8th DualSPHysics Workshop
27-29 January 2026 - Ourense, Spain
cpu
gpu
DualSPHysics
CIM
Centro de Investigación Mariña
UniversidadeVigo
Environmental I Physics I Laboratory
EPhysLab

Efficient multi-GPU execution of DualSPHysics: design, challenges, and results

José M. Domínguez

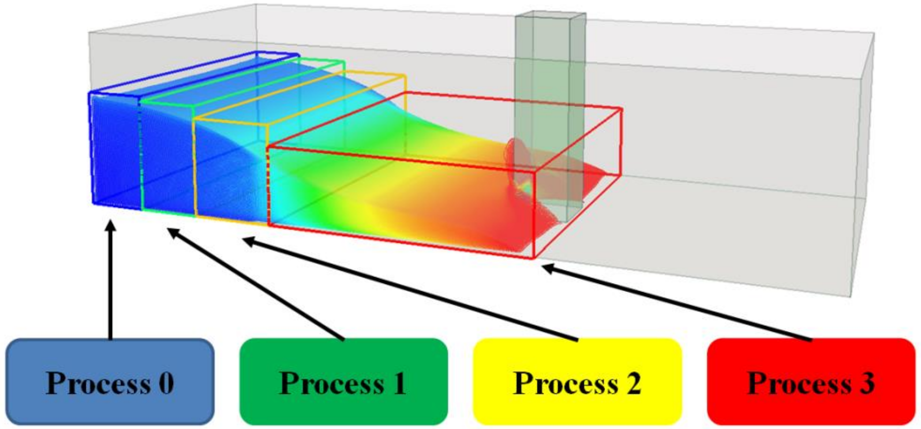
Outline

- Previous work
- New multi-GPU approach
- Implementation details
- Multi-GPU overheads
- Performance results
- Conclusions

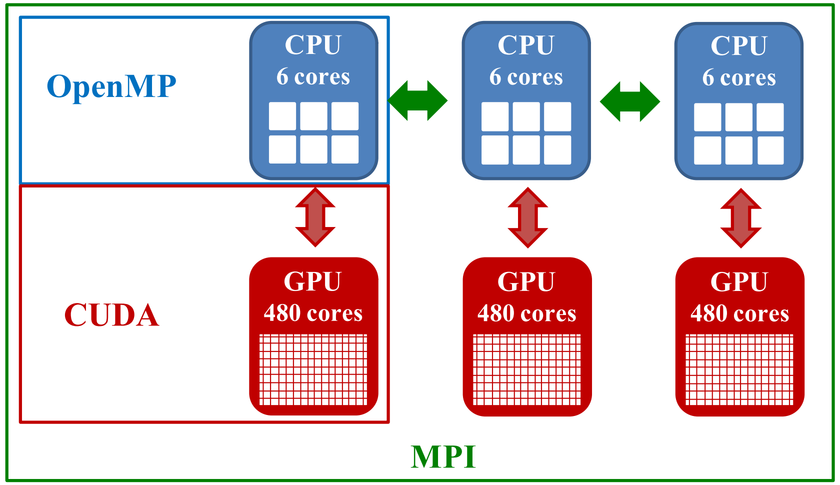
Previous work: Multi-GPU for supercomputers using MPI *(10 years ago…)*

- **DualSPHysics on GPU** made it possible increase the number of particles **from 100k-200k to around 5M**.
- Simulation of **real cases needed** higher resolution and/or larger size (**more particles**).
- However, the memory and performance of one GPU was very limited.
- **The solution** to simulate real cases was to use **many GPUs**.

Physical domain division with dynamic load balancing



- **DualSPHysics Multi-GPU** for supercomputers
- MPI to use large number of GPUs
- Physical domain decomposition
- Dynamic load balancing for homogeneous and heterogeneous clusters.

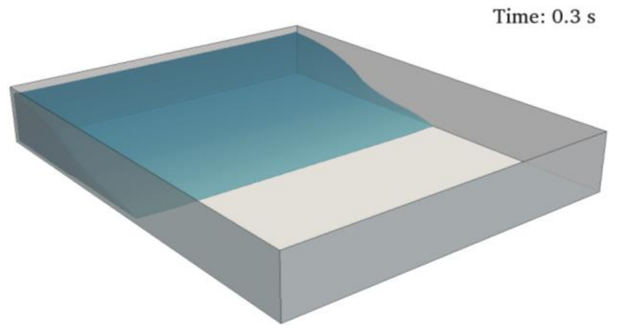


Previous work: Multi-GPU for supercomputers using MPI *(10 years ago…)*

- **DualSPHysics on GPU** made it possible increase the number of particles **from 100k-200k to around 5M**.
- Simulation of **real cases needed** higher resolution and/or larger size (**more particles**).
- However, the memory and performance of one GPU was very limited.
- **The solution** to simulate real cases was to use **many GPUs**.

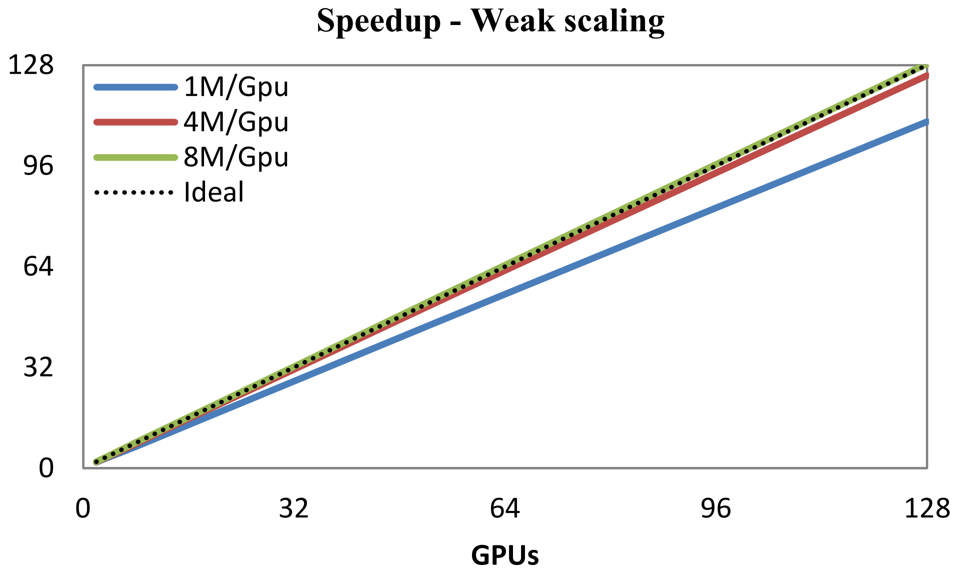
- **DualSPHysics Multi-GPU** for supercomputers
- MPI to use large number of GPUs
- Physical domain decomposition
- Dynamic load balancing for homogeneous and heterogeneous clusters.
- **Very good performance results.** Efficiency close to 100% using 128 GPUs!!

100% efficiency simulating 8M/GPU on 128 GPUs



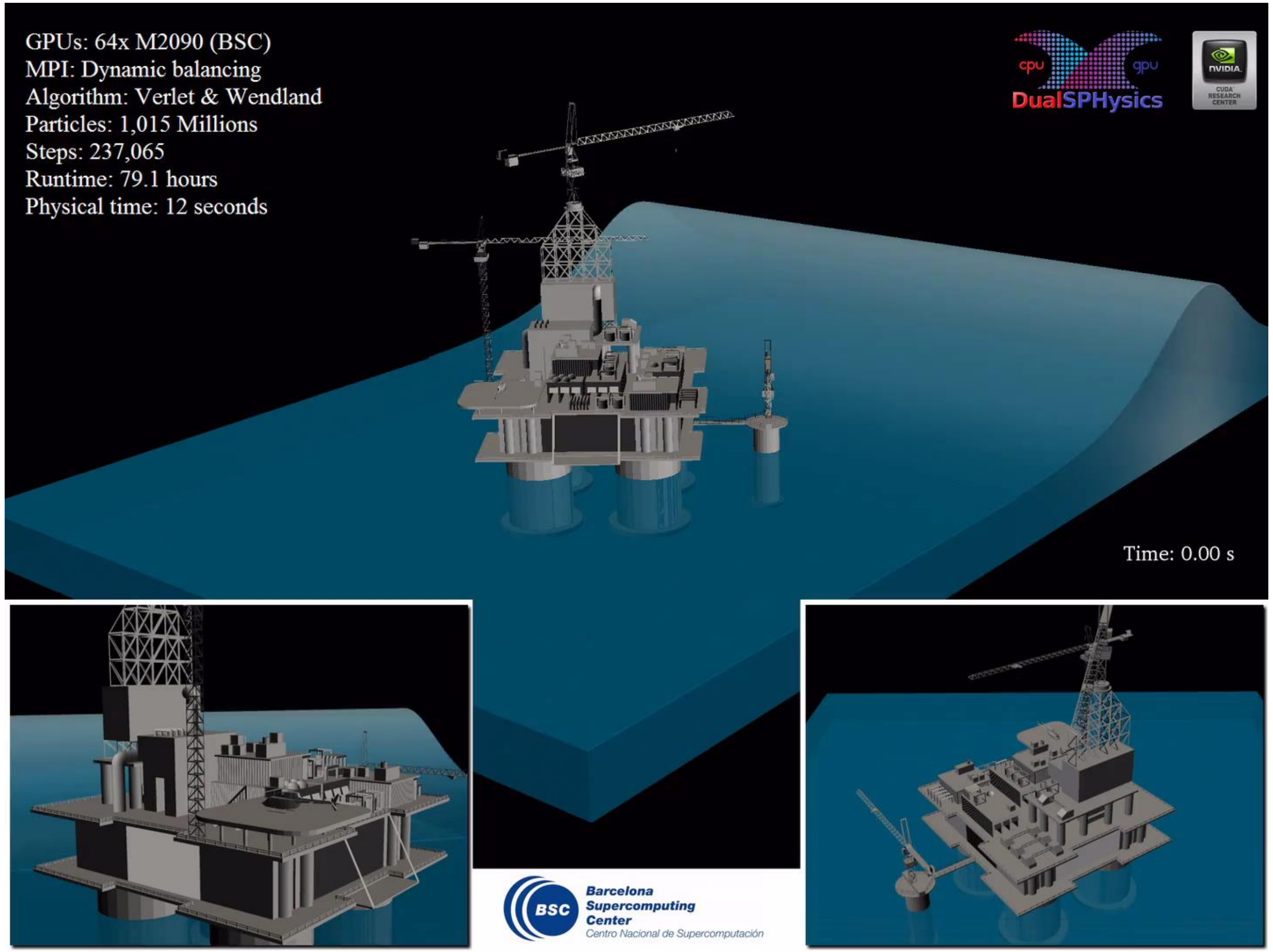
BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

128×



Previous work: Multi-GPU for supercomputers using MPI *(10 years ago…)*

Largest full SPH free-surface fluid simulation in 2013. More than 1 billion particles!!

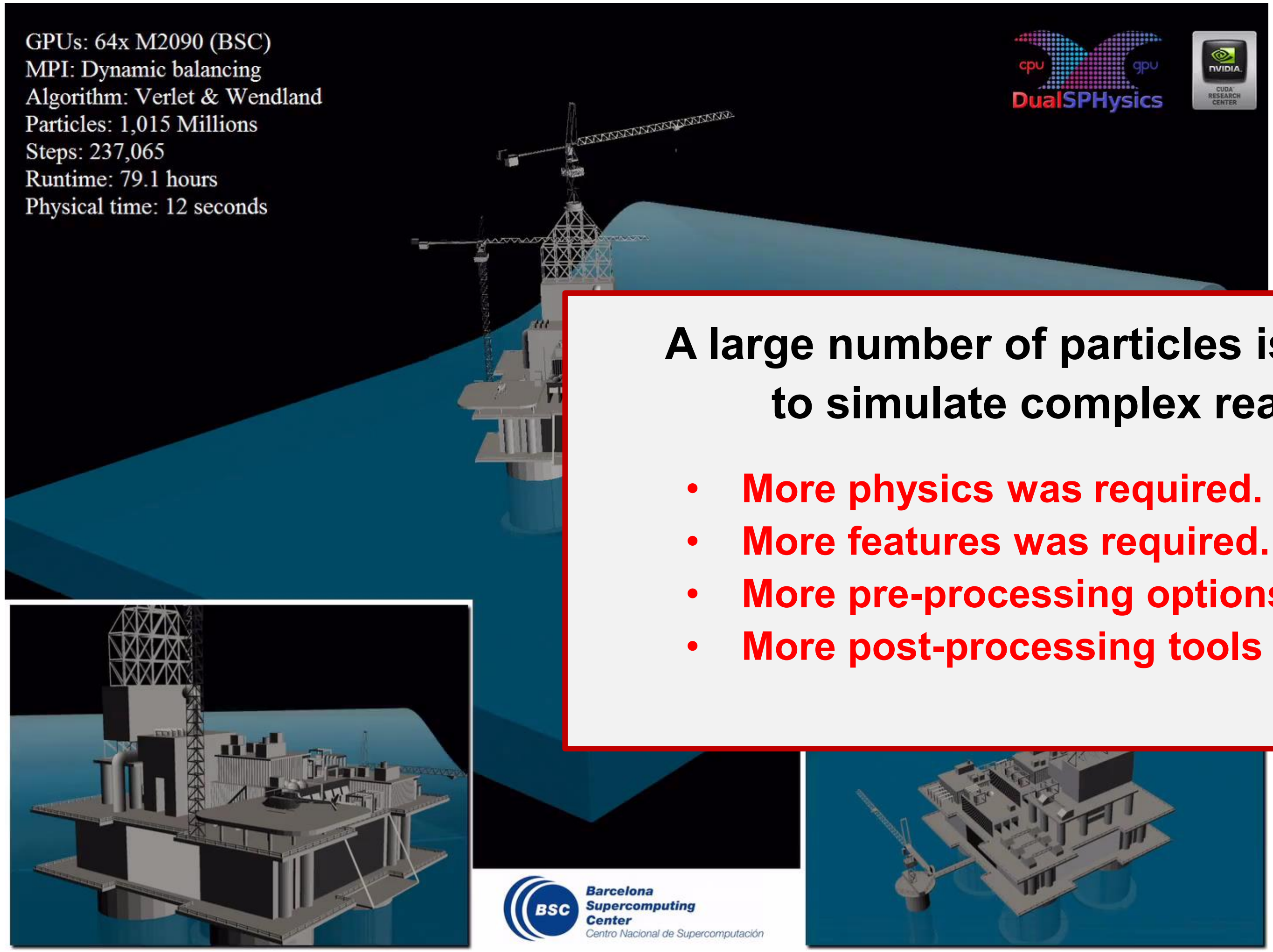

GPUs: 64x M2090 (BSC)
MPI: Dynamic balancing
Algorithm: Verlet & Wendland
Particles: 1,015 Millions
Steps: 237,065
Runtime: 79.1 hours
Physical time: 12 seconds
cpu
gpu
DualSPHysics
NVIDIA
CUDA RESEARCH CENTER
Time: 0.00 s
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

- Large wave interaction with oil rig using **10^9 particles**.
- More than 237,000 simulation steps to simulate **12 physical seconds**.
- **79.1 hours** using **64 GPUs** Tesla M2090.
- **Huge complexity** for pre-processing, simulation and post-processing.

