13th International Workshop on OpenCL and SYCL

IWOCL 2025

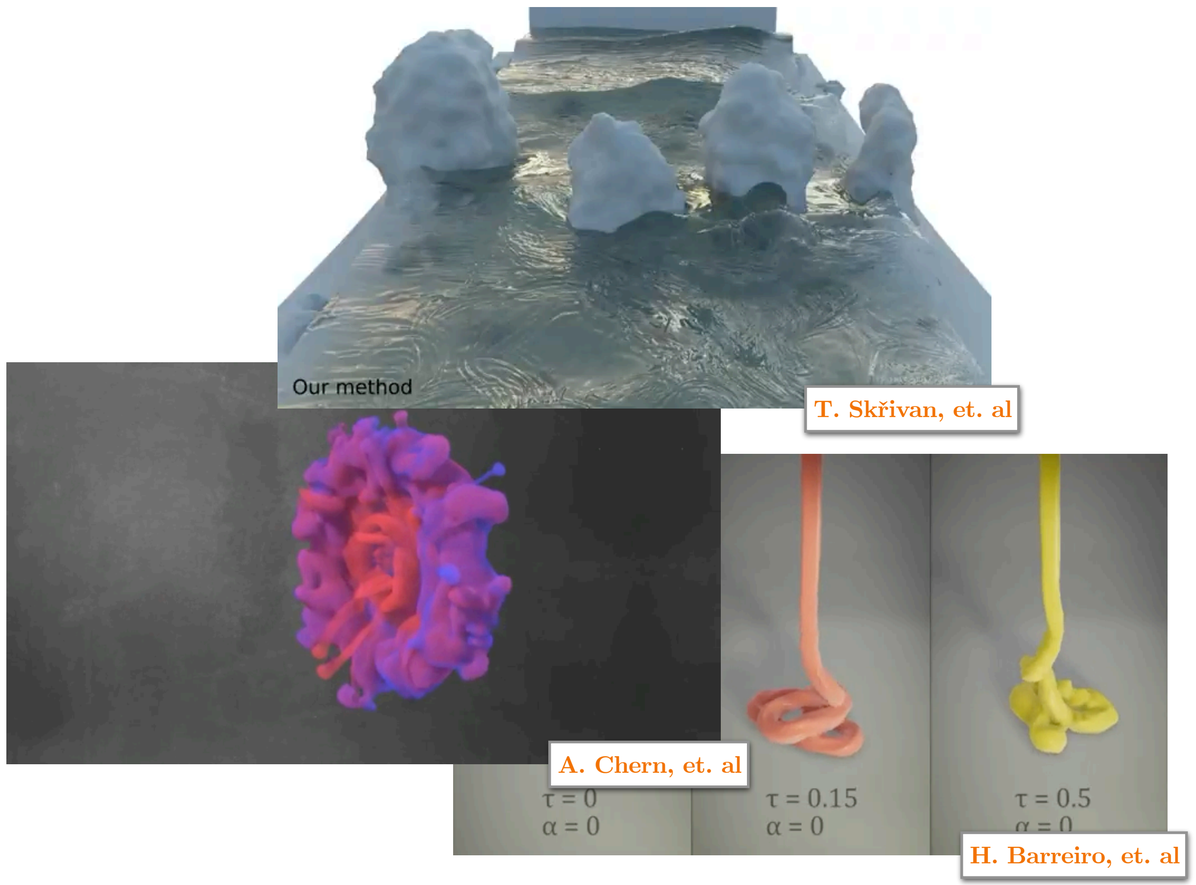
Shamrock: Exascale Hydrodynamics for Astrophysics Using SYCL

Timothée David Cléris, University of Grenoble.

April 7-11, 2025 | Heidelberg, Germany | iwocl.org

Computational Fluid Dynamics

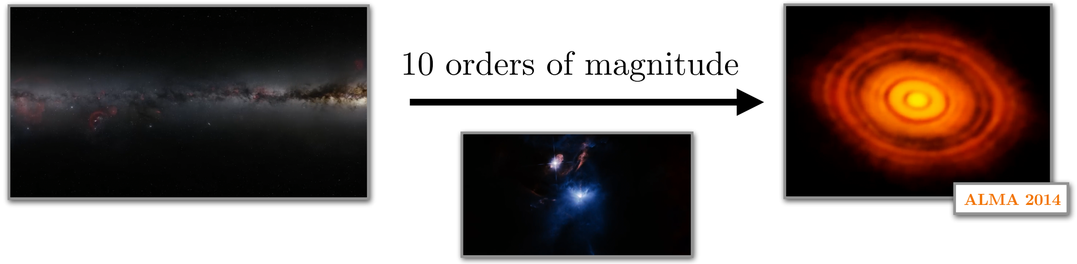
"Normal" world



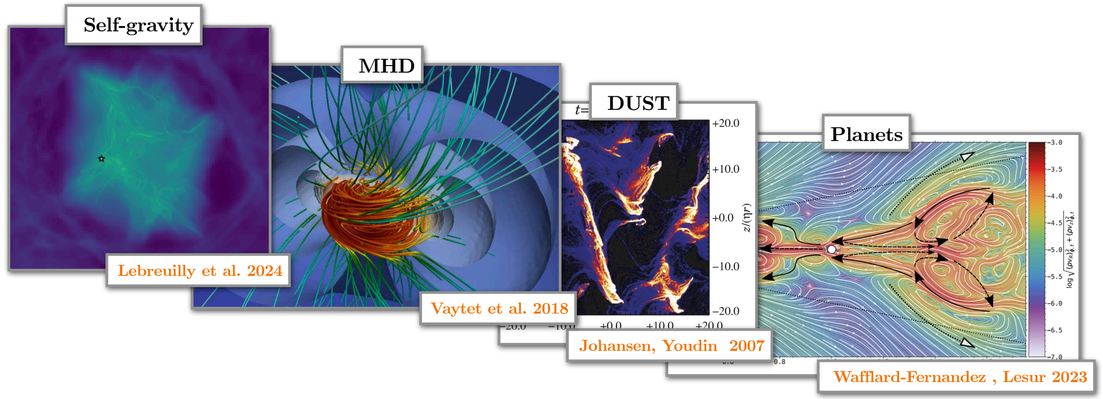
- Small scale range
- Small density range

