERICS, BMT Group, Oak Ridge National Laboratory, Rainpower Norway, American Wave Machines,, National Renewable Energy Laboratory in U.S.A., Atria Power Corporation Ltd., Global Hydro Energy, Carnegie Wave Energy Ltd, etc.

2. DualSPHysics project - Download

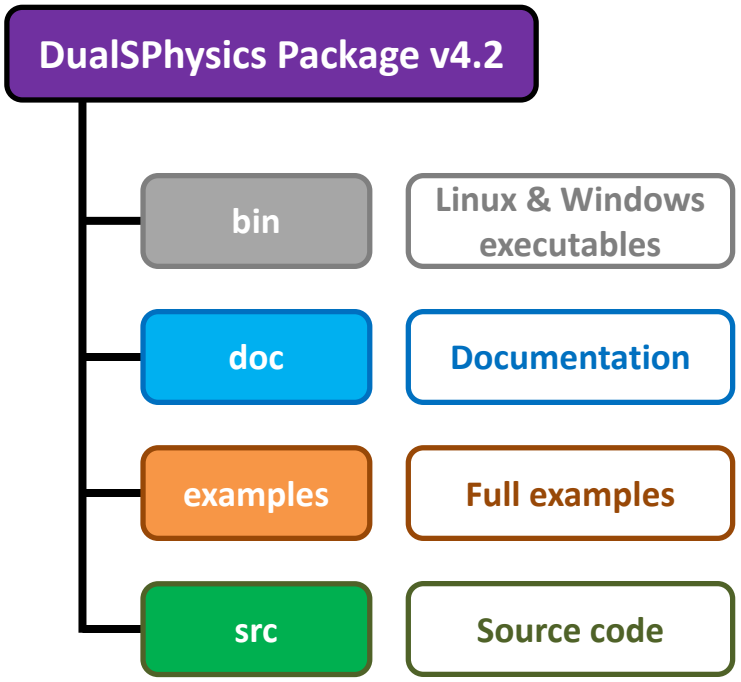
DualSPHysics Package
http://dual.sphysics.org



DualSPHysics Code on GitHub (since v4.2)
https://github.com/DualSPHysics/DualSPHysics



2. DualSPHysics project - Download



Linux & Windows executables:

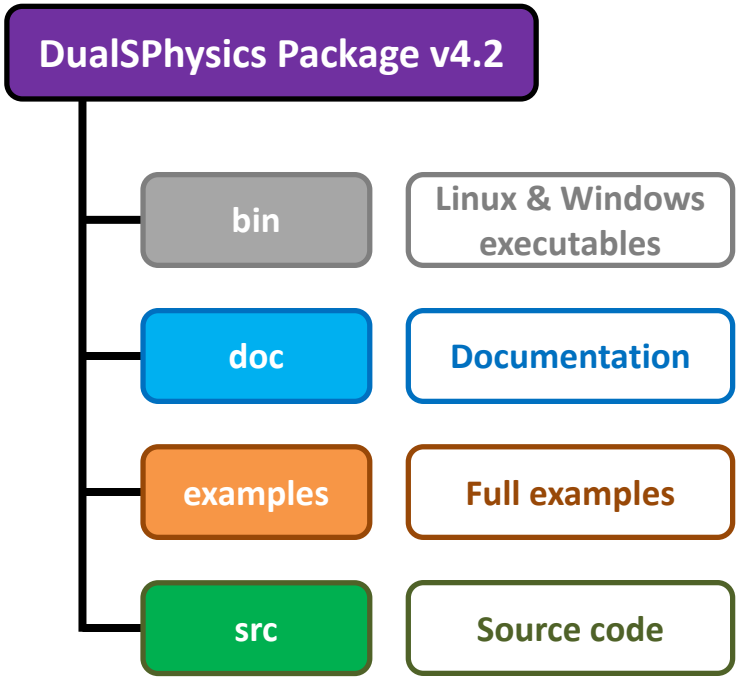
Pre-processing:
- GenCase4

SPH solver:
- DualSPHysics4.2
- DualSPHysics4.0_LiquidGas
- DualSPHysics3.4_LiquidSediment

Post-processing (visualization):
- PartVTK4
- PartVTKOut4
- IsoSurface4

Post-processing (calculations):
- BoundaryVTK4
- ComputeForces4
- FloatingInfo4
- FlowTool4
- MeasureTool4

2. DualSPHysics project - Download



Documentation (guides and other help files) :

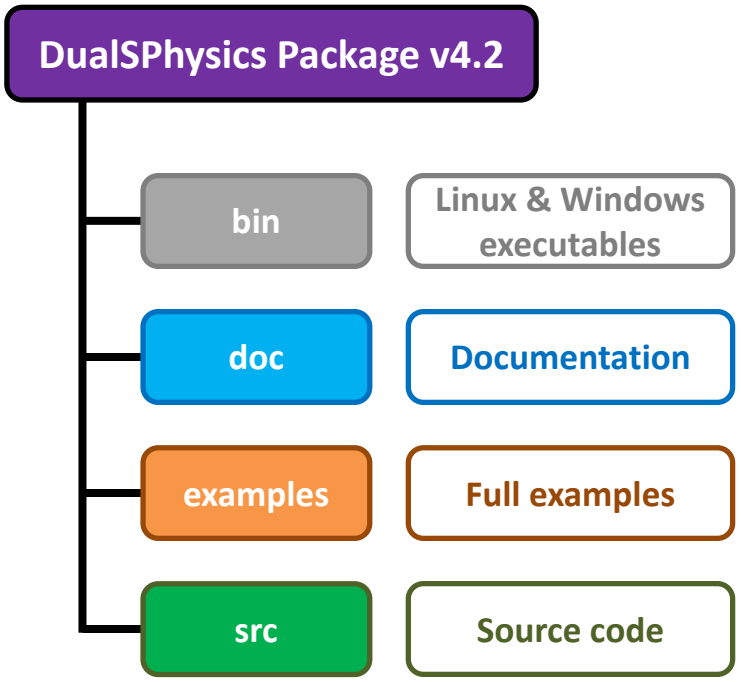
Pre-processing:
- XML_v4.0_GUIDE.pdf
- ExternalModelsConversion.pdf

SPH solver:
- DualSPHysics_v4.2_GUIDE.pdf
- DualSPHysics_v4.0_LiquidGas_GUIDE.pdf