- Very interesting challenge but not very useful.
- Access to a supercomputer is required.
- Too much effort for practical use.
- Many particles do **not allow modelling of complex problems** involving different physical phenomena.

Previous work: Multi-GPU for supercomputers using MPI *(10 years ago…)*

Largest full SPH free-surface fluid simulation in 2013. More than 1 billion particles!!



GPUs: 64x M2090 (BSC)
MPI: Dynamic balancing
Algorithm: Verlet & Wendland
Particles: 1,015 Millions
Steps: 237,065
Runtime: 79.1 hours
Physical time: 12 seconds

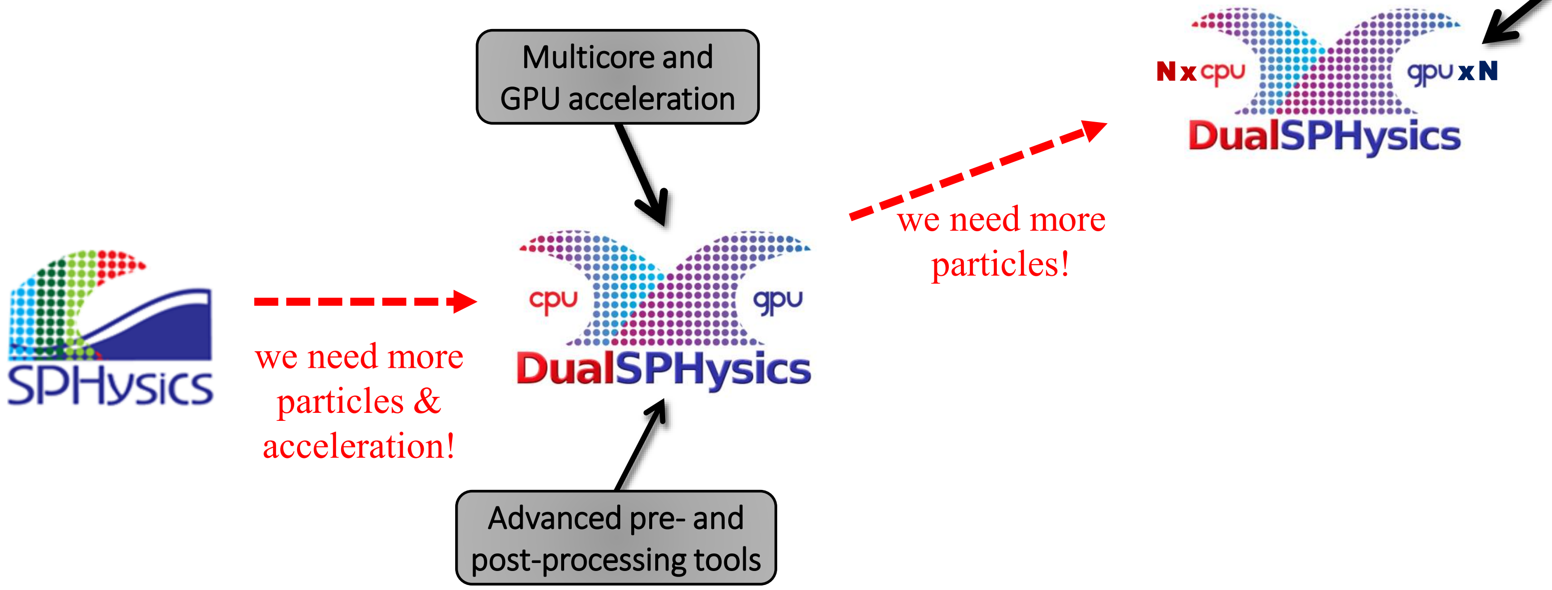
cpu
gpu
DualSPHysics
NVIDIA
CUDA RESEARCH CENTER
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

A large number of particles is not enough to simulate complex real cases.

- More physics was required.
- More features was required.
- More pre-processing options were required.
- More post-processing tools were required.

- Large wave interaction with oil rig using **10^9 particles**.
- More than 237,000 simulation steps to simulate **12 physical seconds**.
- **79.1 hours** using **64 GPUs** Tesla M2090.
- **plexity** for pre-processing, and post-processing.
- sting challenge but not very
- a supercomputer is required.
- Too much effort for practical use.
- Many particles do **not allow modelling of complex problems** involving different physical phenomena.

DualSPHysics evolution



we need more particles & acceleration!

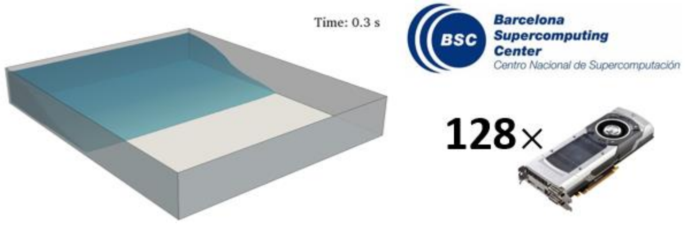
Multicore and GPU acceleration

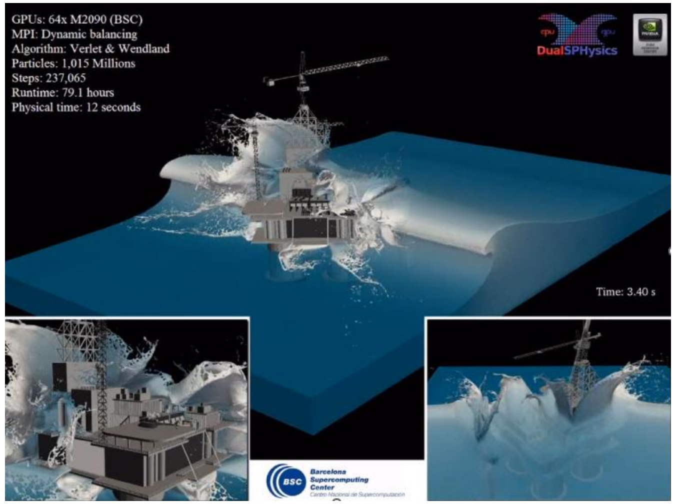
Advanced pre- and post-processing tools

we need more particles!

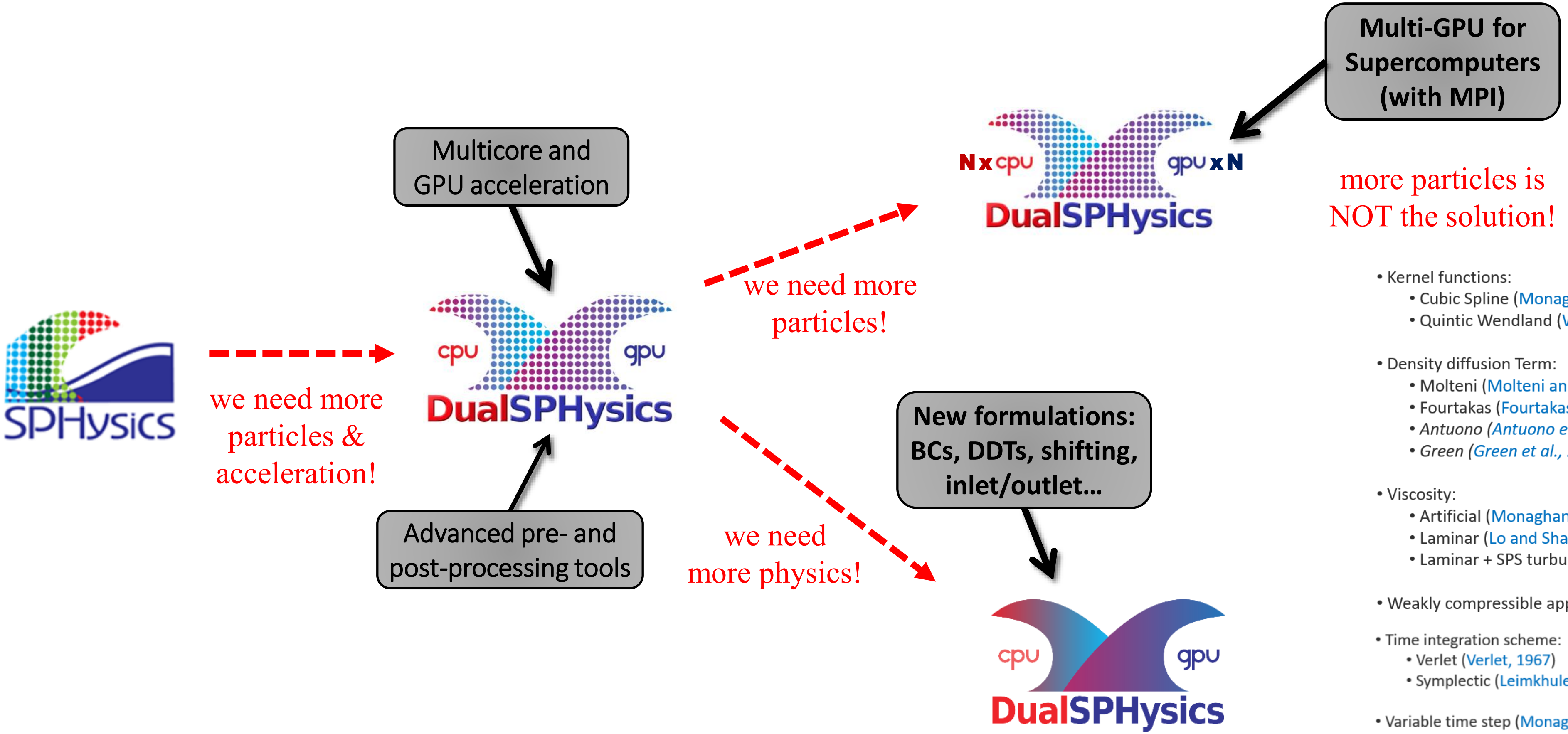
Multi-GPU for Supercomputers (with MPI)

more particles is NOT the solution!


Time: 0.3 s
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación
128×

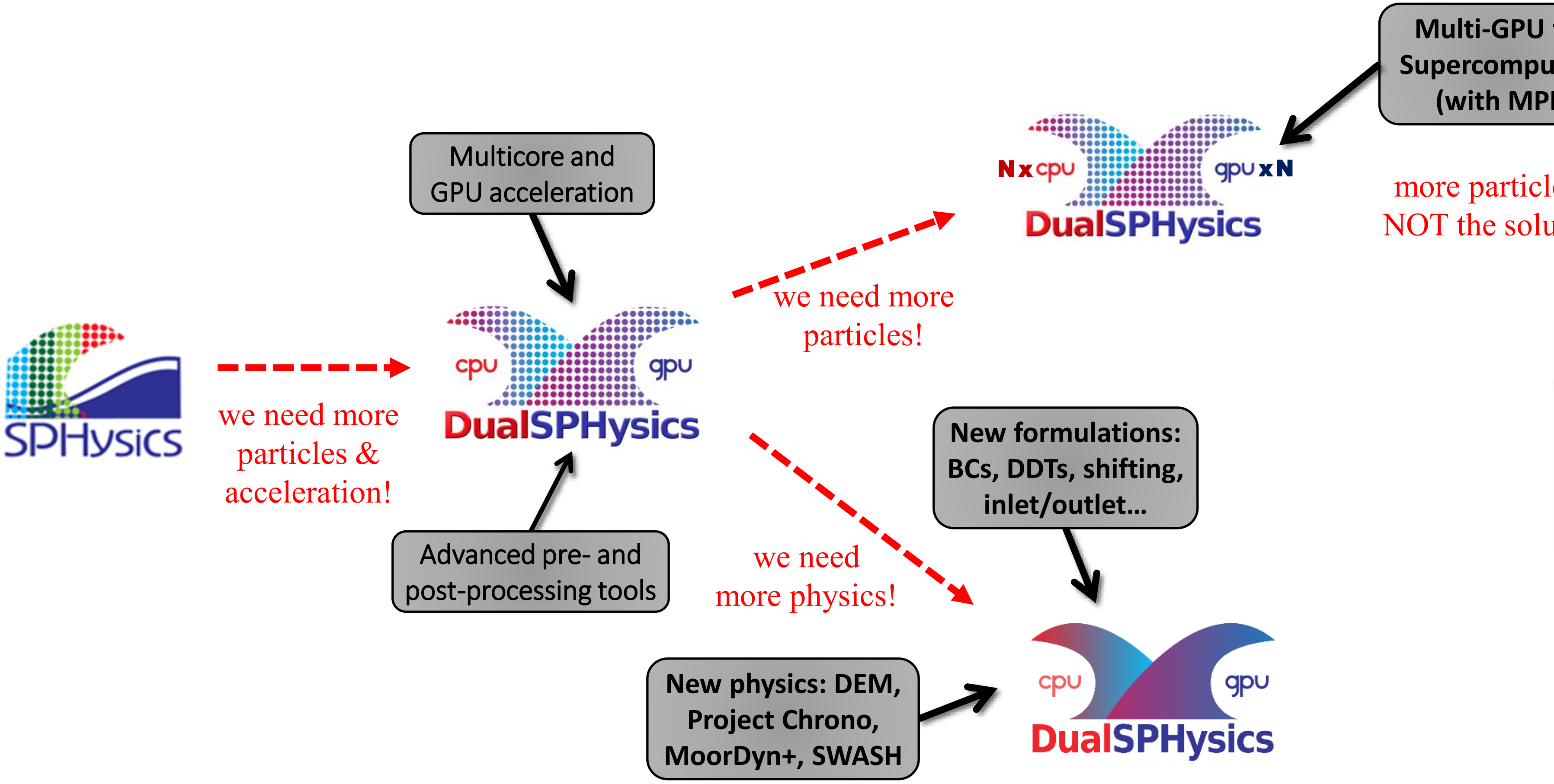

GPUs: 64x M2090 (BSC)
MPI: Dynamic balancing
Algorithm: Verlet & Wendland
Particles: 1,015 Millions
Steps: 237,065
Runtime: 79.1 hours
Physical time: 12 seconds