"Astrophysical" world

Multi-scale



Multi-physics



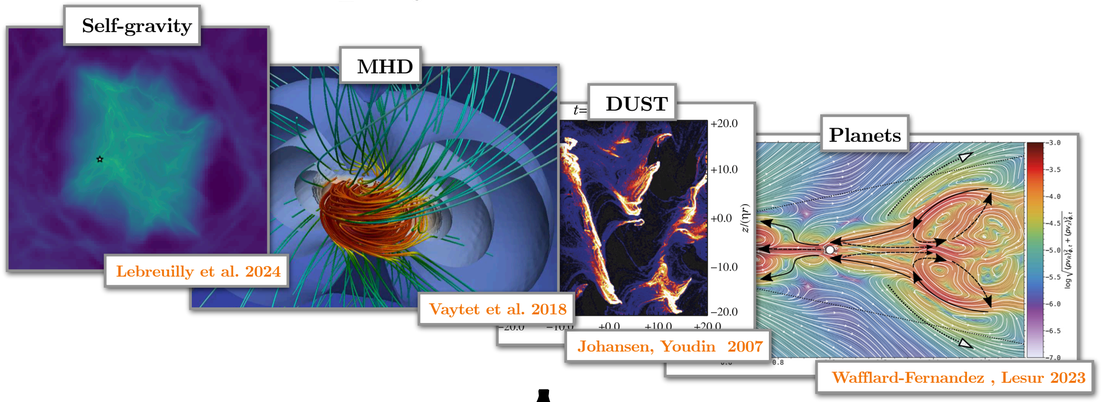
- Large scale range
- Large density range

$\Rightarrow$ fp64

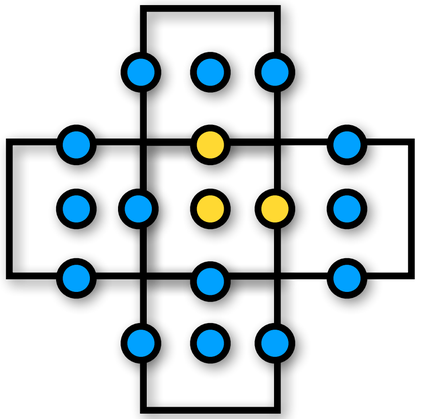
- Many physical effects $\Rightarrow$ DAG scheduler

Computational Fluid Dynamics

"Astrophysical" simulations

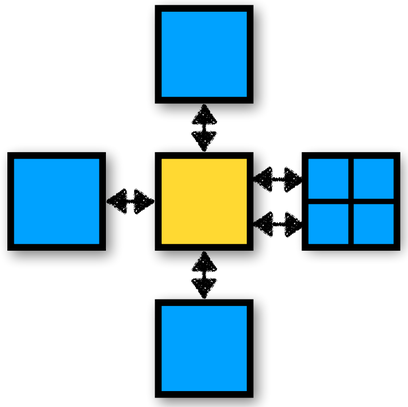


Finite elements



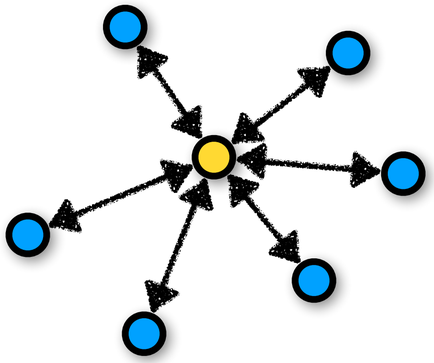
(Fastest)

Finite Volumes



(Most versatile)

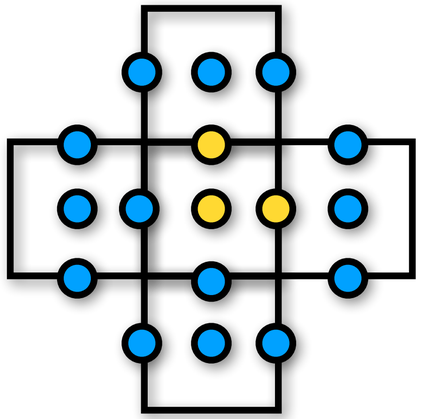
SPH



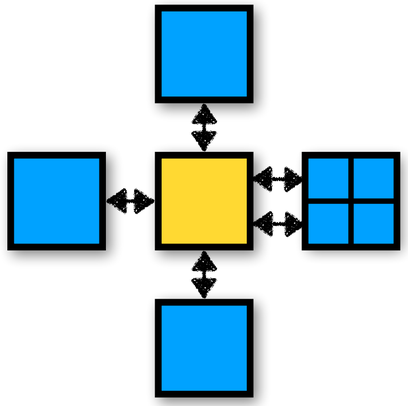
(Weird geometries)

Computational Fluid Dynamics

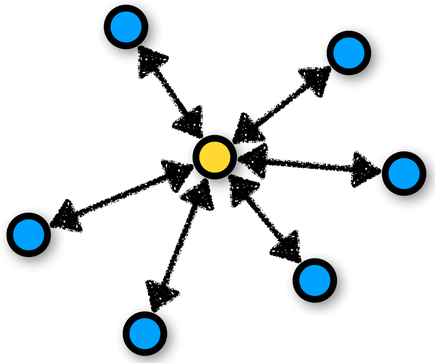
Finite elements

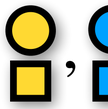
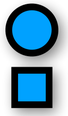


Finite Volumes



SPH

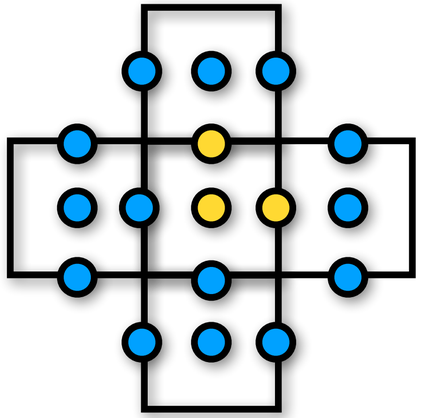


$\gamma($  $)$ & $\mathbb{U}^{t+1}$  $= F\left(\{\mathbb{U}_i^t\}_{i\in\{}$  $\}\right)$

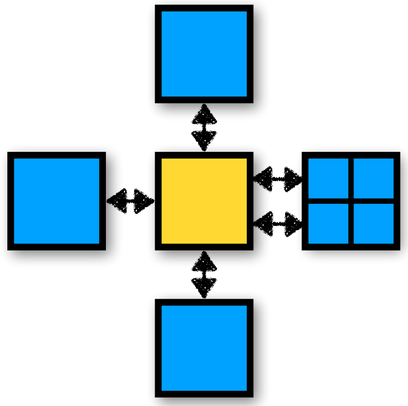
Interaction criterion — Numerical scheme

All solvers are the "same"