Post-processing:
- PostprocessingCalculations_v4.2.pdf

Help of executables
XML Templates for configuration

2. DualSPHysics project - Download



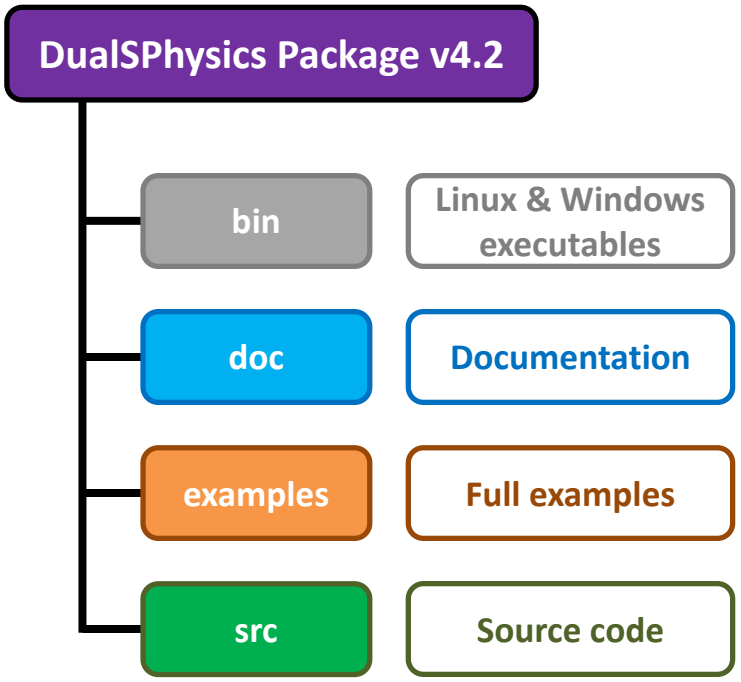
Full examples (also pre-processing & post-processing):

DamBreak	WaveMaker	Floating
Periodicity	WavesFlap	FloatingWaves
MovingSquare	WavesPiston	Pump
ExternalForces	WavesPistonAWAS	DEM
SloshingTank	WaveMakerFile	Pouseuille

Full examples - LiquidSediment & LiquidGas:

Dambreak	ObstacleImpact	DamBreak
	SloshingTank	WetDambreak
	SurfaceTension	DEM

2. DualSPHysics project - Download



Source code ready to compile:

Codes:
- DualSPHysics v4.2
- DualSPHysics v4.0 LiquidGas
- ToVTK (data usage example)

Precompiled libraries:
- Linux (gcc4 & gcc5)
- Windows (Visual Studio 2015)

Compiling:
- Makefiles for Linux
- Project for Visual Studio 2015
- CMake file

2. DualSPHysics project - Download

DUALSPHYSICS V1.2 (2011)
Downloads: 701 (65% Windows)

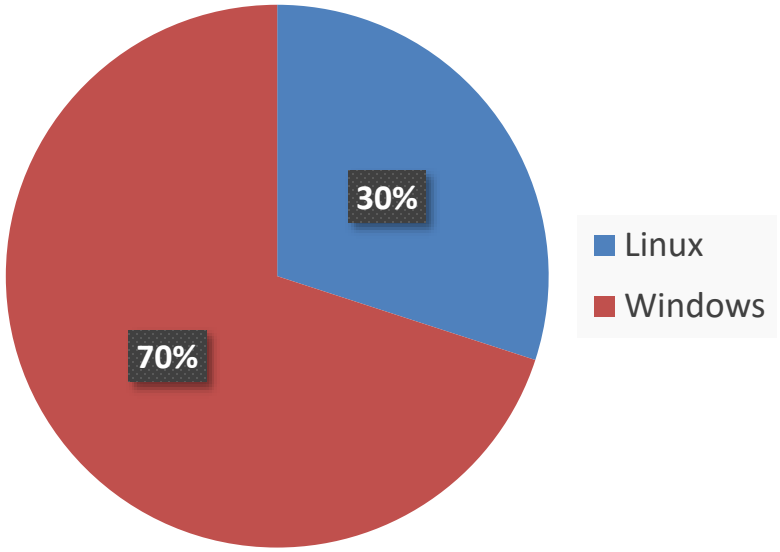
DUALSPHYSICS V2.0 (2012)
Downloads: 6,472 (71% Windows)

DUALSPHYSICS V3.0 (2013)
Downloads: 6,982 (73% Windows)

DUALSPHYSICS V4.0 (2016)
Downloads: 7,072 (72% Windows)

DUALSPHYSICS V4.2 (May 2018)
Downloads: 1051
GitHub downloads: ???

DUALSPHYSICS - ALL VERSIONS
Downloads: 22,278 (70% Windows)



Outline

1. Introduction
    1.1. Smooth Particle Hydrodynamics
    1.2. Why is SPH too slow?
    1.3. High Performance Computing (HPC)
2. DualSPHysics project
3. **SPH formulation**
4. DualSPHysics code
    4.1. Source files
    4.2. Object-Oriented Programming
    4.3. Execution diagram
5. DualSPHysics implementation
    5.1. CPU acceleration
    5.2. GPU acceleration
    5.3. Multi-GPU acceleration

3. SPH formulation: DualSPHysics v4.2

- Time integration scheme:
  - Verlet [Verlet, 1967]
  - Symplectic [Leimkhuler, 1996]
- Variable time step [Monaghan and Kos, 1999]
- Kernel functions:
  - Cubic Spline kernel [Monaghan and Lattanzio, 1985]
  - Quintic Wendland kernel [Wendland, 1995]
  - Gaussian kernel
- Density treatment:
  - Delta-SPH formulation [Molteni and Colagrossi, 2009]
- Viscosity treatments:
  - Artificial viscosity [Monaghan, 1992]
  - Laminar viscosity + SPS turbulence model [Dalrymple and Rogers, 2006]
- Weakly compressible approach using Tait's equation of state