DualSPHysics evolution


SPHysics
we need more particles & acceleration!
Multicore and GPU acceleration
cpu
gpu
DualSPHysics
Advanced pre- and post-processing tools
we need more particles!
N x cpu
gpu x N
DualSPHysics
Multi-GPU for Supercomputers (with MPI)
more particles is NOT the solution!
we need more physics!
New formulations: BCs, DDTs, shifting, inlet/outlet…
cpu
gpu
DualSPHysics

- Kernel functions:
  - Cubic Spline (Monaghan and Lattanzio, 1985)
  - Quintic Wendland (Wendland, 1995)
- Density diffusion Term:
  - Molteni (Molteni and Colagrossi, 2009)
  - Fourtakas (Fourtakas et al., 2019)
  - *Antuono (Antuono et al., 2012)*
  - *Green (Green et al., 2019)*
- Viscosity:
  - Artificial (Monaghan, 1992)
  - Laminar (Lo and Shao, 2002)
  - Laminar + SPS turbulence model (Dalrymple and Rogers, 2006)
- Weakly compressible approach using Tait’s equation of state (Batchelor, 1974)
- Time integration scheme:
  - Verlet (Verlet, 1967)
  - Symplectic (Leimkhuler, 1996)
- Variable time step (Monaghan and Kos, 1999)
- Shifting algorithm (Lind et al., 2012)
- Boundary conditions:
  - Dynamic boundary conditions (Crespo et al., 2007)
  - Modified Dynamic boundary conditions (English et al., 2021)
- Floating objects (Monaghan et al., 2003)
- Periodic open boundaries (Gómez-Gesteira et al., 2012)
- Inflow-outflow boundary conditions (Tafuni et al., 2018)

DualSPHysics evolution



we need more particles & acceleration!

Multicore and GPU acceleration

Advanced pre- and post-processing tools

we need more particles!

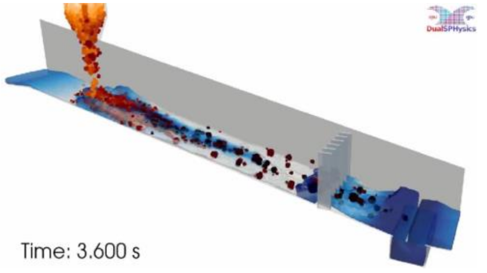
Multi-GPU for Supercomputers (with MPI)

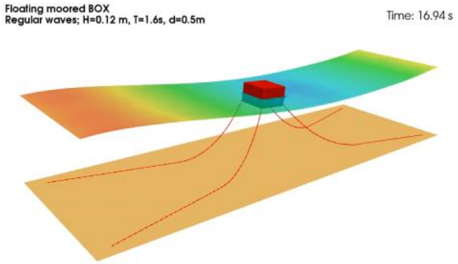
more particles is NOT the solution!

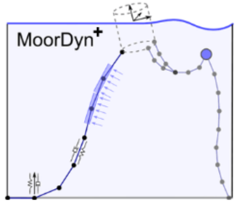
we need more physics!

New formulations: BCs, DDTs, shifting, inlet/outlet…

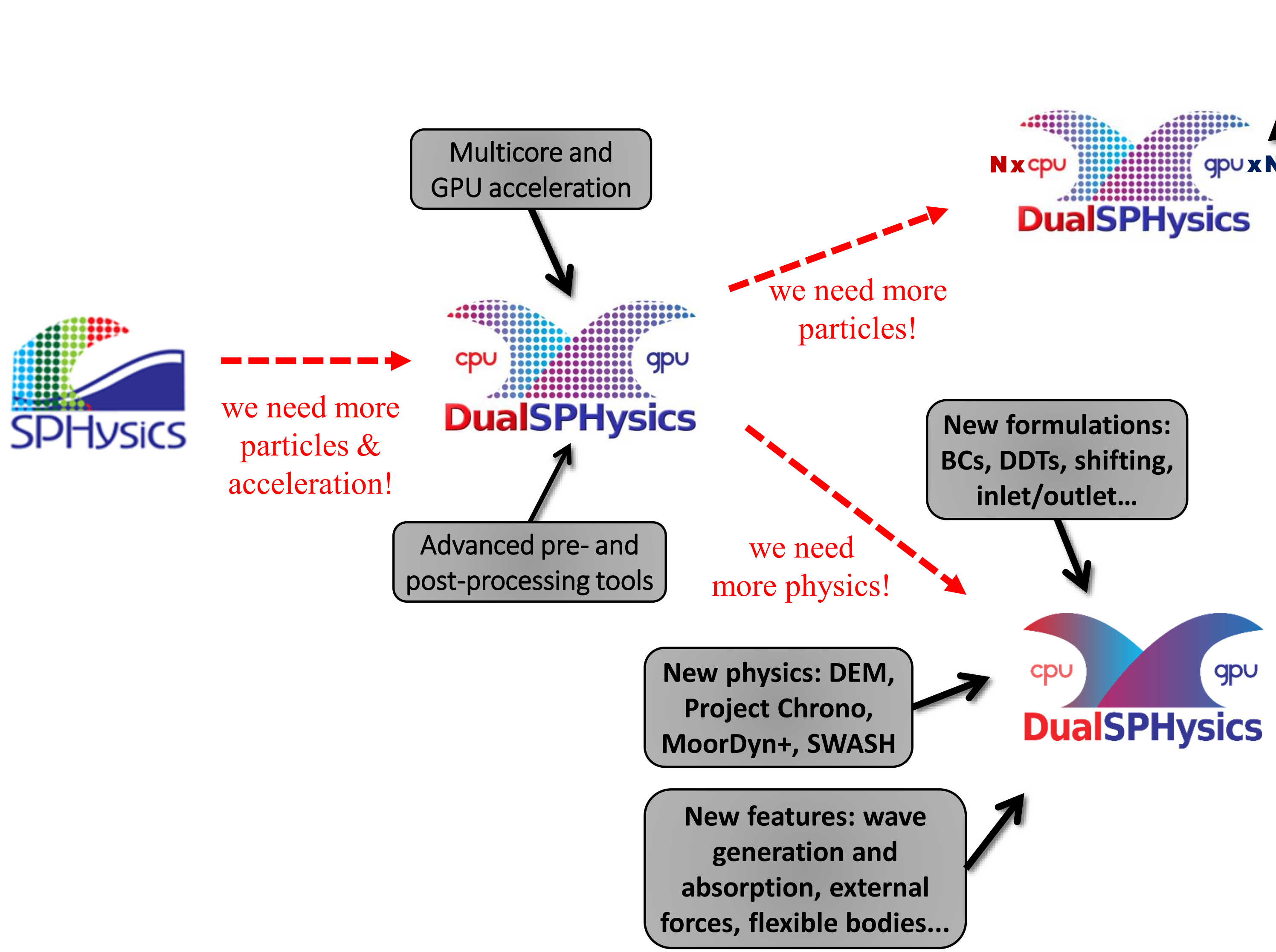
New physics: DEM, Project Chrono, MoorDyn+, SWASH







DualSPHysics evolution



Multicore and GPU acceleration

Advanced pre- and post-processing tools

we need more particles & acceleration!

we need more particles!

we need more physics!

New formulations: BCs, DDTs, shifting, inlet/outlet…

New physics: DEM, Project Chrono, MoorDyn+, SWASH

New features: wave generation and absorption, external forces, flexible bodies…

Multi-GPU for Supercomputers (with MPI)

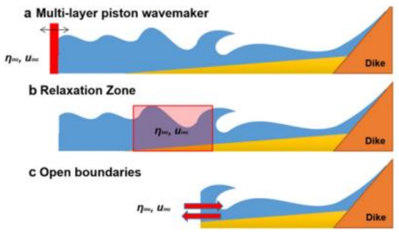
more particles is NOT the solution!

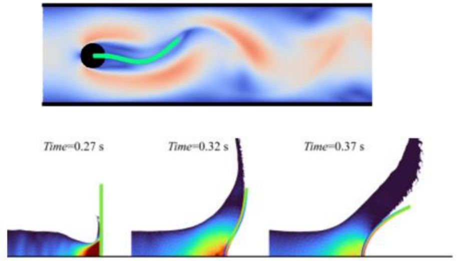
Ocean engineering features:

- Piston- and flap-type long-crested second-order wave generation (Altomare et al., 2017)
- Solitary waves (Domínguez et al., 2019)
- Focused waves (Whittaker et al., 2017)
- Passive and Active Wave Absorption System (Altomare et al., 2017)
- Relaxation Zone method and coupling with wave propagation models (Altomare et al., 2018)
- Non-linear wave generation and absorption using open boundaries (Verbrugghe et al., 2019)

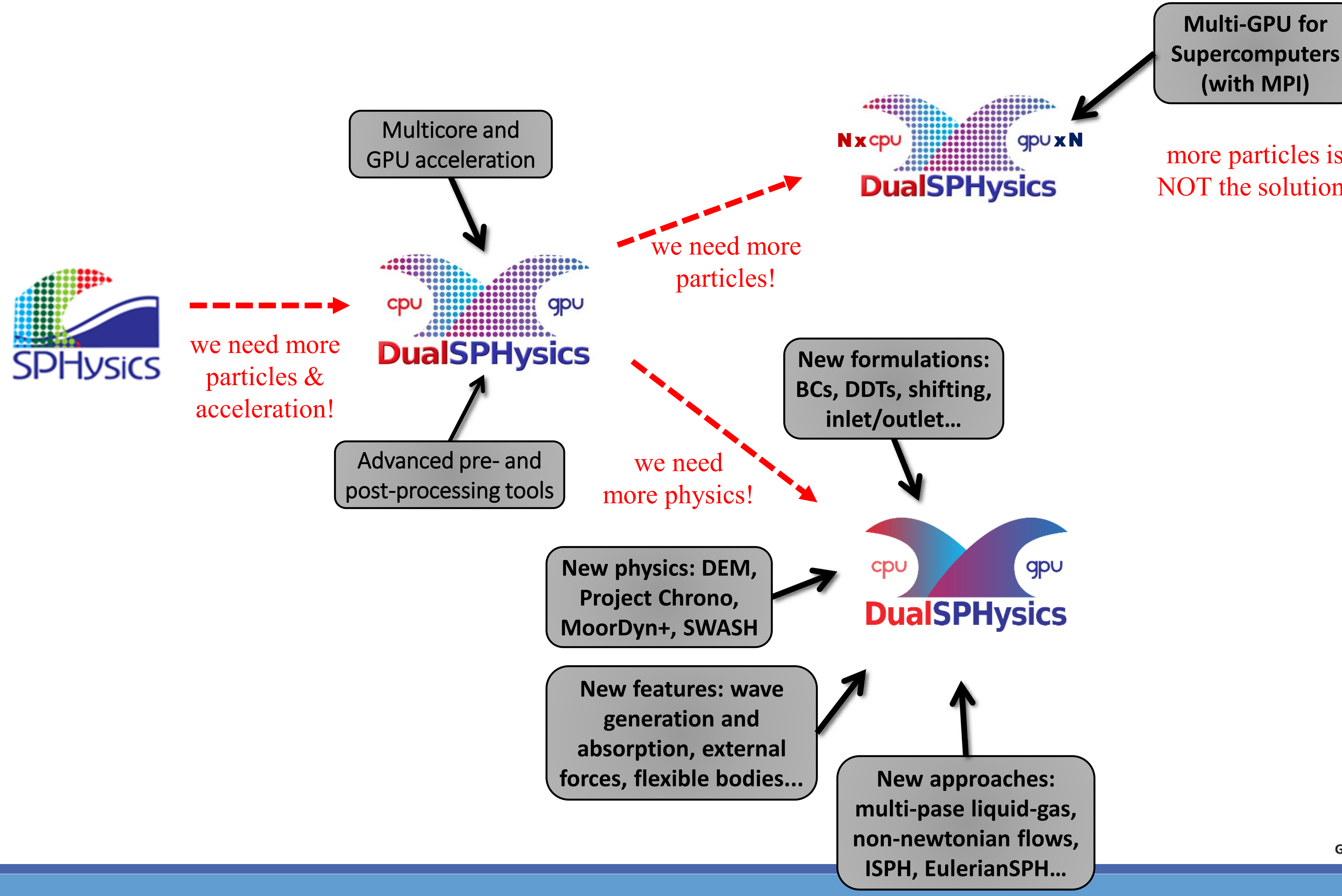
Flexible body approaches:

- Lagrangian formulation for flexible fluid-structure interaction (O'Connor et al., 2021)
- Flexible beams based on co-rotating rigid elements using Project Chrono (Capasso et al., 2022)
- SPH coupling with FEA structural solver using Project Chrono (Martínez-Estévez et al., 2023)





DualSPHysics evolution



we need more particles & acceleration!

Multicore and GPU acceleration

Advanced pre- and post-processing tools

we need more particles!

Multi-GPU for Supercomputers (with MPI)

more particles is NOT the solution!

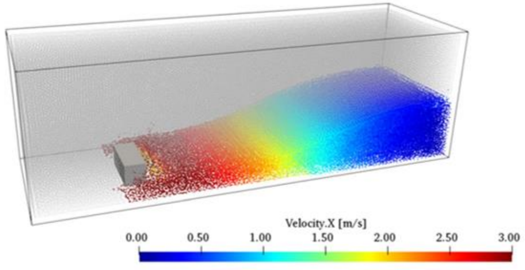
we need more physics!

New formulations: BCs, DDTs, shifting, inlet/outlet...