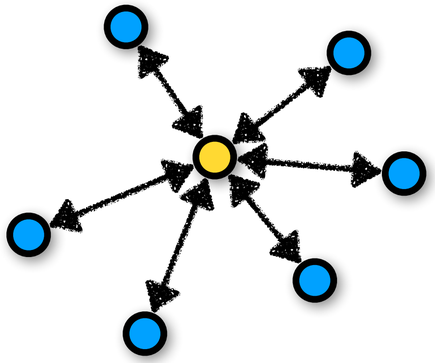
Finite elements



Finite Volumes



SPH



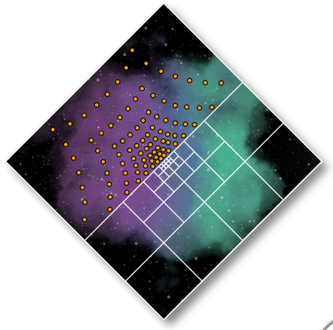
$\gamma($  $)$ & $\mathbb{U}^{t+1}$ $= F\left(\{\mathbb{U}_i^t\}_{i\in\{}$ $\}\right)$

Interaction criterion — Numerical scheme

Criterion γ
Scheme F

Abstraction

The goal in Shamrock



Shamrock

Criterion γ
Scheme F

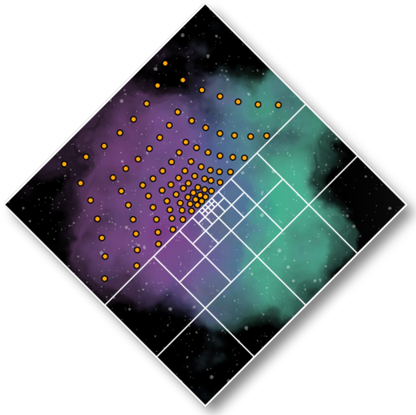
Generic modules

Numerical method
+ Exascale scalability

Optimising SPH ⇔ Optimising AMR

Generic modules are coded **once** for all schemes

Reality is a bit more nuanced but that's the goal

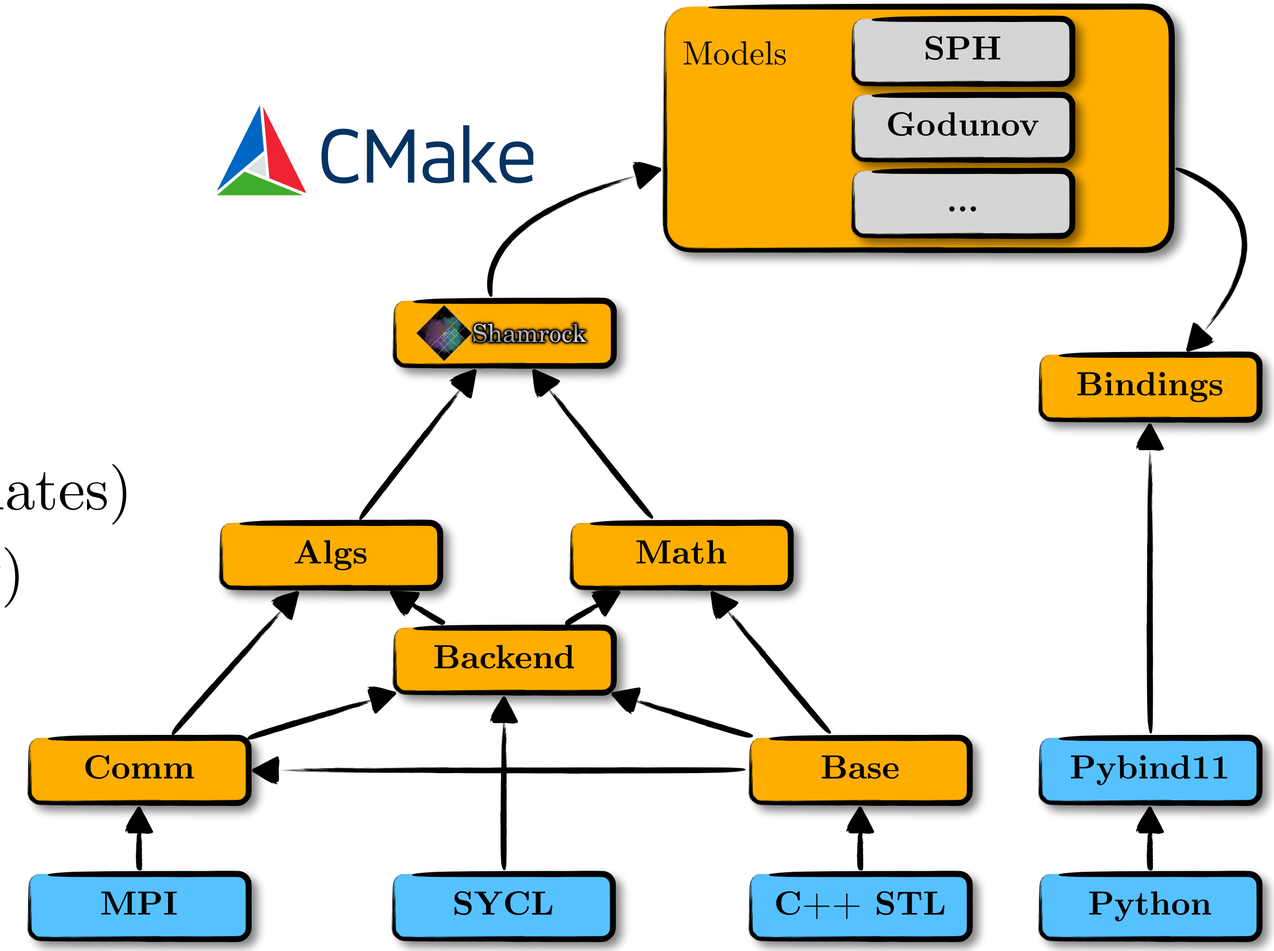


Shamrock

- `C++17` (templated)
- (Multi-GPU) `SYCL + MPI`
- Inter-operable with Python (runscripts)
- No #ifdefs !!! (Runtime switch using templates)
- Multiple Hydro solver (SPH, FE, Godunov)
- Open source CECILL 2.1 License

Publications :

- David--Cléris et al. 2025 (MNRAS), accepted
- David--Cléris 2025 (IOWCL proceeding)
- PHD (defended in 2025)