3. SPH formulation: DualSPHysics v4.2

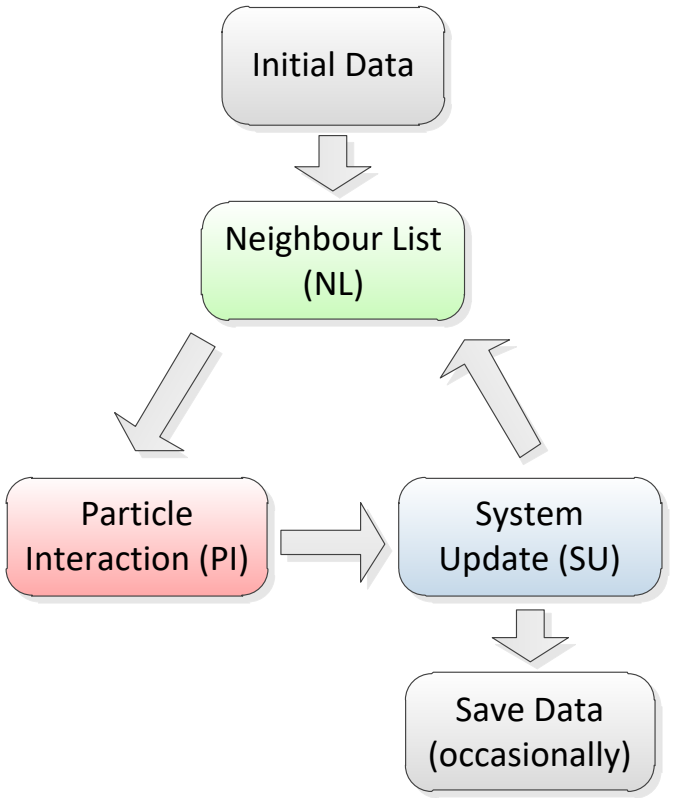
- Shifting algorithm [Lind et al., 2012]
- Dynamic boundary conditions [Crespo et al., 2007]
- Floating objects [Monaghan et al., 2003]
- Periodic open boundaries [Gómez-Gesteira et al., 2012a]
- Coupling with Discrete Element Method (DEM) [Canelas et al., 2016]
- External body forces [Longshaw and Rogers, 2015]
- Double precision [Domínguez et al., 2013c]
- Wave generation [Biesel and Suquet, 1951; Madsen, 1971; Liu and Frigaard, 2001]
- Piston- and flap-type long-crested second-order wave generation
- Passive and Active Wave Absorption System [Altomare et al., 2017]
- Multi-phase (soil-water) [Fourtakas and Rogers, 2016] – executable only
- Multi-phase (liquid-gas) [Mokos et al., 2015]

Outline

1. Introduction
    1.1. Smooth Particle Hydrodynamics
    1.2. Why is SPH too slow?
    1.3. High Performance Computing (HPC)
2. DualSPHysics project
3. SPH formulation
4. DualSPHysics code
    4.1. Source files
    4.2. Object-Oriented Programming
    4.3. Execution diagram
5. **DualSPHysics implementation**
    5.1. CPU acceleration
    5.2. GPU acceleration
    5.3. Multi-GPU acceleration

5. DualSPHysics implementation

For the implementation of SPH, the code is organised in **3 main steps** that are repeated each time step till the end of the simulation.



Neighbour list (NL):
Particles are grouped in cells and reordered to optimise the next step.

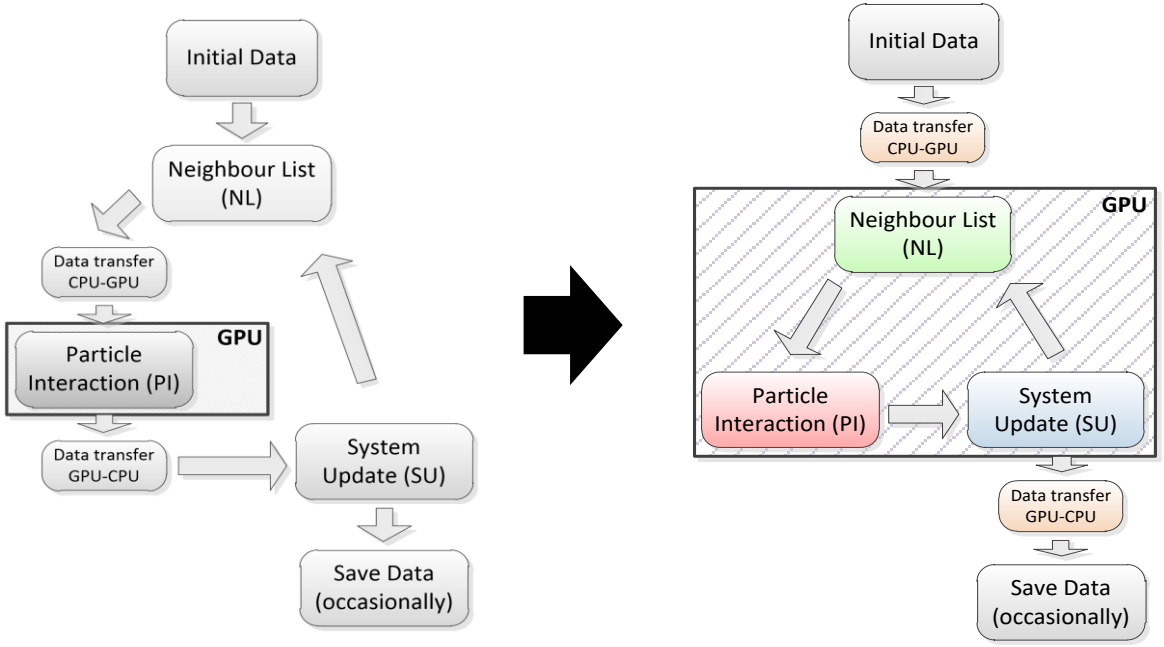
Particle interactions (PI):
Forces between particles are computed, solving momentum and continuity equations.
This step takes **more than 95%** of execution time.

System update (SU):
Starting from the values of computed forces, the magnitudes of the particles are updated for the next instant of the simulation.

5.2. GPU acceleration

Full GPU implementation

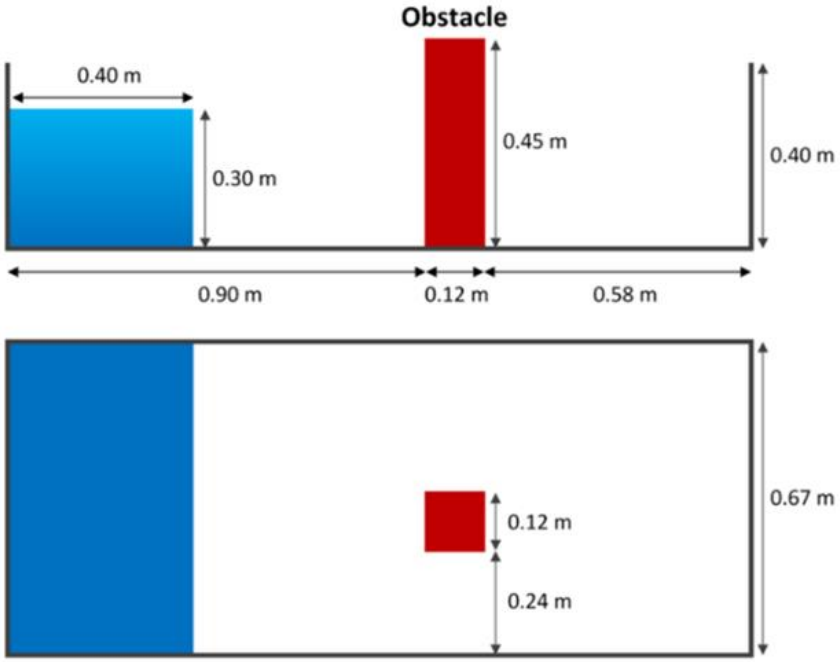
- DualSPHysics was implemented using the CUDA programming language to run SPH method on Nvidia GPUs.
- **GPU is used in all steps** (Neighbour List, Particle Interaction and System Update).
- This approach is the most efficient since:
  - All particle data is kept in GPU memory and the **transfers CPU-GPU are removed**.
  - **Neighbour List and System Update are parallelized**, obtaining a speedup also in this part of the code.