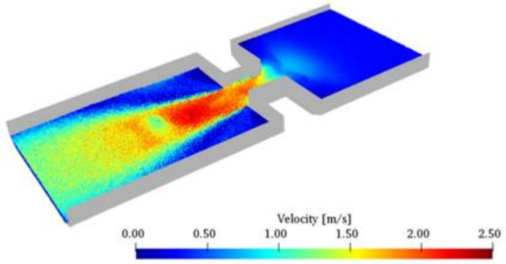
New physics: DEM, Project Chrono, MoorDyn+, SWASH

New features: wave generation and absorption, external forces, flexible bodies...

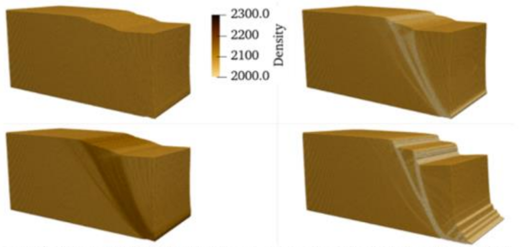
New approaches: multi-pase liquid-gas, non-newtonian flows, ISPH, EulerianSPH...



Liquid and gas: 3-D dam break impacting an obstacle

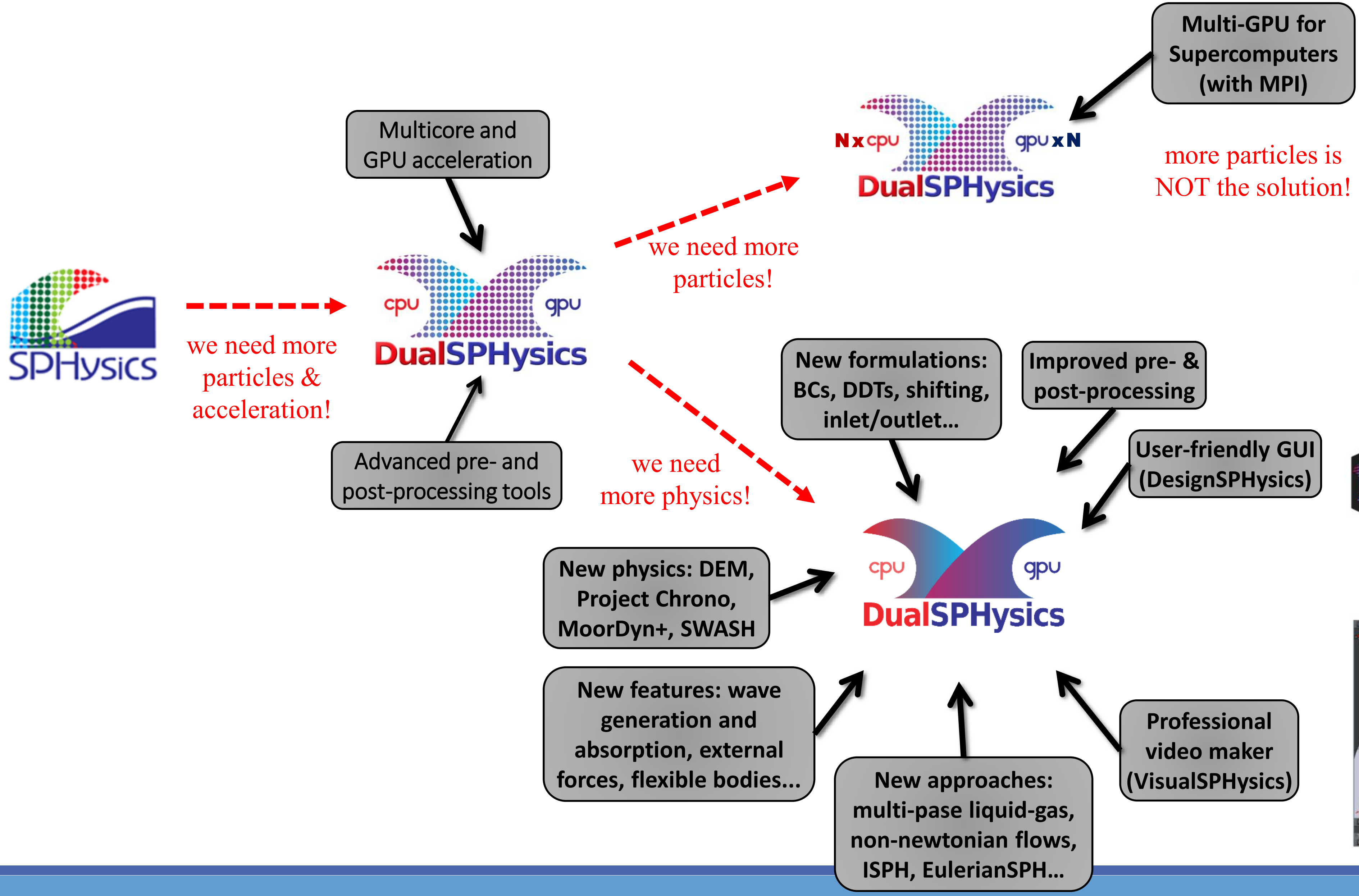


non-Newtonian flows: 3-D dam break over an erodible bed



Geomechanics formulations: 3-D collapse of cohesive granular materials

DualSPHysics evolution



we need more particles & acceleration!

Multicore and GPU acceleration

Advanced pre- and post-processing tools

we need more particles!

Multi-GPU for Supercomputers (with MPI)

more particles is NOT the solution!

we need more physics!

New formulations: BCs, DDTs, shifting, inlet/outlet…

Improved pre- & post-processing

User-friendly GUI (DesignSPHysics)

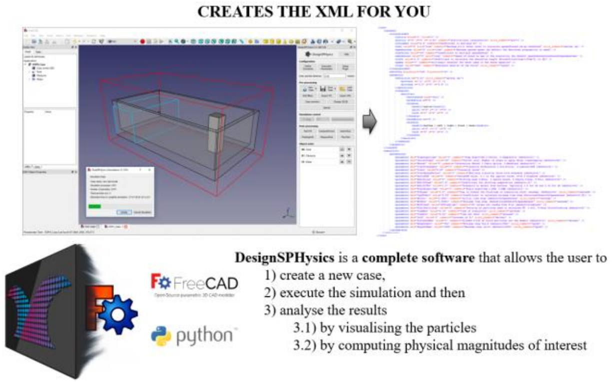
New physics: DEM, Project Chrono, MoorDyn+, SWASH

New features: wave generation and absorption, external forces, flexible bodies…

New approaches: multi-pase liquid-gas, non-newtonian flows, ISPH, EulerianSPH…

Professional video maker (VisualSPHysics)

CREATES THE XML FOR YOU



DesignSPHysics is a **complete software** that allows the user to

1) create a new case,

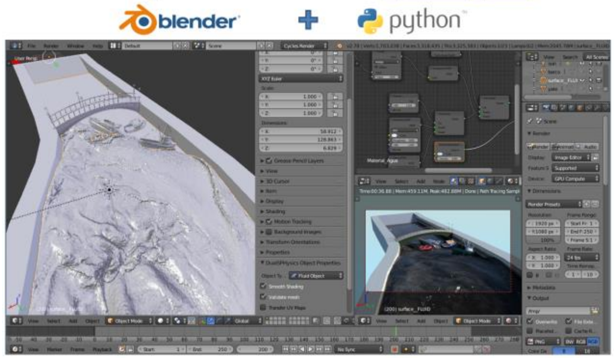
2) execute the simulation and then

3) analyse the results

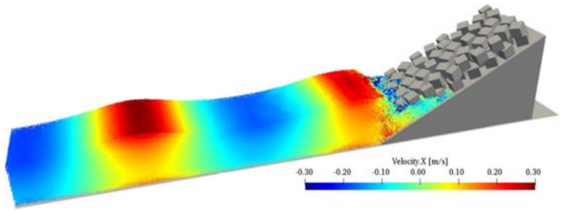
3.1) by visualising the particles

3.2) by computing physical magnitudes of interest

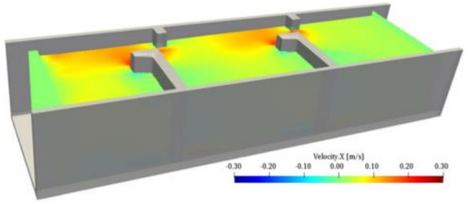
VisualSPHysics: Advanced visualisation tools: http://visual.sphysics.org/



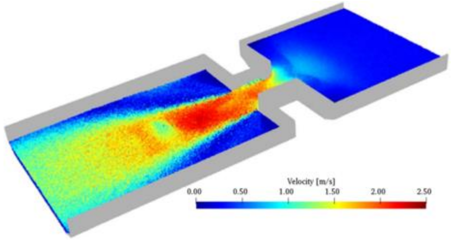
DualSPHysics is now ready for very complex multiphysics simulations!!



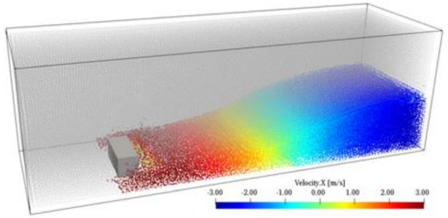
Armour breakwater



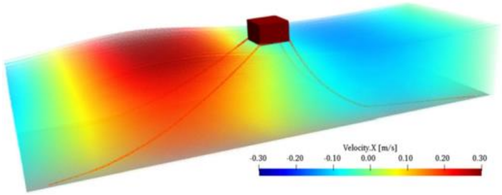
Vertical slot fishway



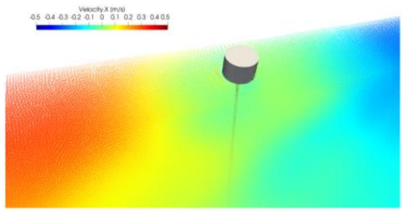
Non-Newtonian dam break



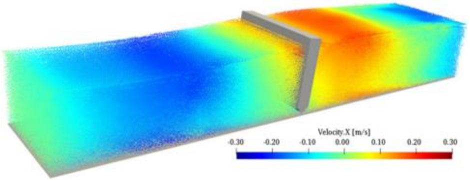
Dam break with liquid & gas



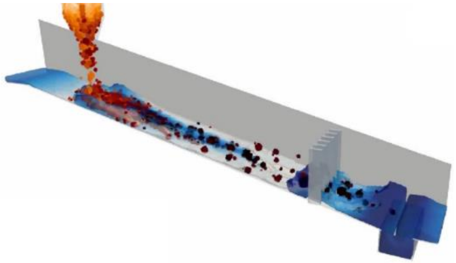
Moored floating body



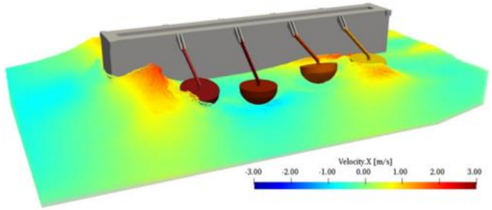
Moored point absorber



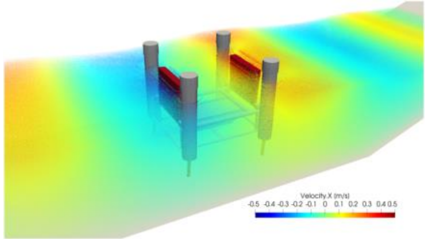
Oscillating wave surge converter



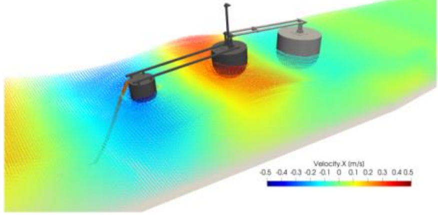
Debris flow with DEM



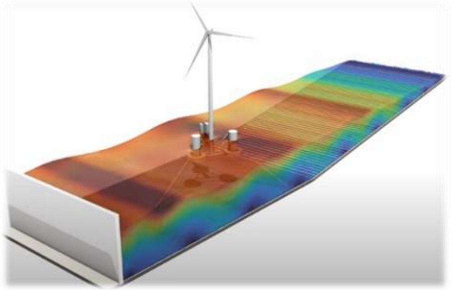
Wave star machine



Floating oscillating wave surge converter

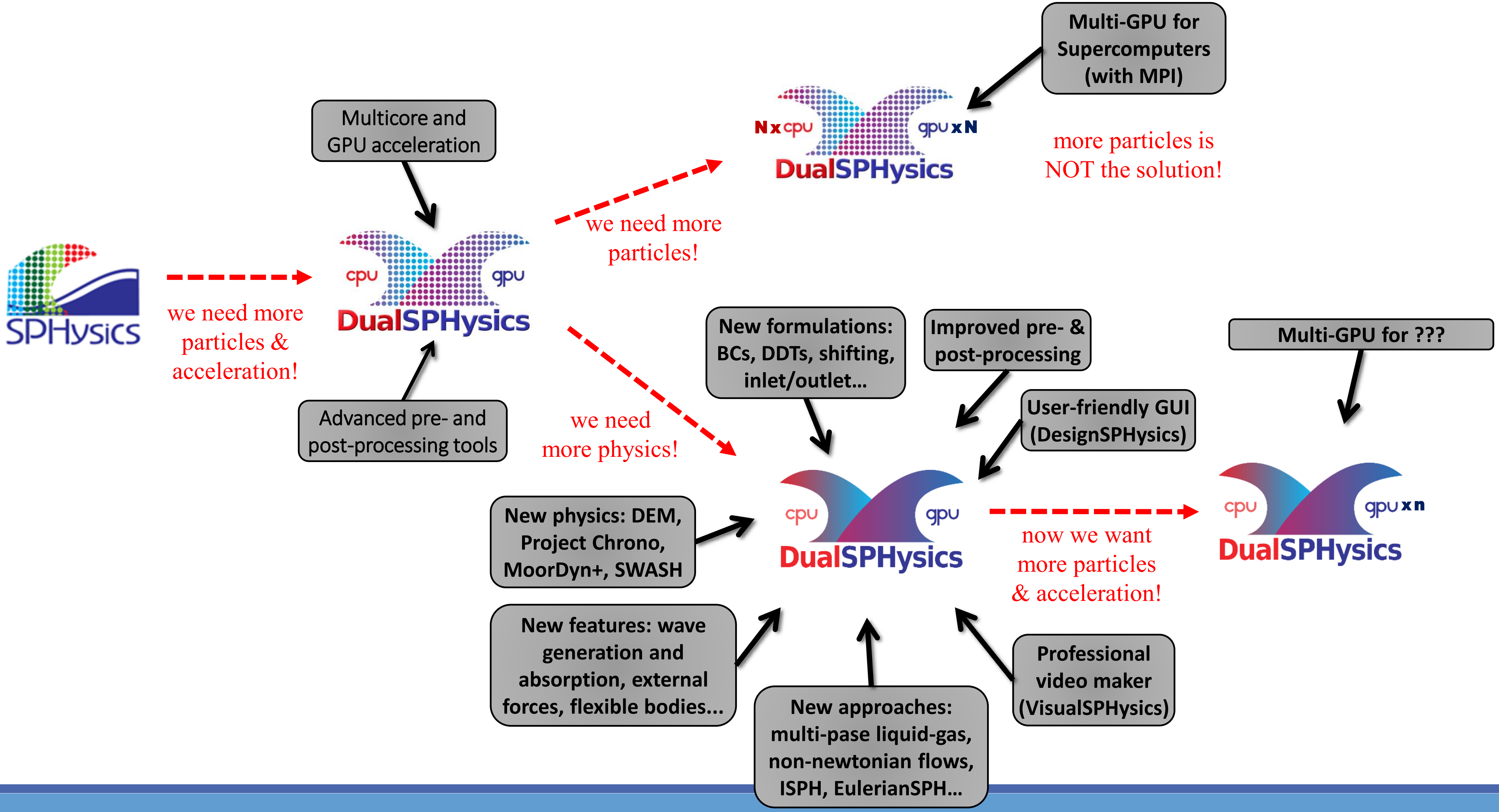


Multi-body attenuator M4



Floating wind turbine

DualSPHysics evolution



we need more particles & acceleration!