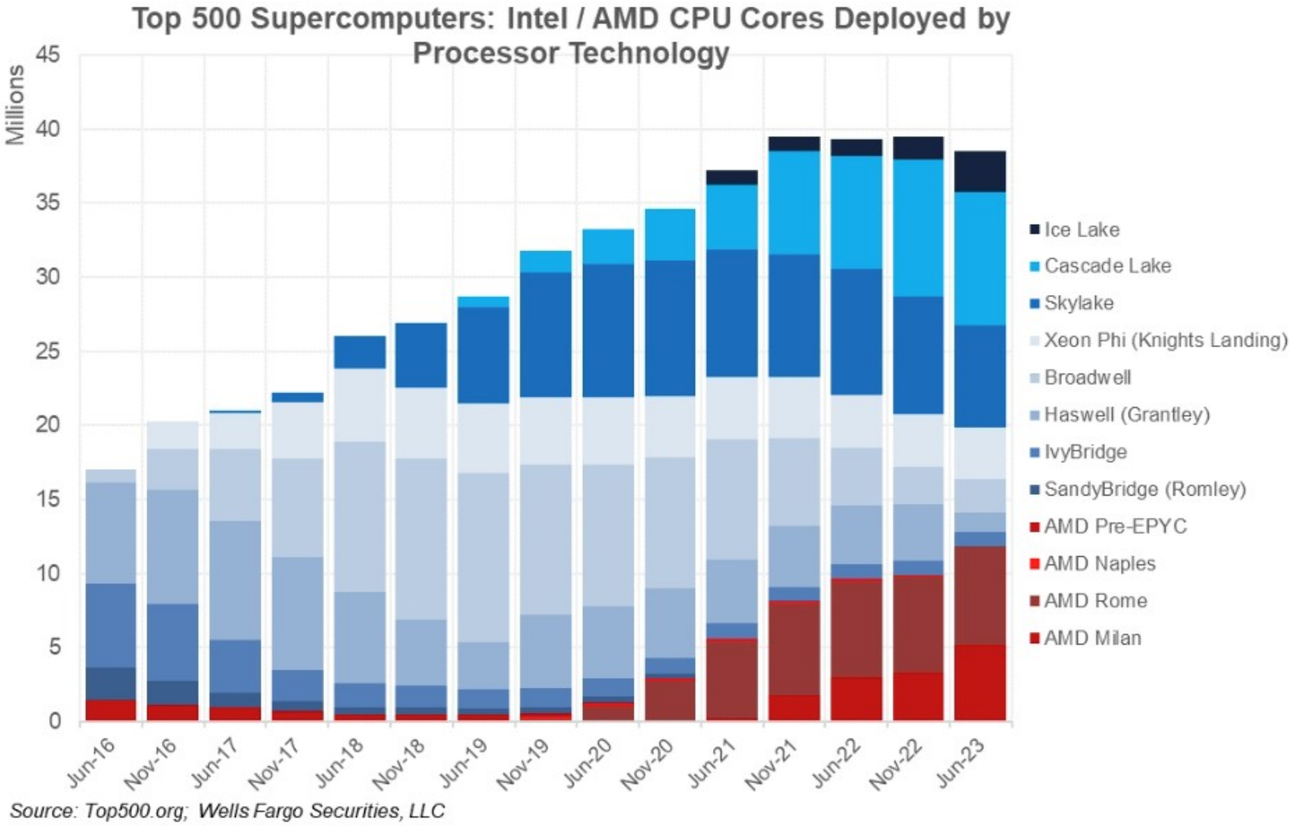


Exascale supercomputers

Supercomputers

Top 500 list :

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
5	**HPC6** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461



⇒ Supercomputers move to GPUs

(Efficiency, density, AI, ...)

(But all vendors involved)

Why SYCL ?

Possibilities :

But :

- Lack of support or performance
- Proprietary
- Hard to maintain

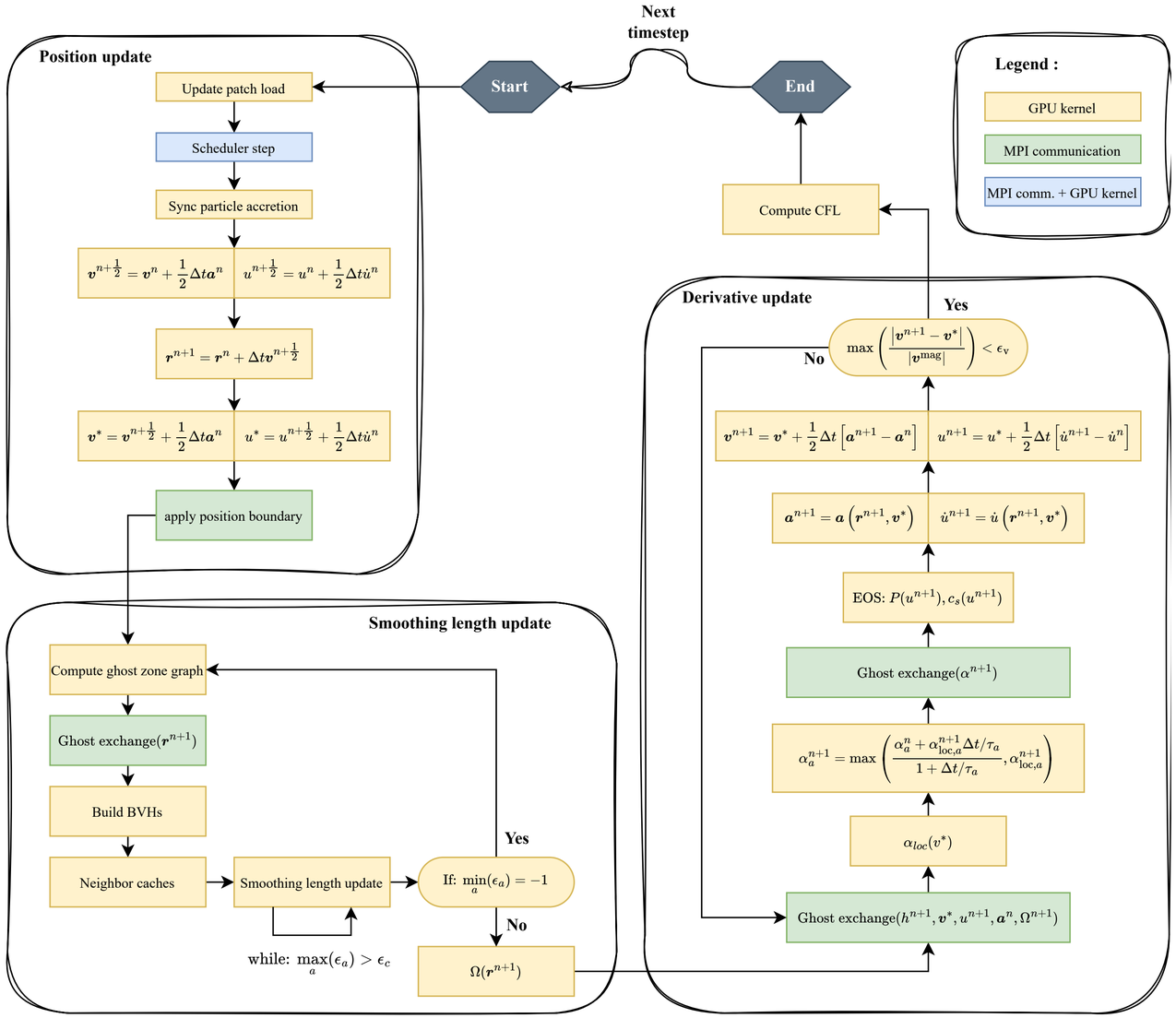
Real contenders :