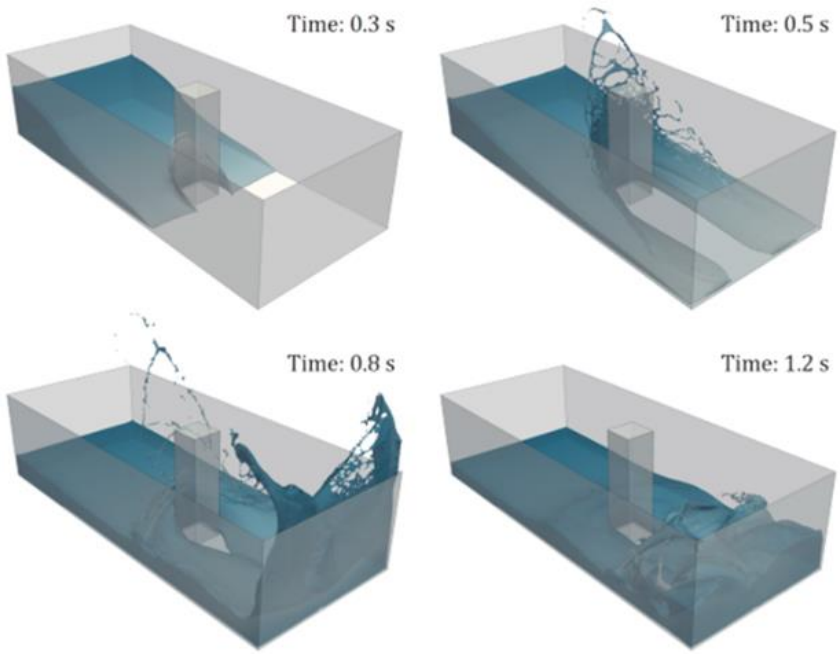


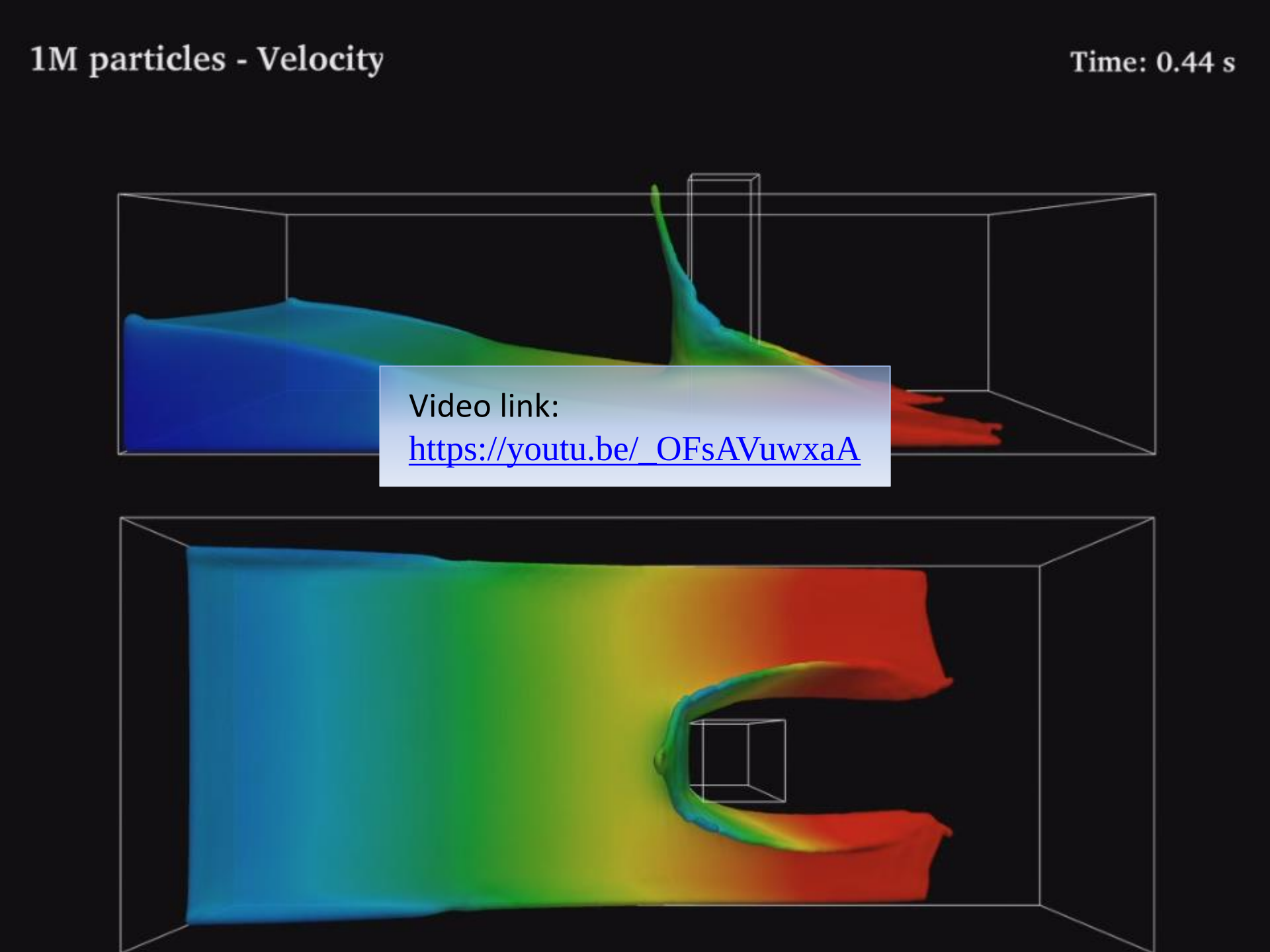
5.2. SPH implementation: CPU acceleration

Testcase for results

- **Dam break flow impacting on a structure** (experiment of Yeh and Petroff at the University of Washington).
- Physical time of simulation is **1.5 seconds**.