Multicore and GPU acceleration

Advanced pre- and post-processing tools

we need more particles!

Multi-GPU for Supercomputers (with MPI)

more particles is NOT the solution!

we need more physics!

New formulations: BCs, DDTs, shifting, inlet/outlet...

Improved pre- & post-processing

User-friendly GUI (DesignSPHysics)

New physics: DEM, Project Chrono, MoorDyn+, SWASH

New features: wave generation and absorption, external forces, flexible bodies...

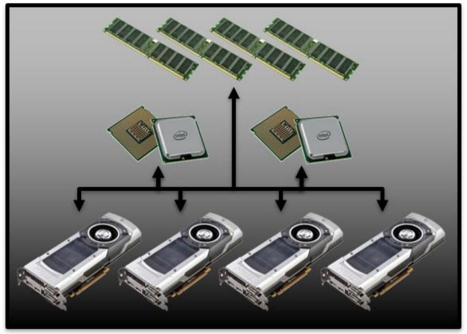
New approaches: multi-pase liquid-gas, non-newtonian flows, ISPH, EulerianSPH...

Professional video maker (VisualSPHysics)

now we want more particles & acceleration!

Multi-GPU for ???

New Multi-GPU approach for single-node



multi-GPU machine

Implementation based on C++ threads and CUDA streams (not MPI)

The target is…

- Multi-GPU **useful for researchers** using DualSPHysics (not computer engineers)
- **Full support** of all current DualSPHysics functionalities
- Aimed at **100-200M** particle simulations **without extra user effort**
- Multi-GPU to run on a workstation or computing node with 4-10 GPUs
- **Accessible hardware** for research groups with limited financial resources

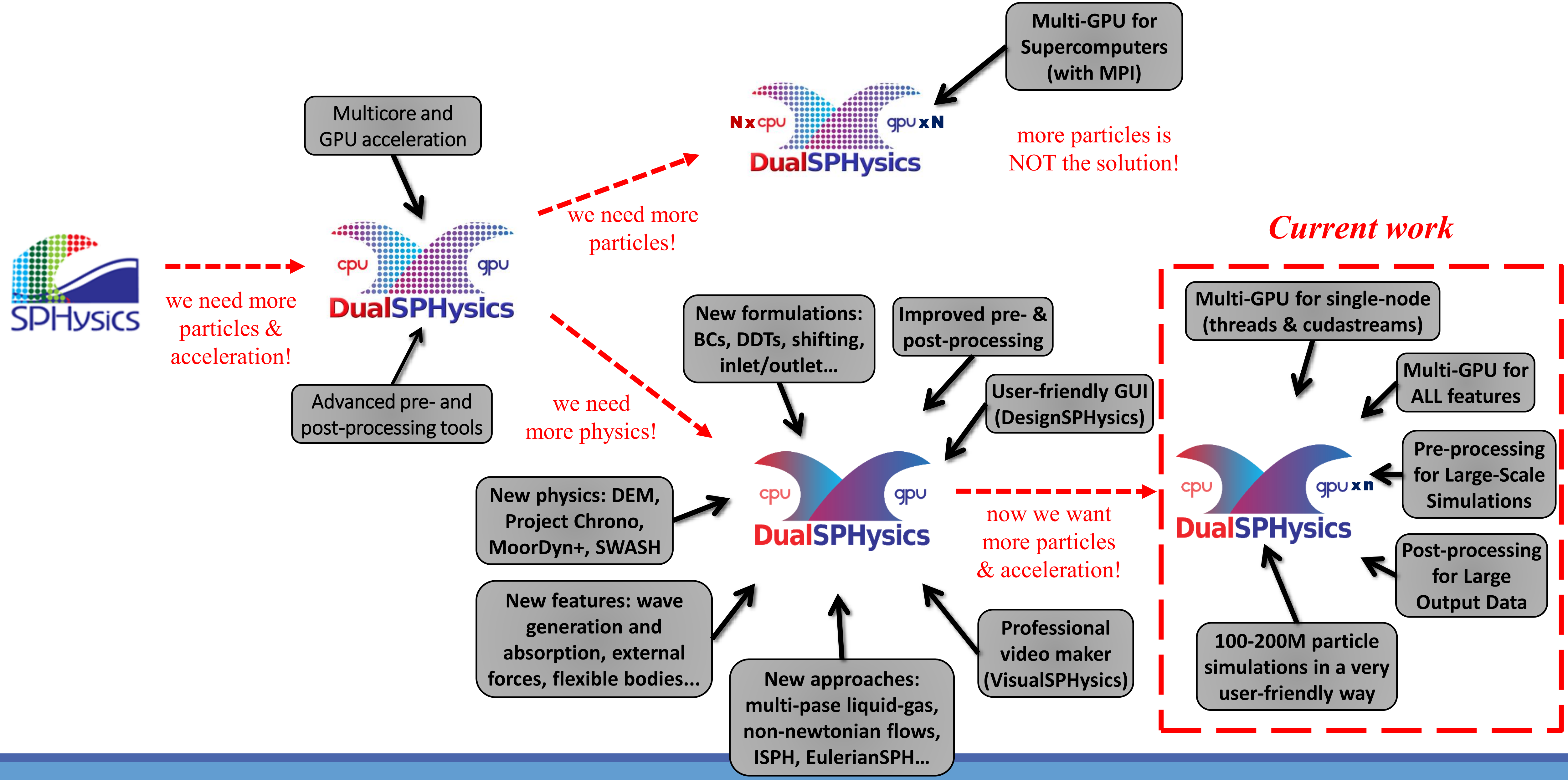
Advantages:

- More portable and easy to use in Linux & Windows
- Simpler code using shared CPU memory for main program data
- More efficient communication. MPI overhead was removed.
- Not special pre-processing and post-processing tools required (more or less...)

Drawbacks:

- Limited number of GPUs (2-10 GPUs)
- Does not work in distributed systems
- Limited size of the simulations?

DualSPHysics evolution



we need more particles & acceleration!

Multicore and GPU acceleration

Advanced pre- and post-processing tools

we need more particles!

Multi-GPU for Supercomputers (with MPI)

more particles is NOT the solution!

we need more physics!

New formulations: BCs, DDTs, shifting, inlet/outlet...

Improved pre- & post-processing

User-friendly GUI (DesignSPHysics)

New physics: DEM, Project Chrono, MoorDyn+, SWASH

New features: wave generation and absorption, external forces, flexible bodies...

New approaches: multi-pase liquid-gas, non-newtonian flows, ISPH, EulerianSPH...

Professional video maker (VisualSPHysics)

now we want more particles & acceleration!

Current work

Multi-GPU for single-node (threads & cudastreams)

Multi-GPU for ALL features

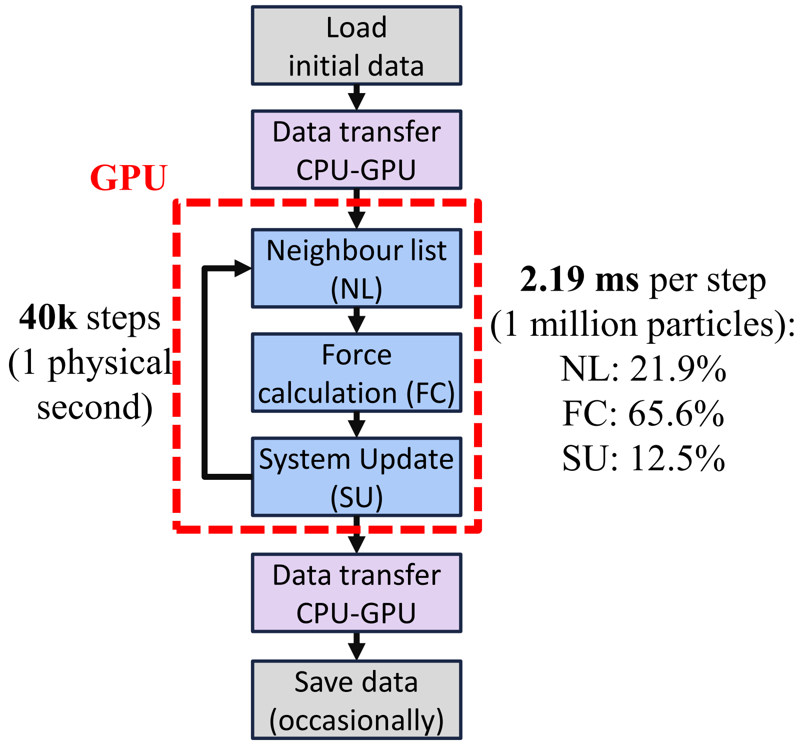
Pre-processing for Large-Scale Simulations

Post-processing for Large Output Data

100-200M particle simulations in a very user-friendly way

Implementation details

Single GPU execution:



Load initial data

Data transfer CPU-GPU

GPU

Neighbour list (NL)

Force calculation (FC)

System Update (SU)

Data transfer CPU-GPU

Save data (occasionally)

40k steps (1 physical second)

2.19 ms per step (1 million particles):

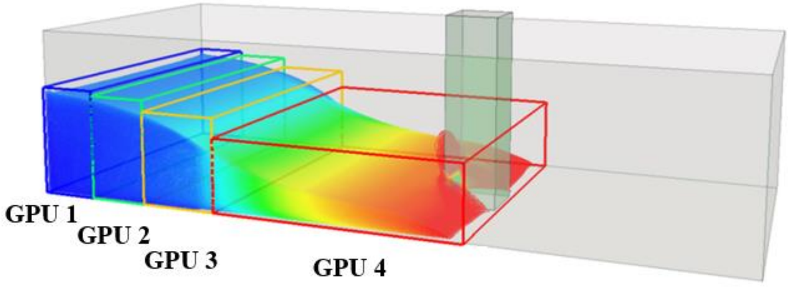
NL: 21.9%

FC: 65.6%

SU: 12.5%

Multi-GPU implementation:

- The **physical domain is divided** into different parts, and each part is computed on a GPU.



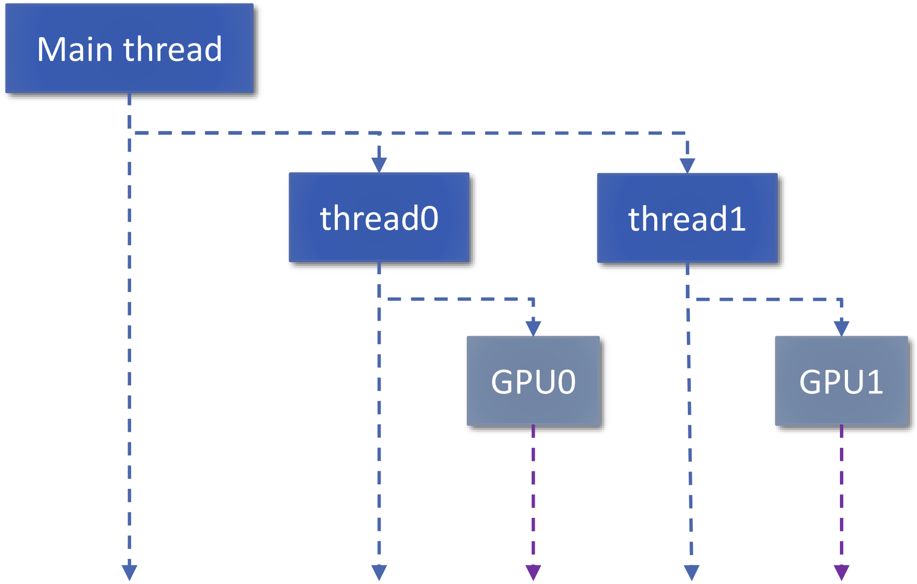
- A **dynamic load balancing** is applied to distribute the workload among the GPUs during simulation

New

- A **single execution process** that uses **multiple threads and multiple GPUs**.
- **Avoids MPI communication** between different processes, and **particle data transfers** are from **GPU to GPU**.
- A **single copy of the execution data** in the CPU that is shared between the threads. No execution data transfers is required.
- A **single process simplifies the implementation of complex functionalities** in DualSPHysics (wave generation, coupling with Chrono, coupling with MoorDynPlus, etc.).