- ✅ C++17 based
- ✅ Open standard
- ✅ Multiple implementations
- ✅ Portability (vendor & performance)
- ❌ Low adoption in astrophysics
- ✅ DAG based scheduling

- ✅ C++17 library / framework
- ❌ Single implementation
- ✅ Portability (vendor & performance)
- ✅ More adoption in astrophysics
- ❌ Aimed mostly at single stream codes

Why we use the DAG-scheduler

Control flow for a single config



- Shamrock relies heavily on dynamic runtime allocations
- Too many control flow possible

  $\Rightarrow$ To much maintenance to schedule manually

  $\Rightarrow$ Out-of order queues

- We chose to use buffers <u>initially</u>

Issues with buffers

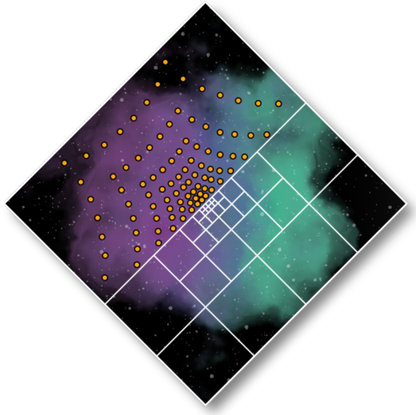
But :

- No standard interop with USM for external libs (MPI, xxBlas, xxFFT, ...)
- Lack of fine memory control (e.g. to do memory pooling)
- Additional latency (Crisci, et al 2024, SYCL-Bench 2020)
- USM allow additional optimizations (e.g. restrict, Coarse-grained events & instant submission)

Migrating :

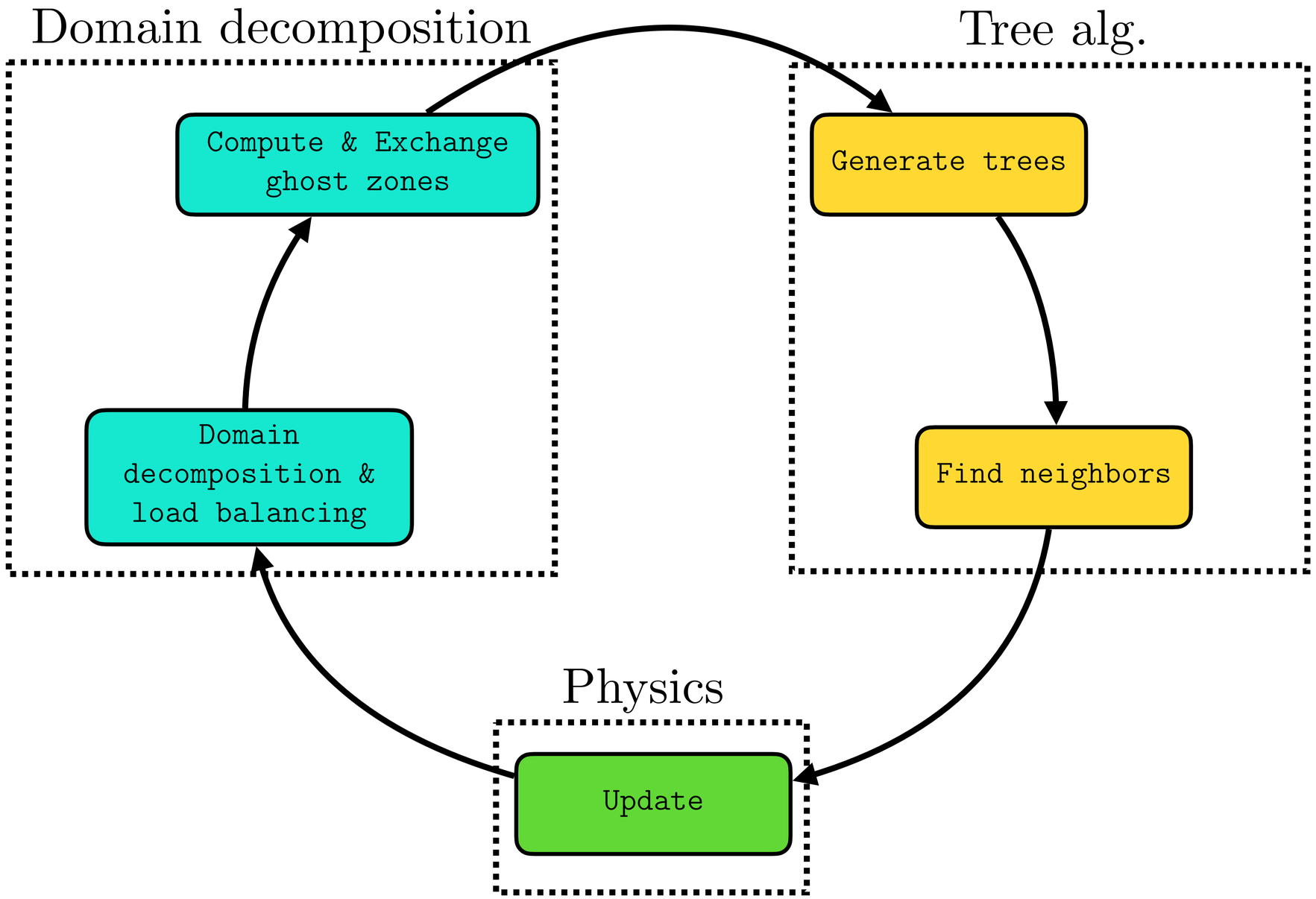
- Need to keep out-of-order queues
- Need to minimize refactoring (use `sycl::buffer` like wrappers)