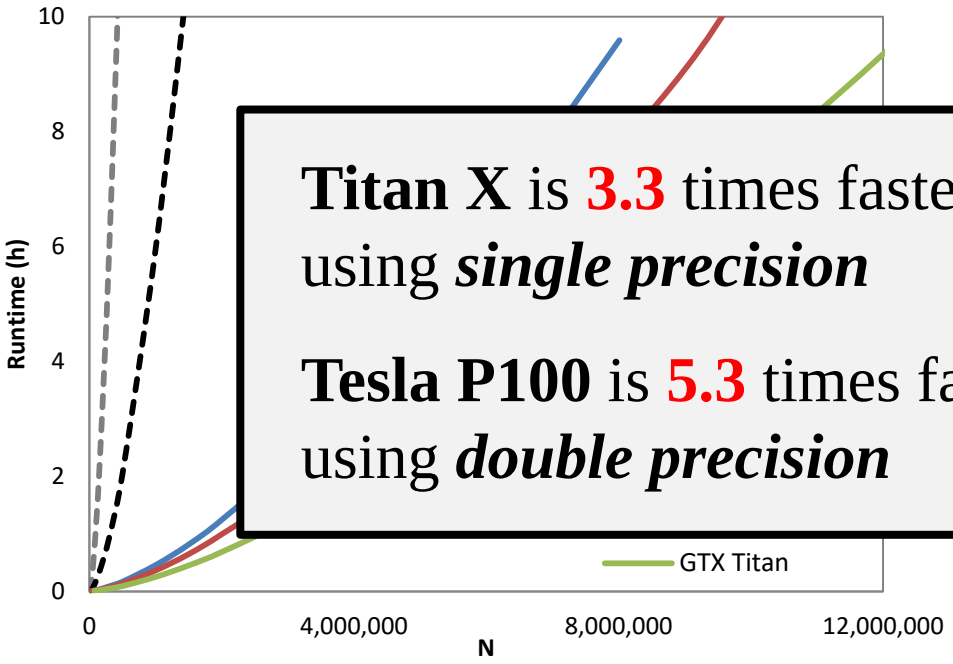




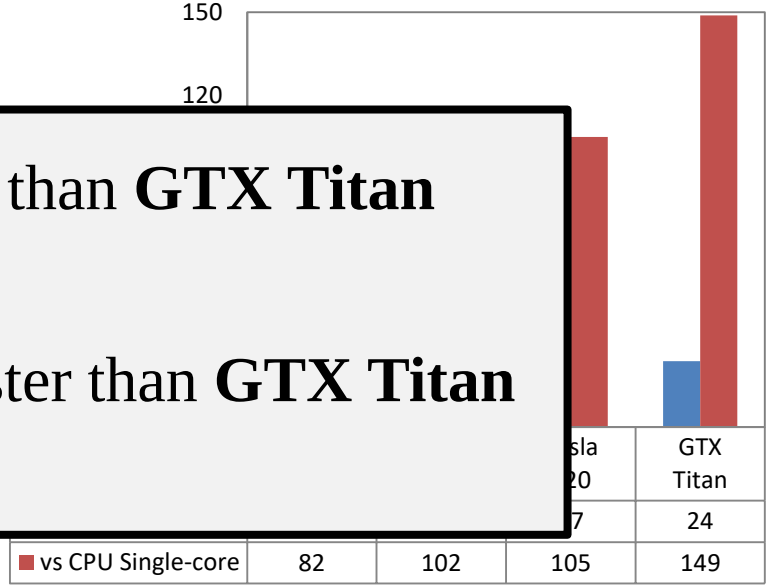

1M particles - Velocity
Time: 0.44 s
Video link:
https://youtu.be/_OFsAVuwxaA

5.2. GPU acceleration: Results

Runtime for CPU and different GPU cards.



Speedups of GPU against CPU simulating 1 million particles.



Titan X is **3.3** times faster than **GTX Titan** using ***single precision***

Tesla P100 is **5.3** times faster than **GTX Titan** using ***double precision***

After optimising the performance of DualSPHysics on CPU and GPU...

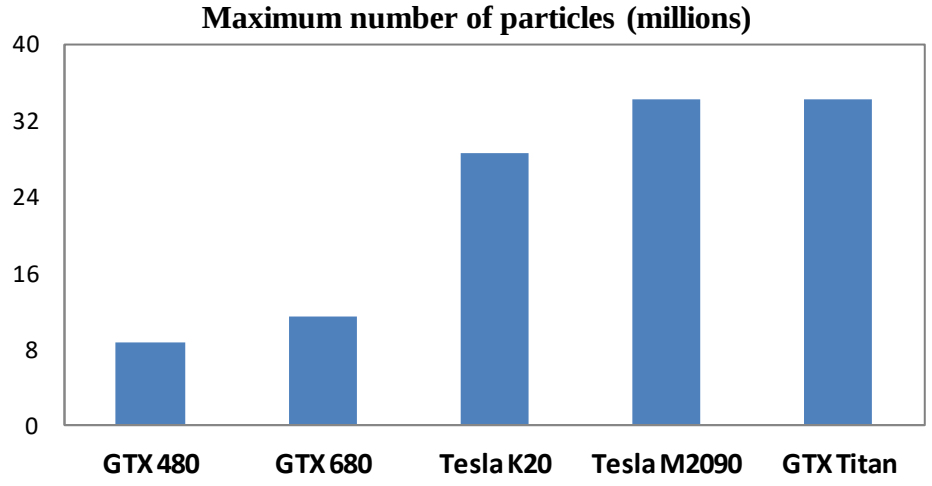
The most powerful GPU (**GTX Titan**) is **149 times faster** than CPU (single core execution) and **24 times faster** than the CPU using all 8 cores.

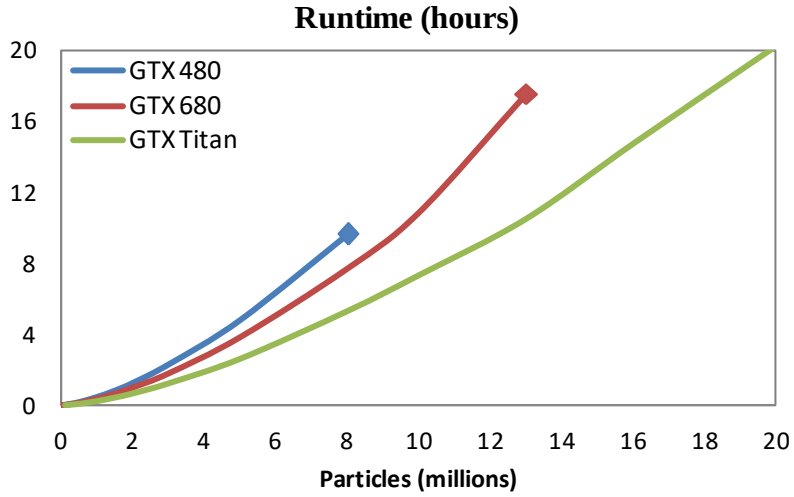
5.2. GPU acceleration: Results

The simulation of **real cases implies huge domains with a high resolution**, which implies simulating tens or hundreds of million particles.