Implementation details

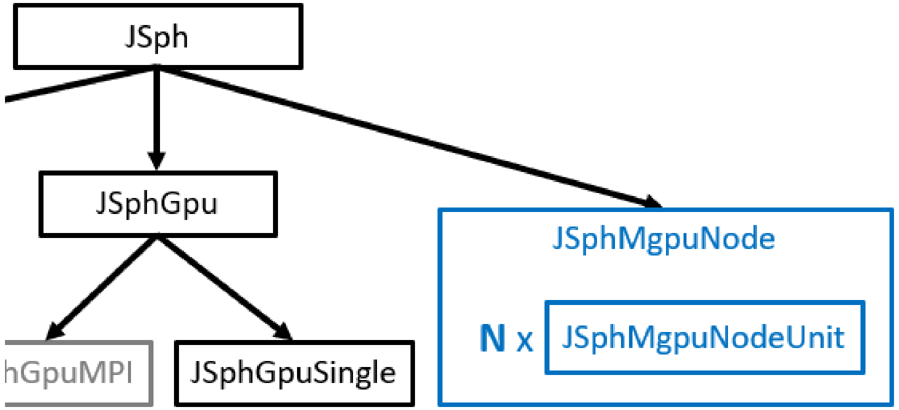
- **Standard C++ threads and CUDA streams**. Not MPI or OpenMP.
- Synchronisation between CPU threads via *std::mutex* objects and *std::atomic* variables.
- Synchronisation between GPUs and overlapping between calculation and transfers via *cudaStream* and *cudaEvent* objects (in multi-GPU almost all transfers are asynchronous).



- A main CPU thread manages the simulation and other threads
- One CPU thread is created per GPU
- Several synchronisation levels:
  - between all threads (n+1 threads)
  - between threads with GPU (n threads)
  - between two specific threads (2 threads)
- Explicit synchronisation between CPU threads and its GPU due:
  - Several GPU tasks are running at the same time on each GPU
  - All transfers are asynchronous
- All particle data are maintained in GPU memory. Data transfers are done directly between GPUs.

Implementation details

- Two new main classes: JSphMgpuNode and JSphMgpuNodeUnit.
- Most non-SPH features (wave generation, damping, couplings…) depend on JSph and are common to CPU, GPU and multi-GPU (*that is good!*).
- JSphGpuSingle and JSphMgpuNodeUnit use the same CUDA code for SPH (*that is good!*).
- NL code is different and more complicated for multi-GPU but does not usually require changes for new formulations (*that is ok!*).
- JSphMgpuNodeUnit includes arrays of particle data (like JSphGpu) and the SPH method code (like JSphGpuSingle).
- JSphMgpuNode (main thread) provides:
  - Synchronised access to non-SPH features in JSph from other threads.
  - Combines partial results (reduction and gather operations).



Multi-GPU overheads

- **Multi-GPU requires significant data transfers for each step:**
  - Particle interaction requires data from neighbouring GPU particles (including cell information).
  - The particles move and change from one GPU to another (all the data of these particles must be moved).
  - Load balancing redistributes particles and their data among multiple GPUs (only when necessary).
- **Multi-GPU requires new calculations not present in single GPU:**
  - Detection of particles changing GPUs.
  - Add arriving particles and remove departing ones.
  - Calculate cell information for neighbouring particles.
  - Evaluate each GPU's performance to improve load balancing.
  - Calculate new possible load balances to improve performance.
- **Multi-GPU requires synchronisation between the GPUs:**
  - Dt calculation starting from all particles.
  - Floating objects motion with particles on different GPUs.
  - Calculation of fluid elevation and other magnitudes for wave generation.
  - Coupling with other solvers (Chrono, MoorDynPlus, etc.).
  - Many other functionalities managed by the main thread.

Multi-GPU overheads

- **Multi-GPU requires significant data transfers for each step:**
  - Particle interaction requires data from neighbouring GPU particles (including cell information).
  - The particles move and change from one GPU to another (all the data of these particles must be moved).
  - Load balanci ecessary).
- **Multi-GPU requi**
  - Detection of
  - Add arriving
  - Calculate cel
  - Evaluate each
  - Calculate nev
- **Multi-GPU requi**
  - Dt calculation starting from all particles.
  - Floating objects motion with particles on different GPUs.
  - Calculation of fluid elevation and other magnitudes for wave generation.
  - Coupling with other solvers (Chrono, MoorDynPlus, etc.).
  - Many other functionalities managed by the main thread.

So, implementing SPH for multi-GPU may not be complicated, but achieving an **efficient multi-GPU implementation of DualSPHysics** with all its functionalities **is not easy**.

✓ This implementation minimises the number and size of data transfers between GPUs.

✓ Data transfers overlap with calculations using asynchronous transfers (although it is never perfect).

✓ This implementation minimises the synchronisation points.

Multi-GPU results

Hardware:

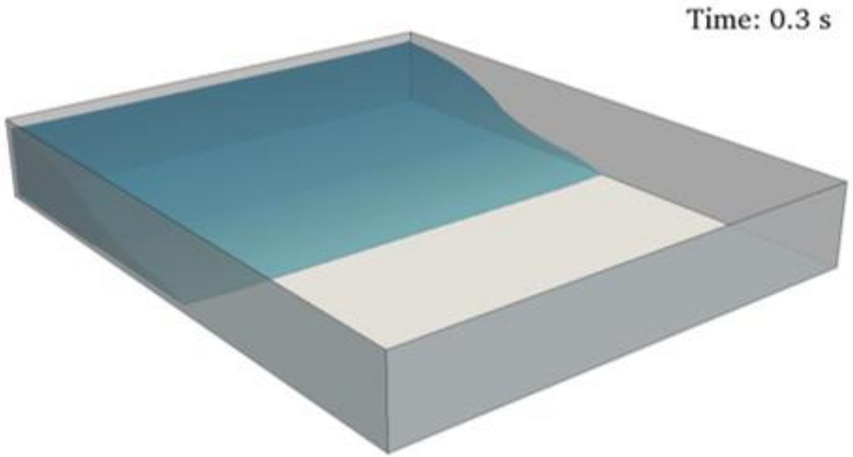
- **CPU:** 2x AMD EPYC 7282 at 2.8 GHz (16 cores)
- **GPU:** **8x NVIDIA L40S** (48GB):
  - **Architecture:** Ada Lovelace
  - **Memory:** 48 GB GDDR6 with ECC
  - **Memory Bandwidth:** 864 GB/s
  - **Interface:** PCIe 4.0 x16 (no NVLink)
  - **CUDA cores:** 18,176 (142 Multiprocessors)
  - **FP32 Performance:** 91.6 TFLOPS

Performance results:

- **Weak efficiency** for 2, 4 and 8 GPUs
- **Strong scalability** for 4 and 8 GPUs

Testcases:

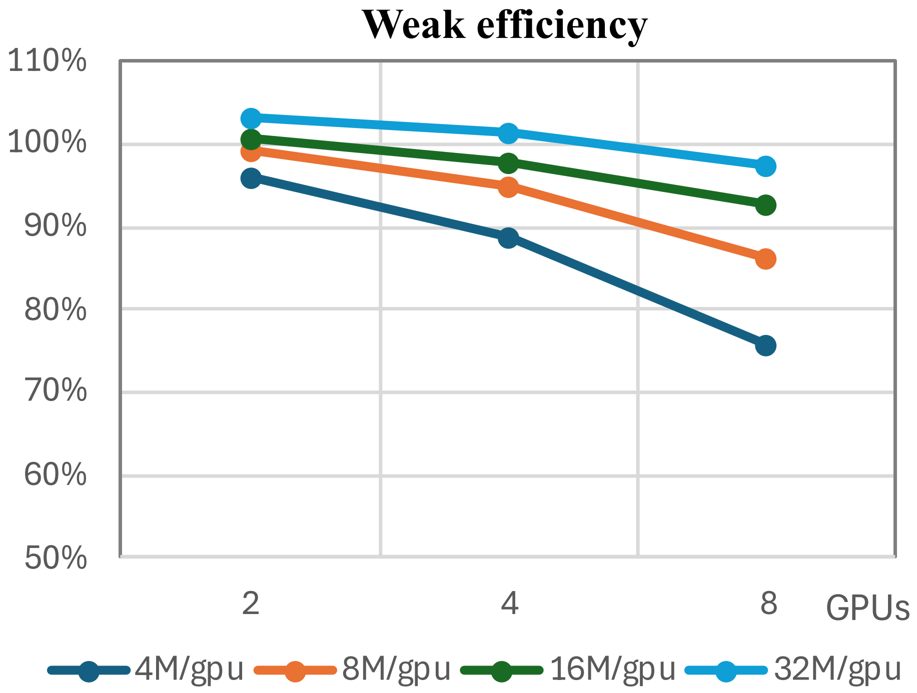
- **Dam break flow** (different versions):
  - Basic formulation
  - Advanced formulation
  - Advanced formulation + floating body
  - Advanced formulation + 8 floating bodies
  - Advanced formulation with Dynamic Load Balancing
- **Particles:** 4M to 256M



Similar testcase as used with multi-GPU MPI

Multi-GPU results

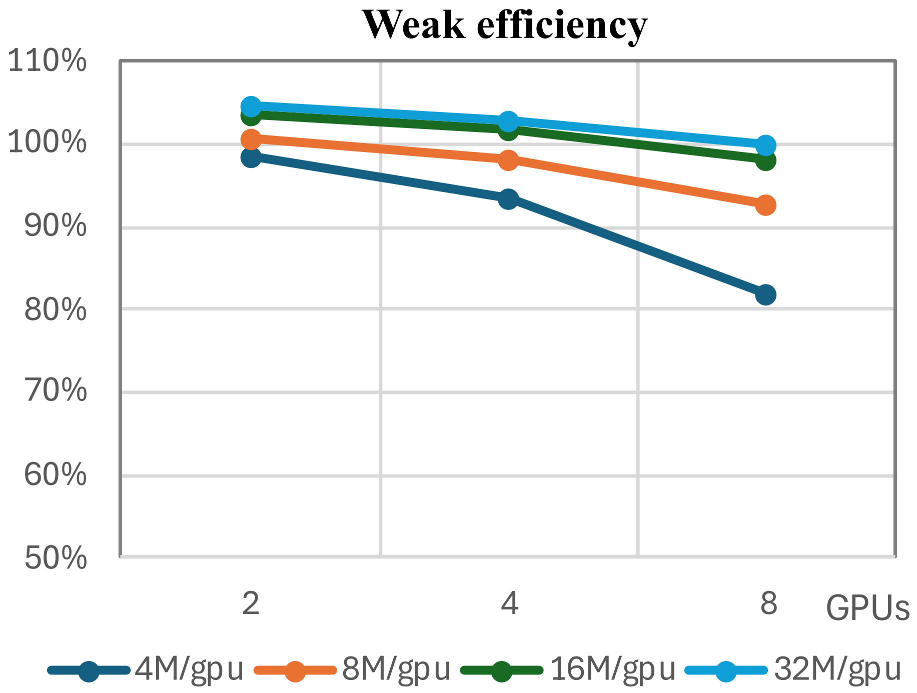
Testcase 1: Basic formulation	
Algorithm:	Verlet
Viscosity:	Artificial
DDT:	none
Boundaries:	DBC
Floating bodies:	none
Single-GPU runtimes (16M)	
Runtime step:	2.19 ms / (steps*M)
Runtime NL/FC/SU:	21.9% / 65.6% / 12.5%
Runtime FC-Fluid:	61.4%
Runtime FC-Bound:	1.2%
Runtime mDBC:	none
Runtime Floating:	none



GPUs	4M/gpu	8M/gpu	16M/gpu	32M/gpu
1	100.0%	100.0%	100.0%	100.0%
2	96.0%	99.1%	100.6%	103.3%
4	88.9%	94.8%	97.9%	101.3%
8	75.8%	86.3%	92.7%	97.4%

Multi-GPU results

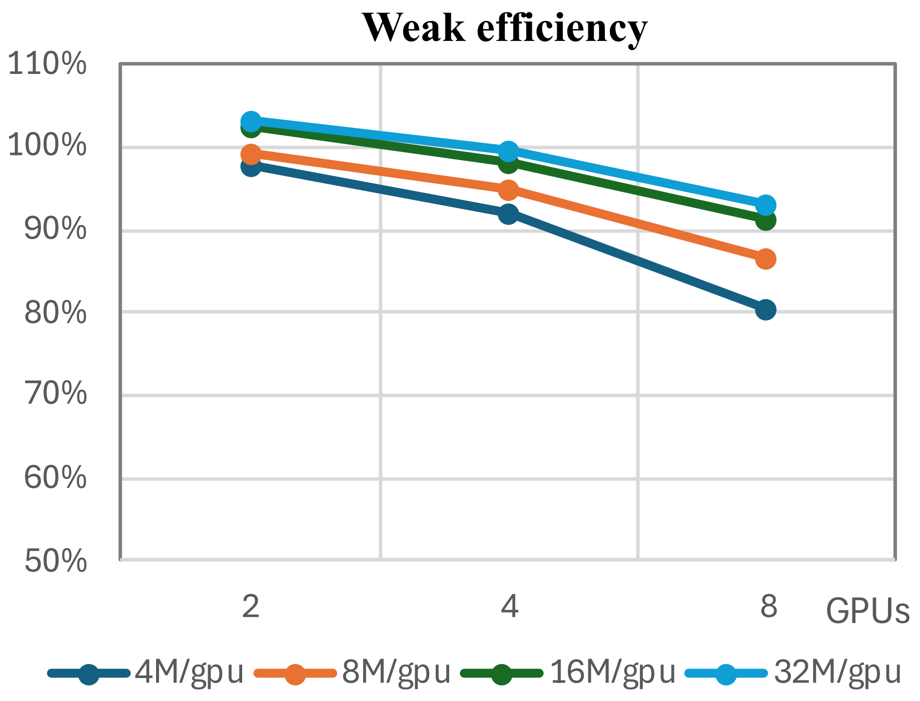
Testcase 2: Advanced formulation	
Algorithm:	Symplectic
Viscosity:	Laminar + SPS
DDT:	Fourtakas Full
Boundaries:	mDBC no-slip + no penetration
Floating bodies:	none
Single-GPU runtimes (16M)	
Runtime step:	6.47 ms / (steps*M)
Runtime NL/FC/SU:	21.1% / 68.5% / 10.4%
Runtime FC-Fluid:	59.7%
Runtime FC-Bound:	1.2%
Runtime mDBC:	4.3%
Runtime Floating:	none



GPUs	4M/gpu	8M/gpu	16M/gpu	32M/gpu
1	100.0%	100.0%	100.0%	100.0%
2	98.6%	100.8%	103.5%	104.6%
4	93.5%	98.3%	101.9%	102.9%
8	82.0%	92.6%	98.1%	100.1%