⚠ **No performance comparison yet (on-going)**



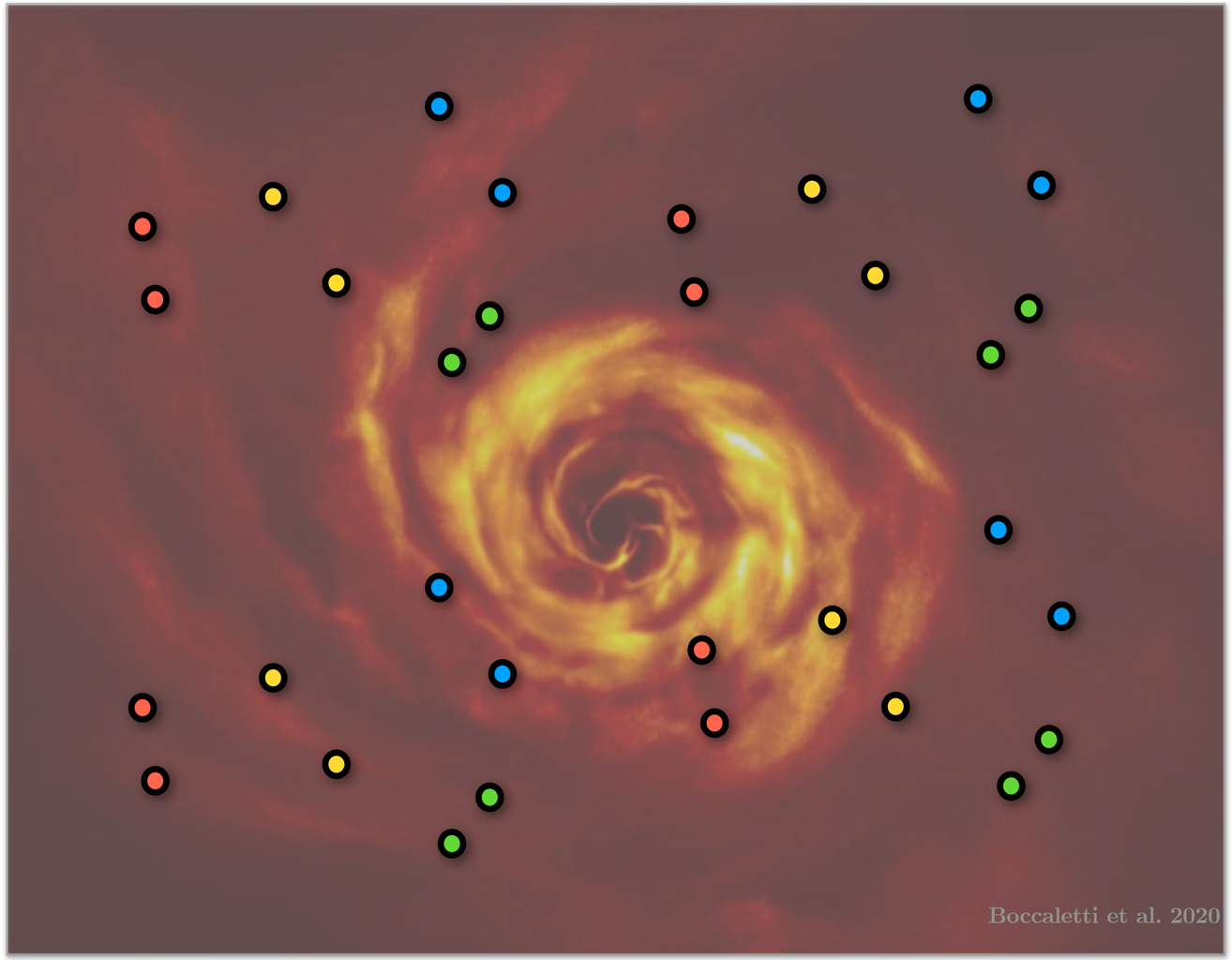
Shamrock

The main steps in Shamrock



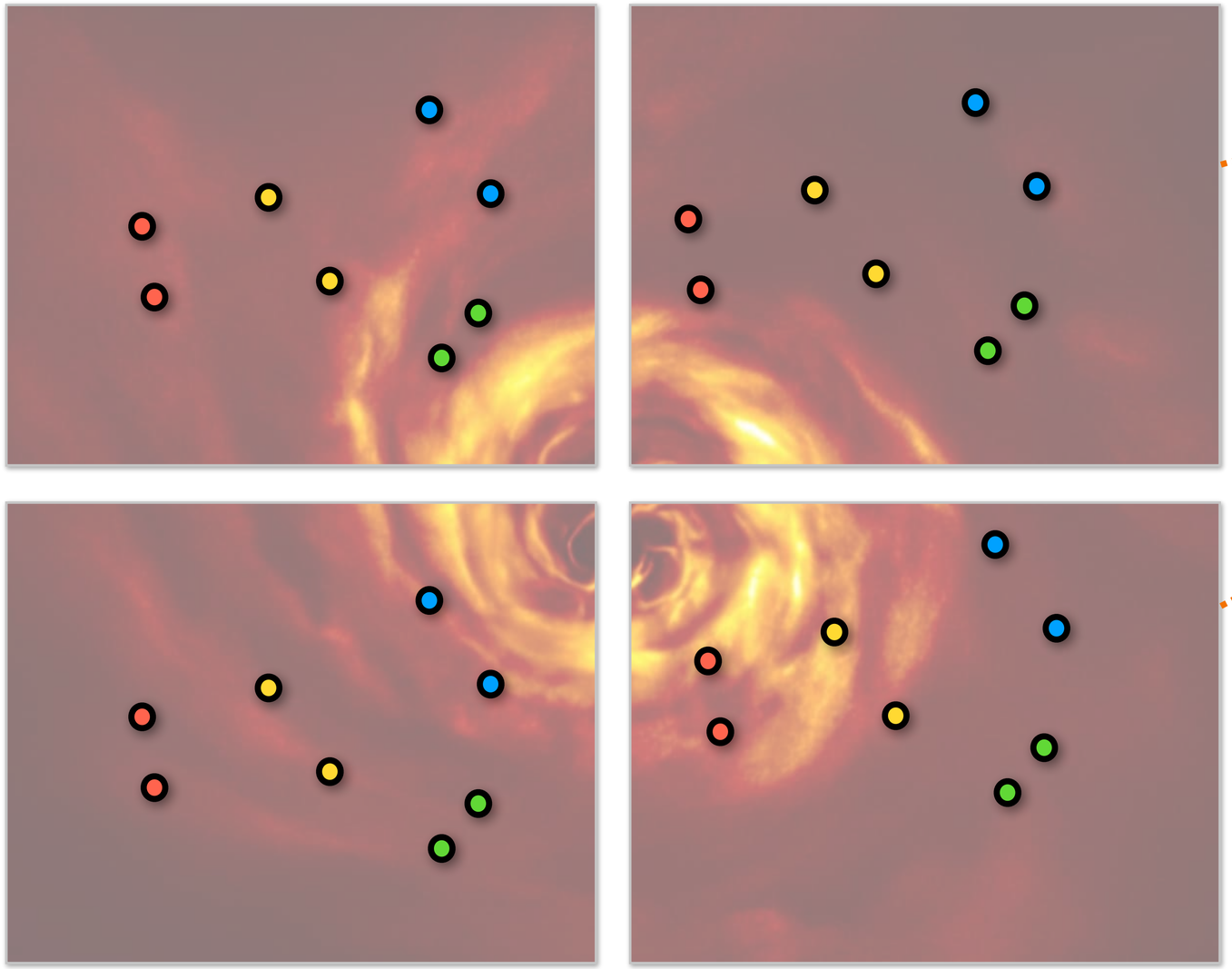
Global idea

Physical space :