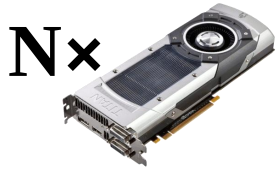
The use of one GPU presents important **limitations**:

- Maximum number of particles depends on the memory size of GPU.
- Time of execution increases rapidly with the number of particles.



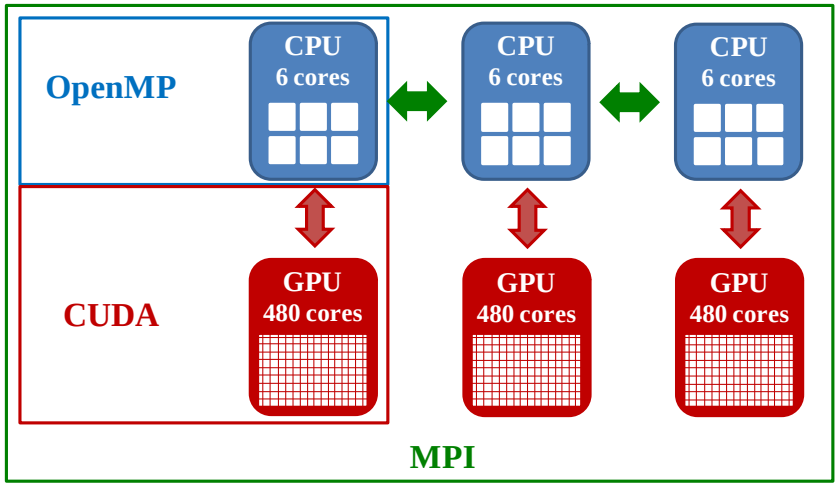


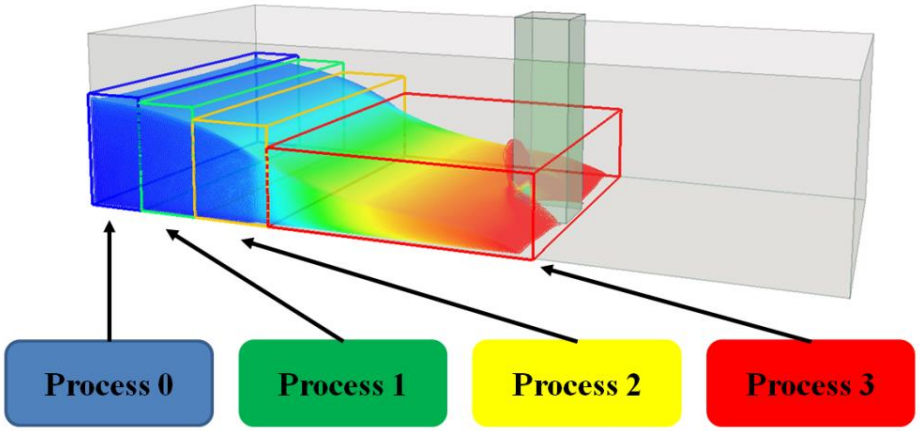
5.3. Multi-GPU implementation



MPI is used to combine resources of multiple machines connected via network.

The **physical domain** of the simulation **is divided among the different MPI processes**. Each process only needs to assign resources to manage a subset of the total amount of particles for each subdomain.





5.3. Multi-GPU implementation

The use of MPI implies an **overcost**:

- **Communication**: Time dedicated to the interchange of data between processes.
- **Synchronization**: All processes must wait for the slowest one.

Solutions:

- **Overlapping** between force computation and communications: while data is transferred between processes, each process can compute the force interactions among its own particles. In the case of GPU, the CPU-GPU transfers can also be overlapped with computation using *streams* and *pinned memory*.
- **Load balancing**. A dynamic load balancing is applied to minimise the difference between the execution times of each process.

5.3. Multi-GPU implementation

Dynamic load balancing

Due to the nature Lagrangian of the SPH method, is necessary to balance the load throughout the simulation.

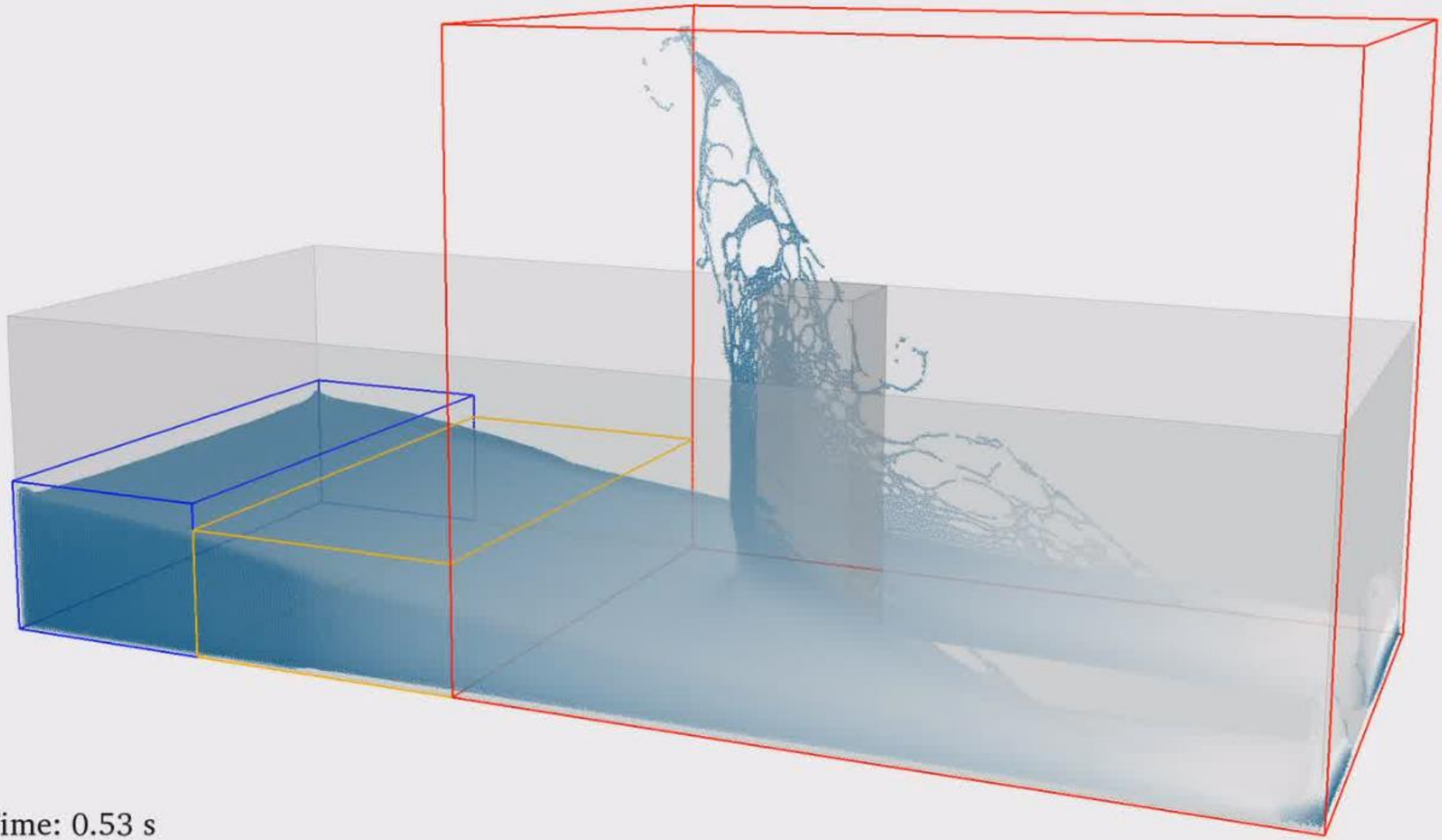
FIRST approach according to the **number of fluid particles**