Multi-GPU results

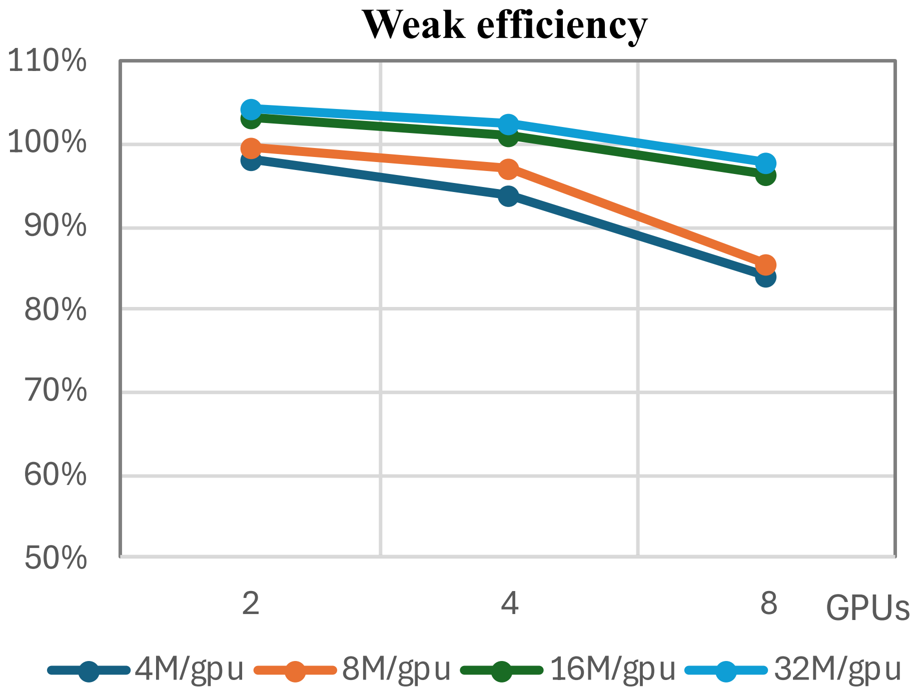
Testcase 3: Advanced + 1 floating body	
Algorithm:	Symplectic
Viscosity:	Laminar + SPS
DDT:	Fourtakas Full
Boundaries:	mDBC no-slip + no penetration
Floating bodies:	1x large floating body
Single-GPU runtimes (16M)	
Runtime step:	7.83 ms / (steps*M)
Runtime NL/FC/SU:	17.6% / 69.8% / 12.6%
Runtime FC-Fluid:	58.6%
Runtime FC-Bound:	1.0%
Runtime mDBC:	7.0%
Runtime Floating:	6.8%



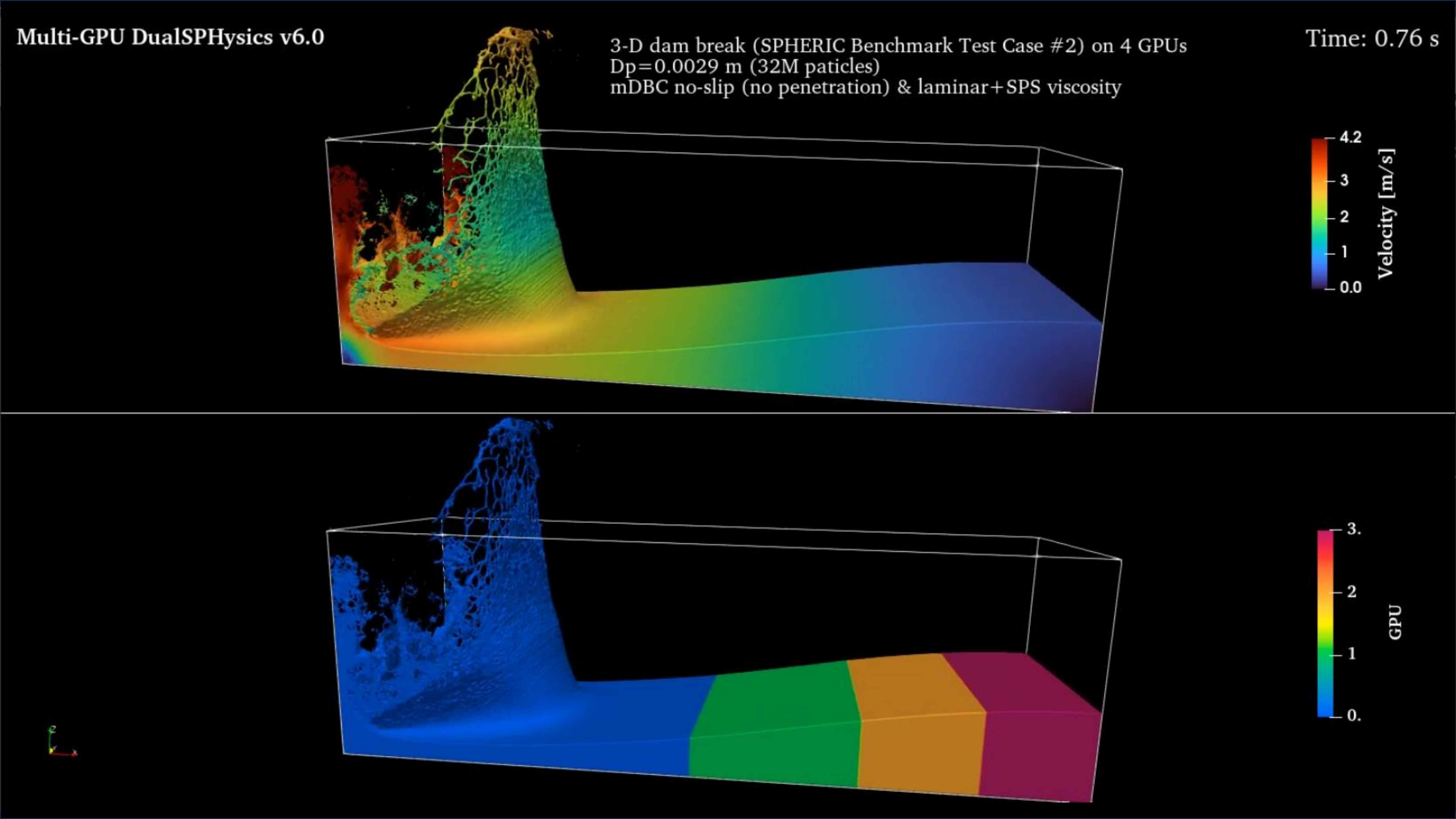
GPUs	4M/gpu	8M/gpu	16M/gpu	32M/gpu
1	100.0%	100.0%	100.0%	100.0%
2	97.8%	99.3%	102.4%	103.2%
4	91.9%	94.8%	98.3%	99.7%
8	80.4%	86.5%	91.2%	93.0%

Multi-GPU results

Testcase 4: Advanced + 8 floating bodies	
Algorithm:	Symplectic
Viscosity:	Laminar + SPS
DDT:	Fourtakas Full
Boundaries:	mDBC no-slip + no penetration
Floating bodies:	8x floating bodies
Single-GPU runtimes (16M)	
Runtime step:	7.65 ms / (steps*M)
Runtime NL/FC/SU:	18.0% / 72.4% / 9.6%
Runtime FC-Fluid:	60.8%
Runtime FC-Bound:	1.1%
Runtime mDBC:	7.3%
Runtime Floating:	3.7%



GPUs	4M/gpu	8M/gpu	16M/gpu	32M/gpu
1	100.0%	100.0%	100.0%	100.0%
2	98.2%	99.7%	103.4%	104.3%
4	93.9%	97.2%	101.0%	102.4%
8	84.2%	85.4%	96.3%	97.9%



Multi-GPU DualSPHysics v6.0

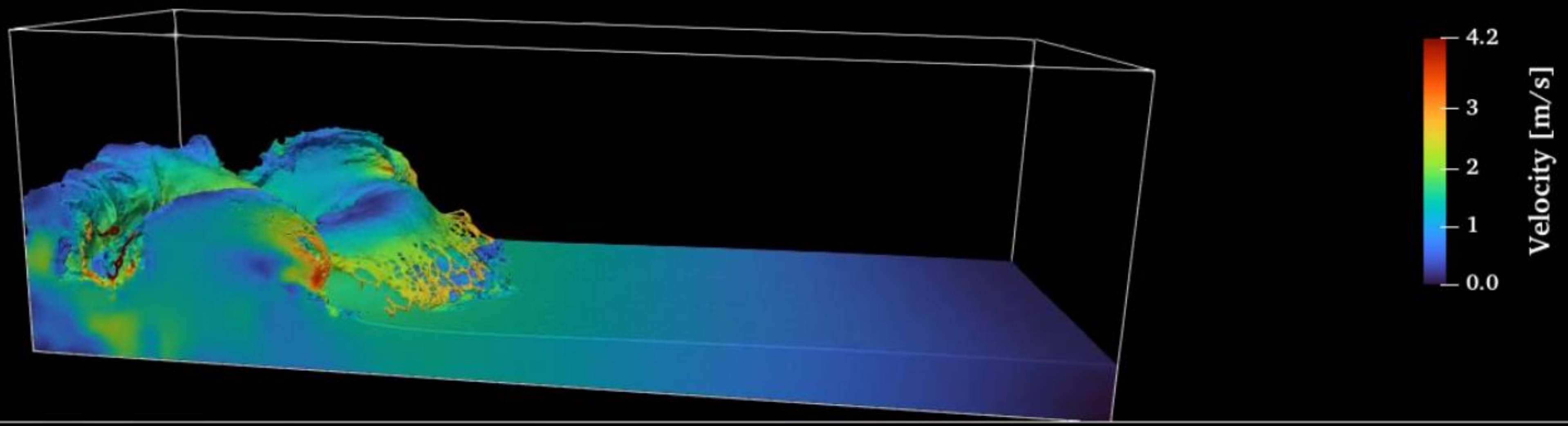
3-D dam break (SPHERIC Benchmark Test Case #2) on 4 GPUs
Dp=0.0029 m (32M paticles)
mDBC no-slip (no penetration) & laminar+SPS viscosity

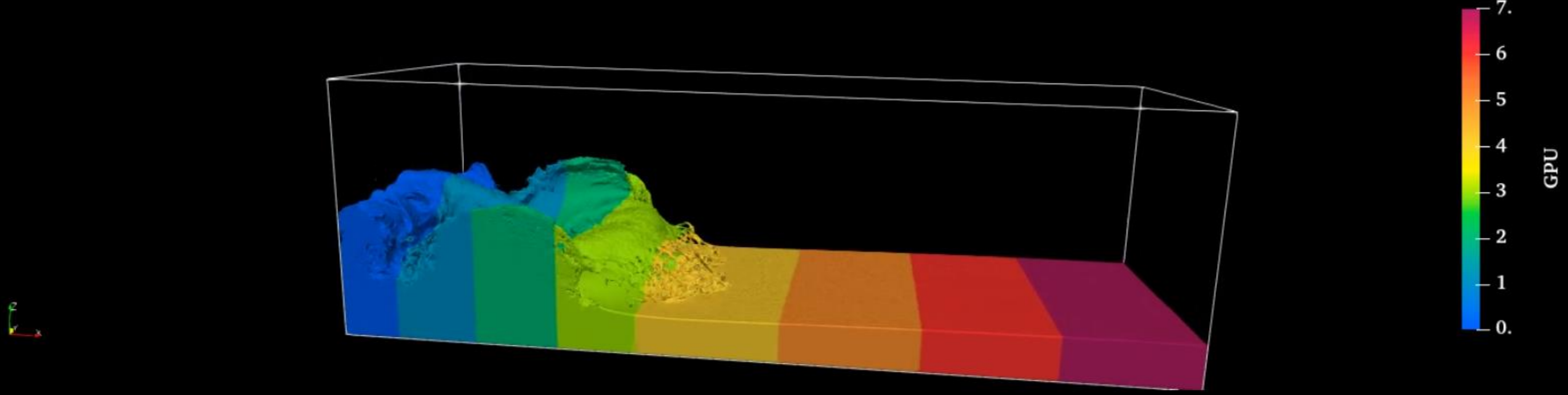
Time: 0.76 s

Multi-GPU DualSPHysics v6.0

Time: 1.53 s

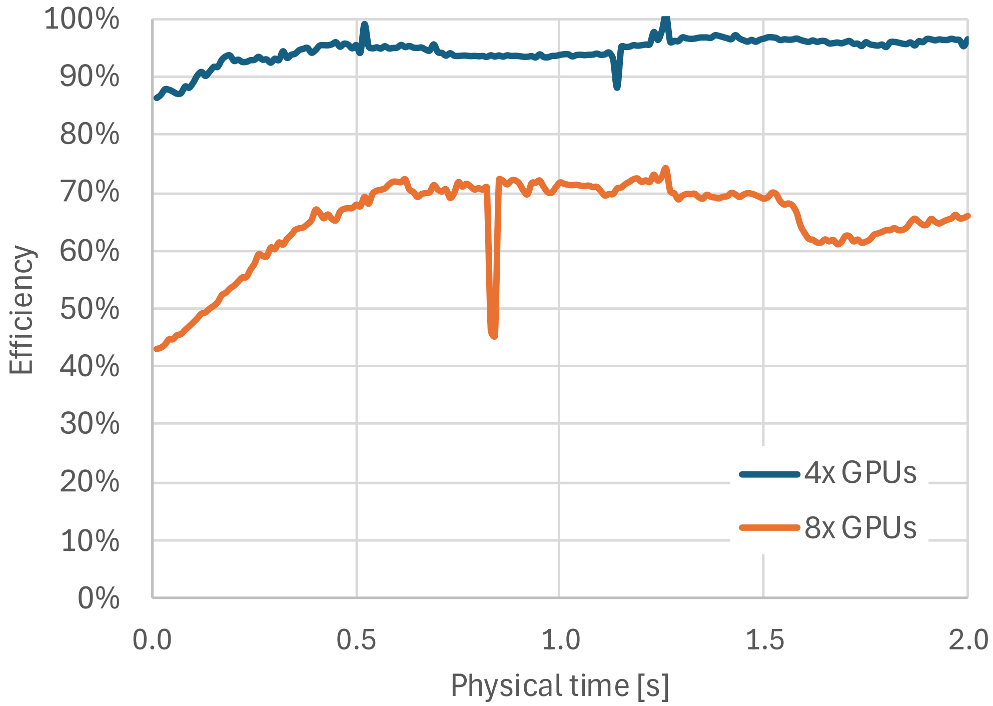
3-D dam break (SPHERIC Benchmark Test Case #2) on 8 GPUs
Dp=0.0029 m (32M paticles)
mDBC no-slip (no penetration) & laminar+SPS viscosity