Global idea

Physical space :



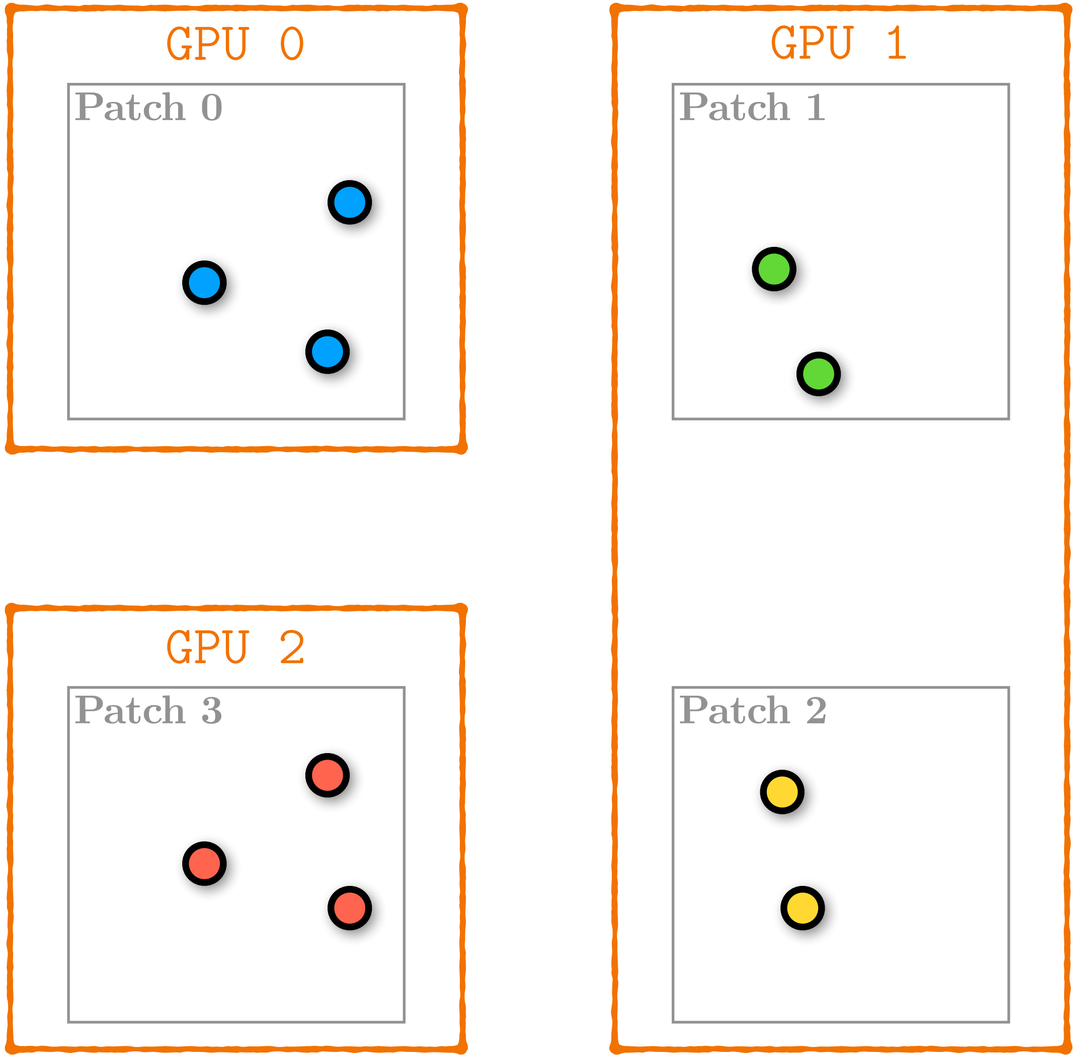
Patches

- Particles move between patches
- A patch can split
- A patch can merge

Domain decomposition

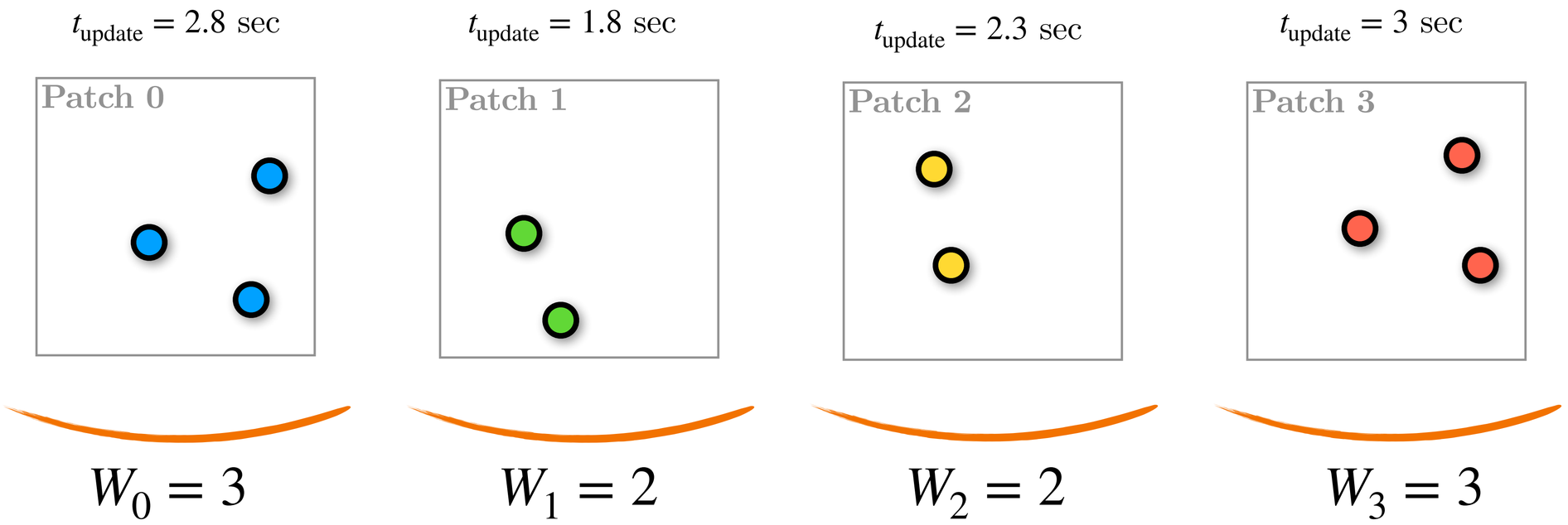
- Patch are distributed across GPUs

Patch decomposition
independent
of the GPU count !