> The number of particles must be redistributed after some time steps to get the workload balanced among the processes and minimise the synchronisation time.

SECOND approach according to the **required computation time** of each device

> Enables the adaptation of the code to the features of a heterogeneous cluster achieving a better performance.

GPUs: 3 x GTX480
MPI: Dynamic Balancing-Np
Particles: 6 Millions
Steps: 42,624
Runtime: 2.6 hours



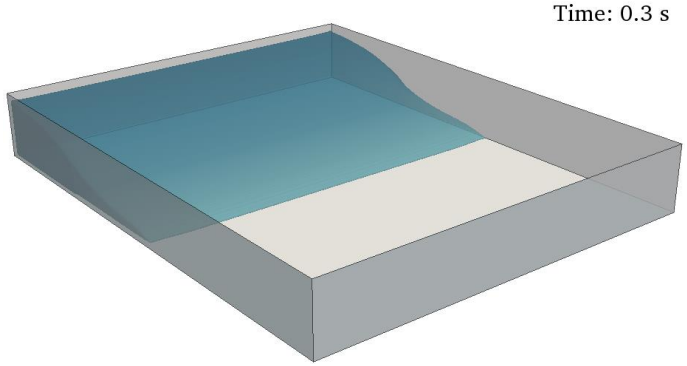
Time: 0.53 s

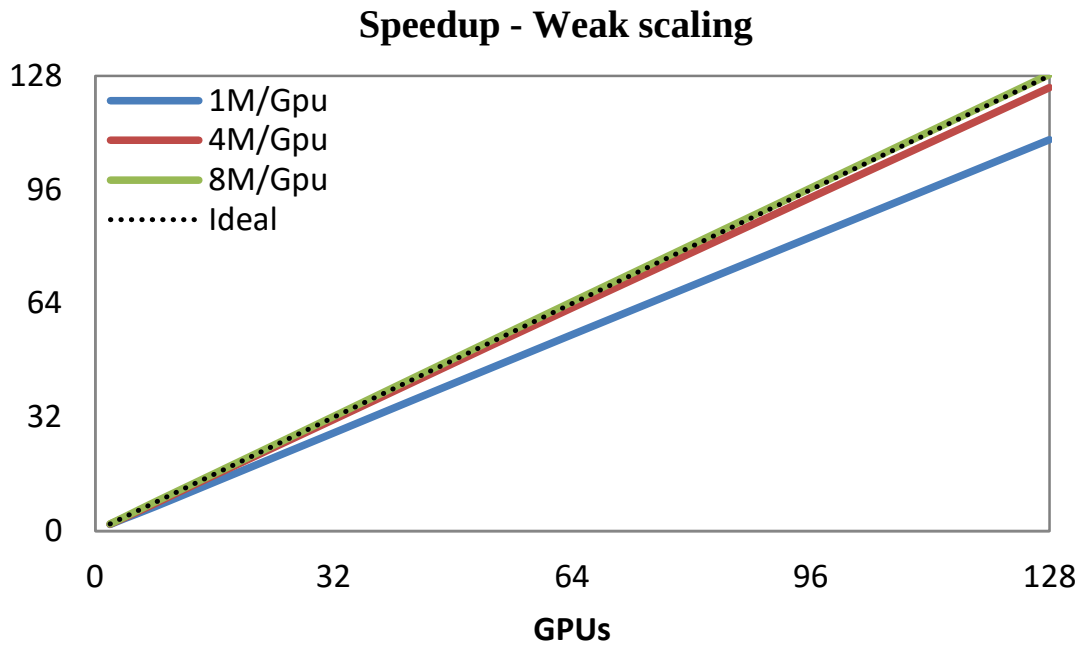
5.3. Multi-GPU: Performance

Efficiency close to 100% simulating 4M/GPU with 128 GPUs Tesla M2090 of BSC.

This is possible because the time dedicated to tasks exclusive of the multi-GPU executions (communication between processes, CPU-GPU transfers and load balancing) is minimum.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