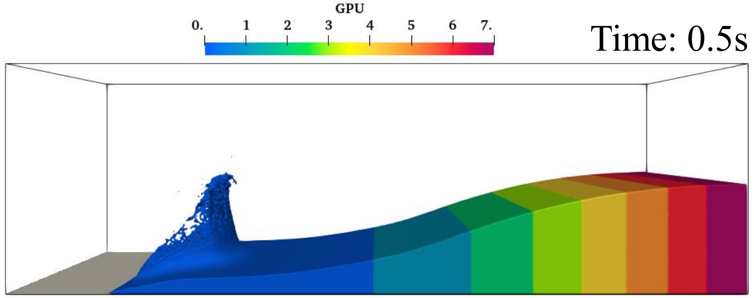




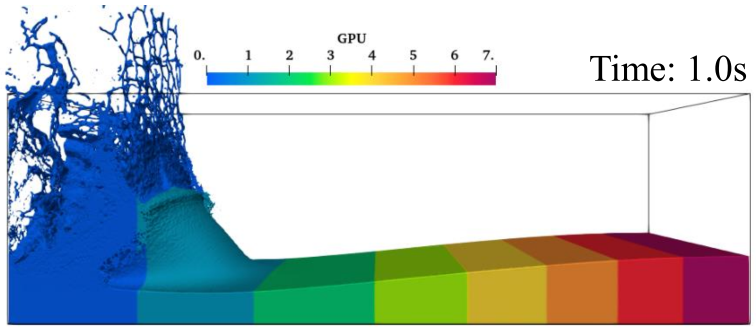
Multi-GPU results

SPHERIC Benchmark Test Case #2	
Algorithm:	Symplectic
Viscosity:	Laminar + SPS
DDT:	Fourtakas Full
Boundaries:	mDBC no-slip + no penetration
Runtimes (32M)	
Single-GPU	13.3 h
4 GPUs	3.5 h (3.8x faster)
8 GPUs	2.5 h (5.3x faster)

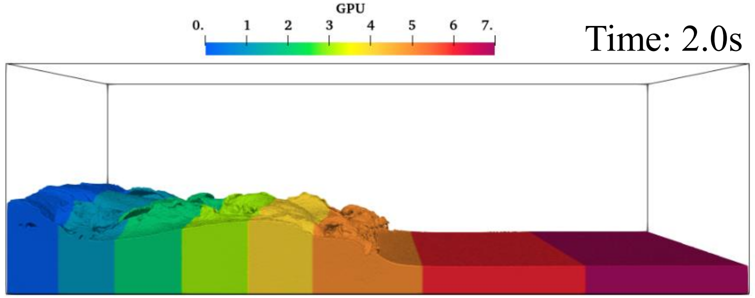




Time: 0.5s



Time: 1.0s



Time: 2.0s

Multi-GPU results

Largest full SPH free-surface fluid simulation in 2013. More than 1 billion particles!!

GPUs: 64x M2090 (BSC)
MPI: Dynamic balancing
Algorithm: Verlet & Wendland
Particles: 1,015 Millions
Steps: 237,065
Runtime: 79.1 hours
Physical time: 12 seconds

cpu
gpu
DualSPHysics
NVIDIA
CUDA RESEARCH CENTER
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

- Large wave interaction with oil rig using **10^9 particles**.
- More than 237,000 simulation steps to simulate **12 physical seconds**.
- **79.1 hours** using **64 GPUs** Tesla M2090.
- **Huge complexity** for pre-processing, simulation and post-processing.

challenge but not very

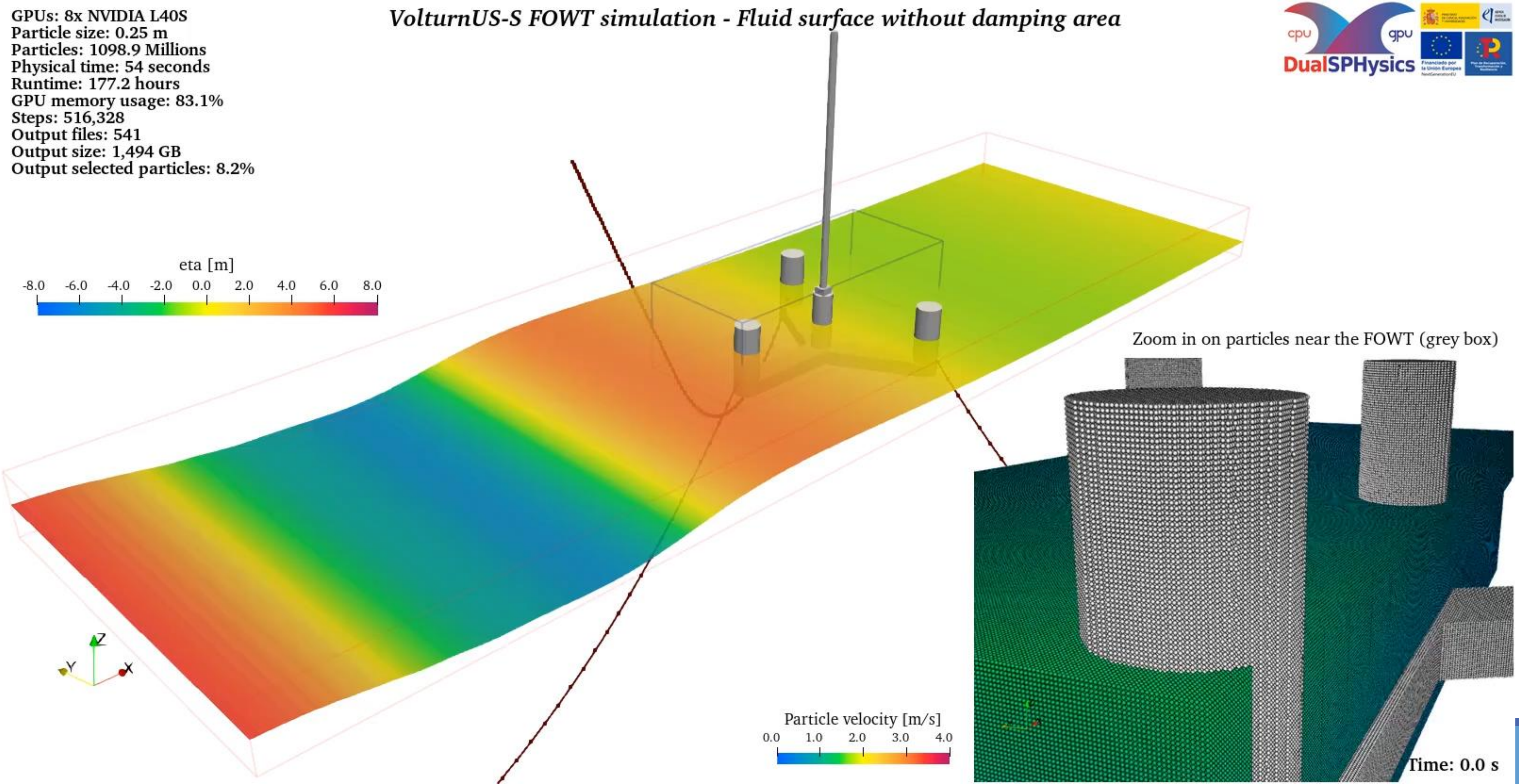
rcomputer is required.

- Too much effort for practical use.
- Many particles do **not allow modelling of complex problems** involving different physical phenomena.

This simulation with 10^9 particles is possible with…

- 8x L40S (45 GB) in less than 27.4 hours (2.9 times faster)

Multi-GPU results



GPUs: 8x NVIDIA L40S
Particle size: 0.25 m
Particles: 1098.9 Millions
Physical time: 54 seconds
Runtime: 177.2 hours
GPU memory usage: 83.1%
Steps: 516,328
Output files: 541
Output size: 1,494 GB
Output selected particles: 8.2%

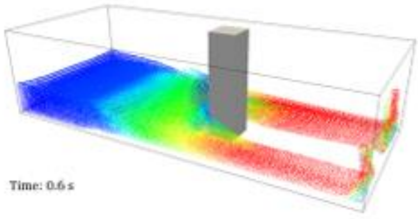
VolturnUS-S FOWT simulation - Fluid surface without damping area

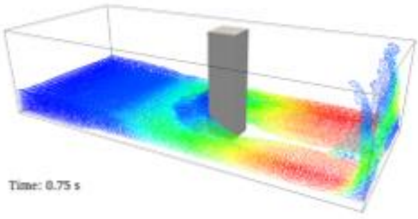
Zoom in on particles near the FOWT (grey box)

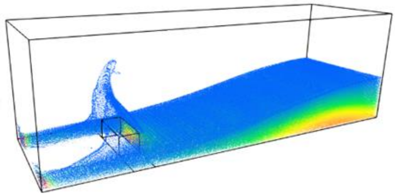
Multi-GPU in package v6.0 beta

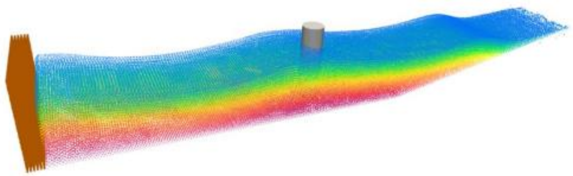
Examples using multi-GPU:

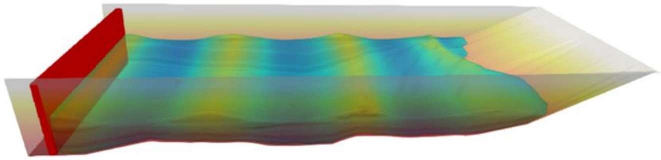
- main/01_DamBreak
- main/18_Bathymetry
- mdbc/04_DamBreak
- mdbc/07_WavesCylinder



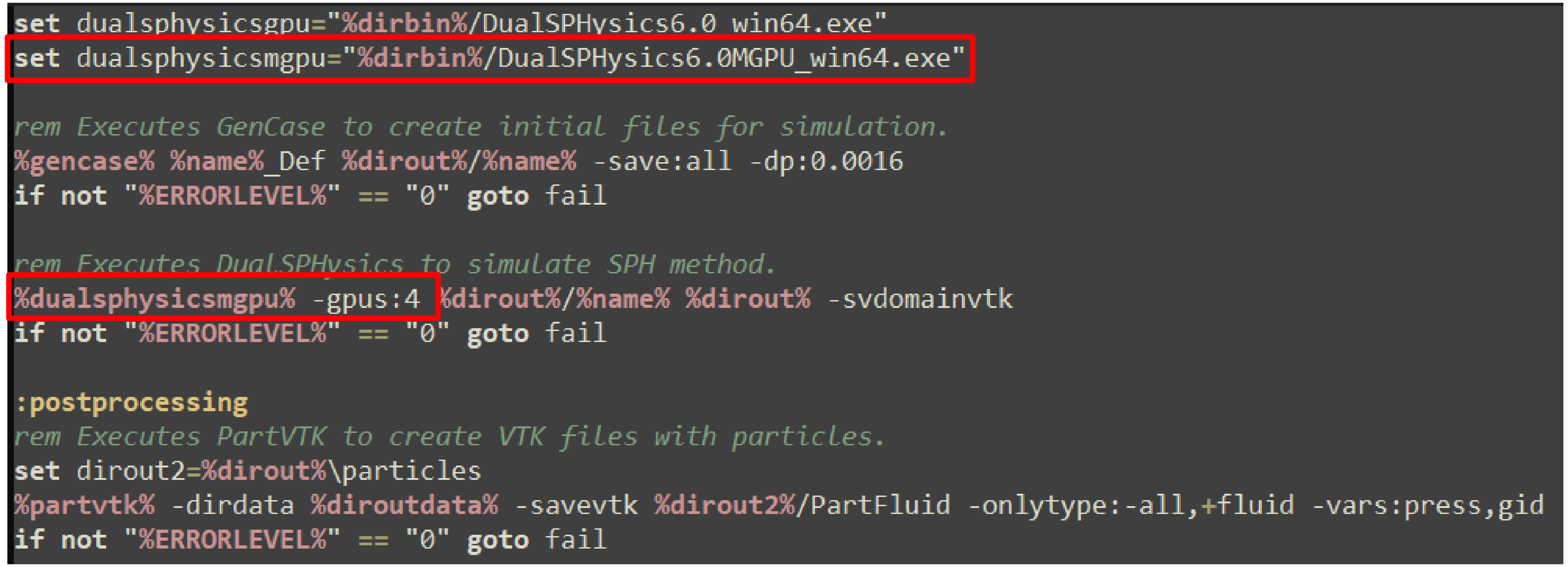








main/01_DamBreak/wCaseDambreak_win64_4GPU.bat



```
set dualsphysicsgpu="%dirbin%/DualSPHysics6.0_win64.exe"
set dualsphysicsmgpu="%dirbin%/DualSPHysics6.0MGPU_win64.exe"

rem Executes GenCase to create initial files for simulation.
%gencase% %name%_Def %dirout%/%name% -save:all -dp:0.0016
if not "%ERRORLEVEL%" == "0" goto fail

rem Executes DualSPHysics to simulate SPH method.
%dualsphysicsmgpu% -gpus:4 %dirout%/%name% %dirout% -svdomainvtk
if not "%ERRORLEVEL%" == "0" goto fail

:postprocessing
rem Executes PartVTK to create VTK files with particles.
set dirout2=%dirout%\particles
%partvtk% -dirdata %diroutdata% -savevtk %dirout2%/PartFluid -onlytype:-all,+fluid -vars:press,gid
if not "%ERRORLEVEL%" == "0" goto fail
```

Conclusions

- The multi-GPU version **is not finished yet** (periodic boundaries and inlet/outlet are missing), but...
- Right now, we can already simulate **real complex cases with more than 1 billion particles.**
- Improvements in pre-processing and post-processing allow us to address **large multi-GPU simulations without extra difficulty for the user.**
- Good efficiency simulating **simple and complex cases.**
- **Efficiency close to 100% simulating 8-16M/GPU on 8 GPUs.**

For developers…

- Multi-GPU code is **more complicated** than single-GPU code.
- However, **most of the CUDA code is the same** for single- and multi-GPU code.
- Some important **changes in the particle data arrays** and elsewhere **make it easier** to implement new SPH formulations and new features.