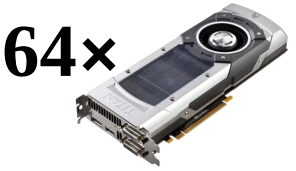




$$S(N) = \frac{T(N_{ref}) \cdot N}{T(N) \cdot N_{ref}}$$

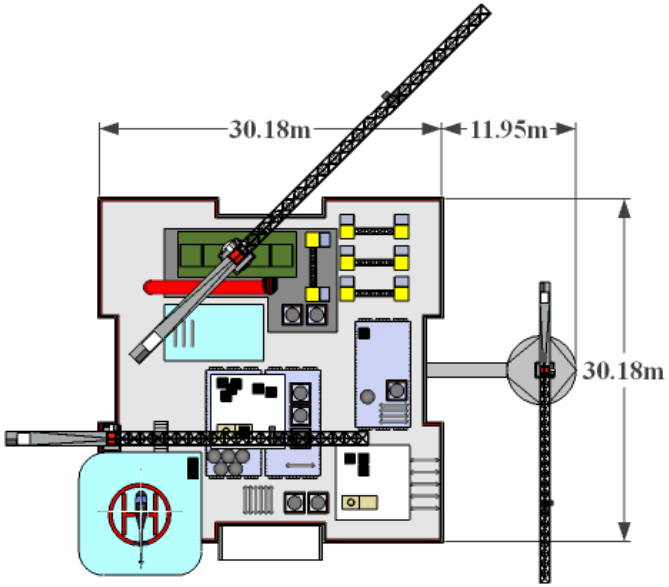
$$E(N) = \frac{S(N)}{N}$$

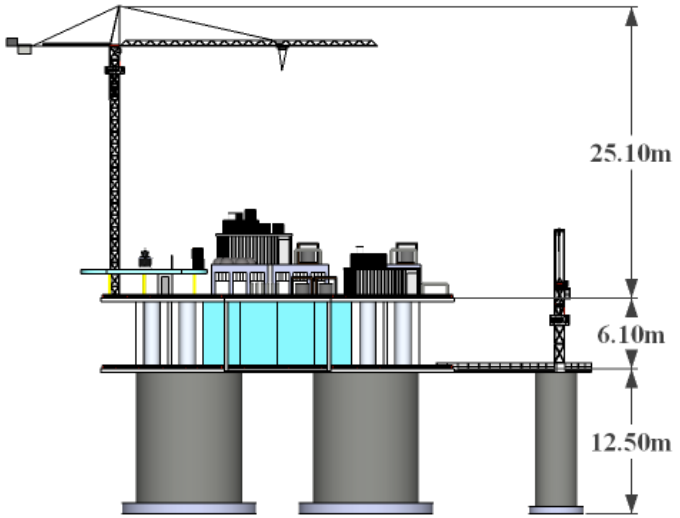
5.3. Multi-GPU: Large simulations



Simulation of 1 billion SPH particles

Large wave interaction with oil rig using 10^9 particles





dp= 6 cm, h= 9 cm
np = 1,015,896,172 particles
nf = 1,004,375,142 fluid particles
physical time= 12 sec
# of steps = 237,065 steps
runtime = 79.1 hours

using 64 GPUs Tesla M2090 of the BSC-CNS

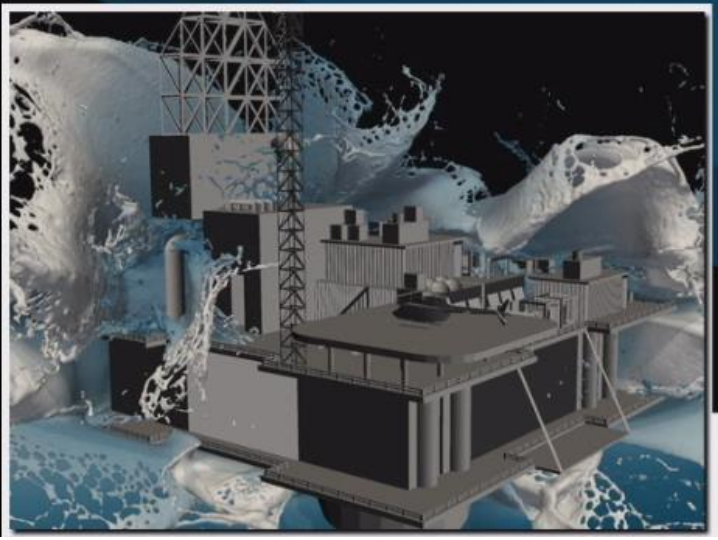
GPUs: 64x M2090 (BSC)
MPI: Dynamic balancing
Algorithm: Verlet & Wendland
Particles: 1,015 Millions
Steps: 1,237,015
Runtime: 79.1 hours
Physical time: 12 seconds

Simulación de un billón de partículas SPH

DualSPHysics

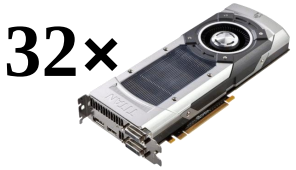
Video link:
https://youtu.be/B8mP9E75D08

Time: 3.36 s



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

5.3. Multi-GPU: Large simulations



Simulation of a real case

Using 3D geometry of the beach Itzurun in Zumaia-Guipúzcoa (Spain) in Google Earth

32 x M2090 (BSC)

Particles: **265 Millions**

Physical time: **60 seconds**

Steps: 218,211

Runtime: **246.3 hours**

32 x M2090 (BSC) Particles: **265 Millions** Physical time: **60 seconds** Runtime: **246.3 hours**

5.3. Multi-GPU: New approach

Consumers can now easily purchase desktop machines or a single compute node with 4-8 GPUs for only a few thousand euros.

New Multi-GPU code optimized for Multi-GPU machines

CUDA (and OpenMP), not MPI

Only for several GPUs in the same machine
Typical clusters have 2, 4 or 8 GPUs

Simulations with 100-200M particles
120M using 4x GTX Titan (6GB)

5.3. Multi-GPU: New approach

Consumers can now easily purchase desktop machines or a single compute node with 4-8 GPUs for only a few thousand euros.

New Multi-GPU code optimized for Multi-GPU machines

CUDA (and OpenMP), not MPI

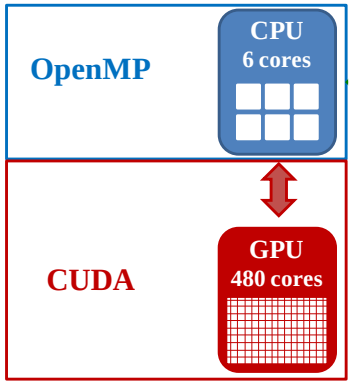
Only for several GPUs in the same machine
Typical clusters have 2, 4 or 8 GPUs

Simulations with 100-200M particles
120M using 4x GTX Titan (6GB)

5.3. Multi-GPU: New approach

DualSPHysics

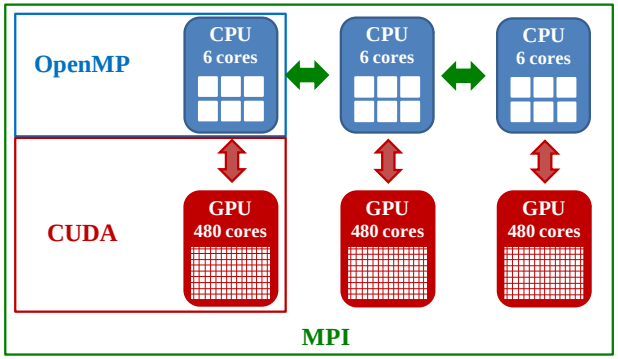
One GPU card



2011-today

Release of open-source code

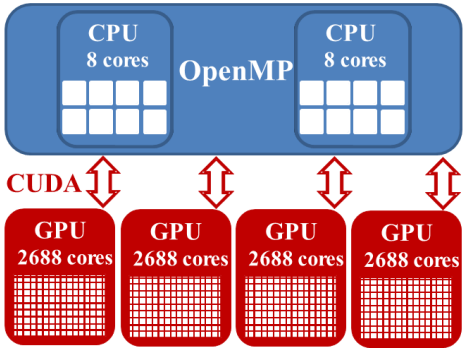
GPU clusters



since 2012

Simulations on Supercomputing Centers (BSC)

Desktop/single-node GPU



2019

To be released as open-source code

Thank you for your attention