Domain decomposition

- We attribute a "estimated load" to each patch
- We load balance according to that load

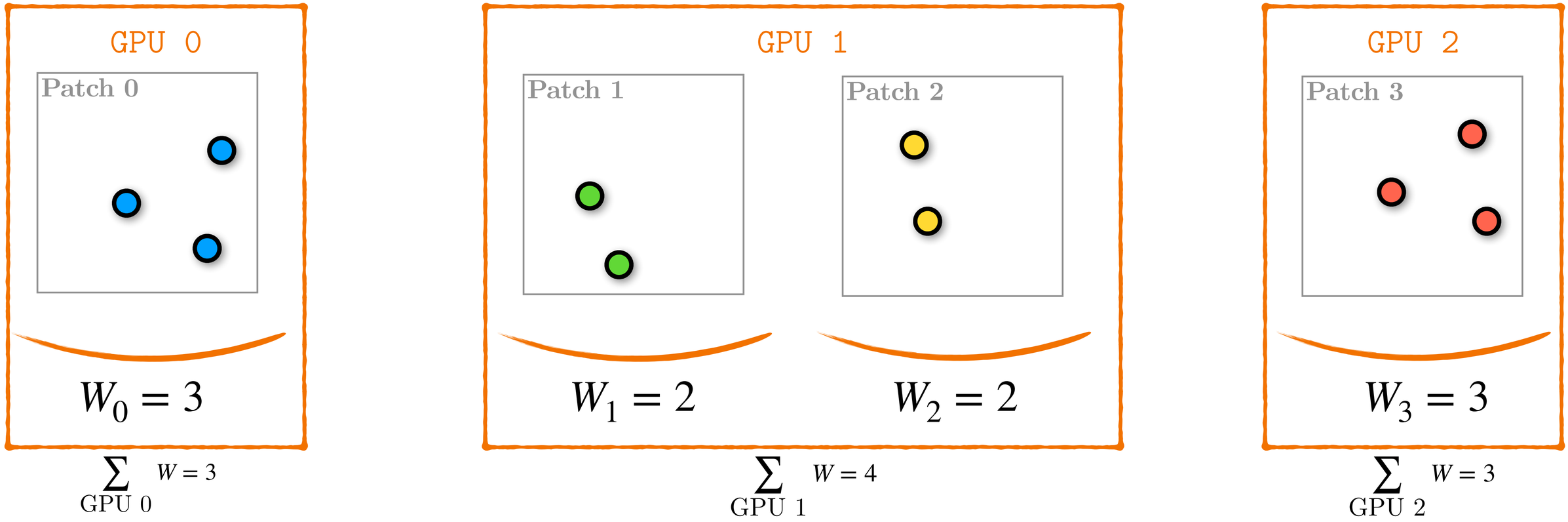


Simple load model : $W_i = N_{\text{part},i}$

Domain decomposition

- We attribute a “estimated load” to each patch
- We load balance according to that load

  (Using hilbert curve for ordering)

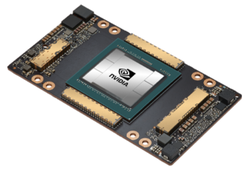


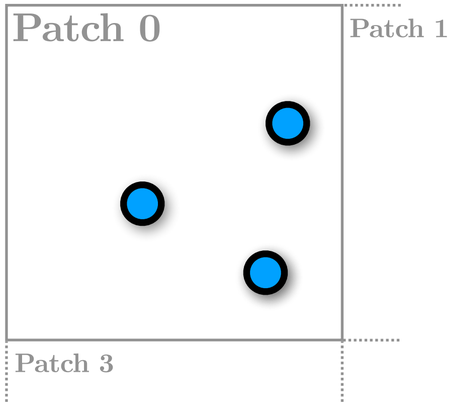
Perfect load model = optimal load balancing !

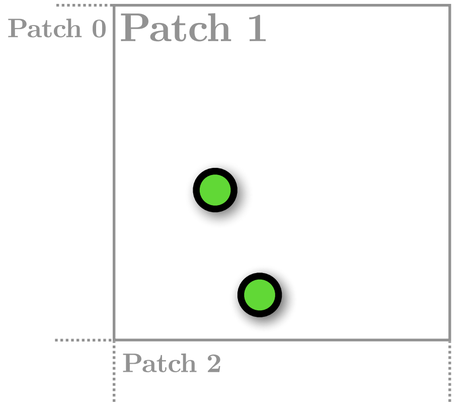
(Motivate better load models)

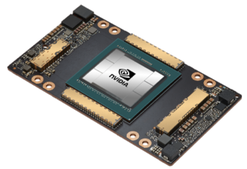
Domain decomposition

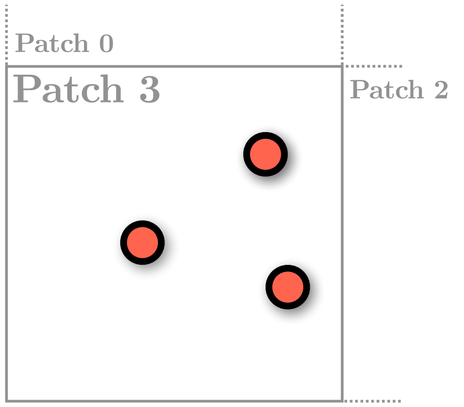
- Need to communicate ghost zones

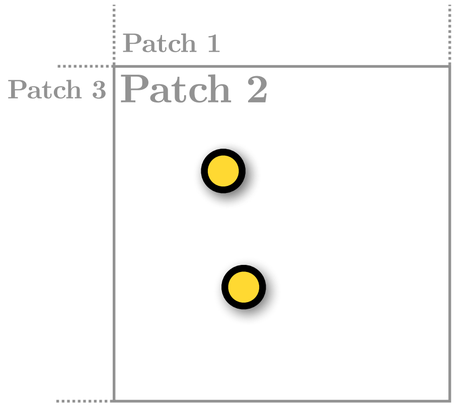






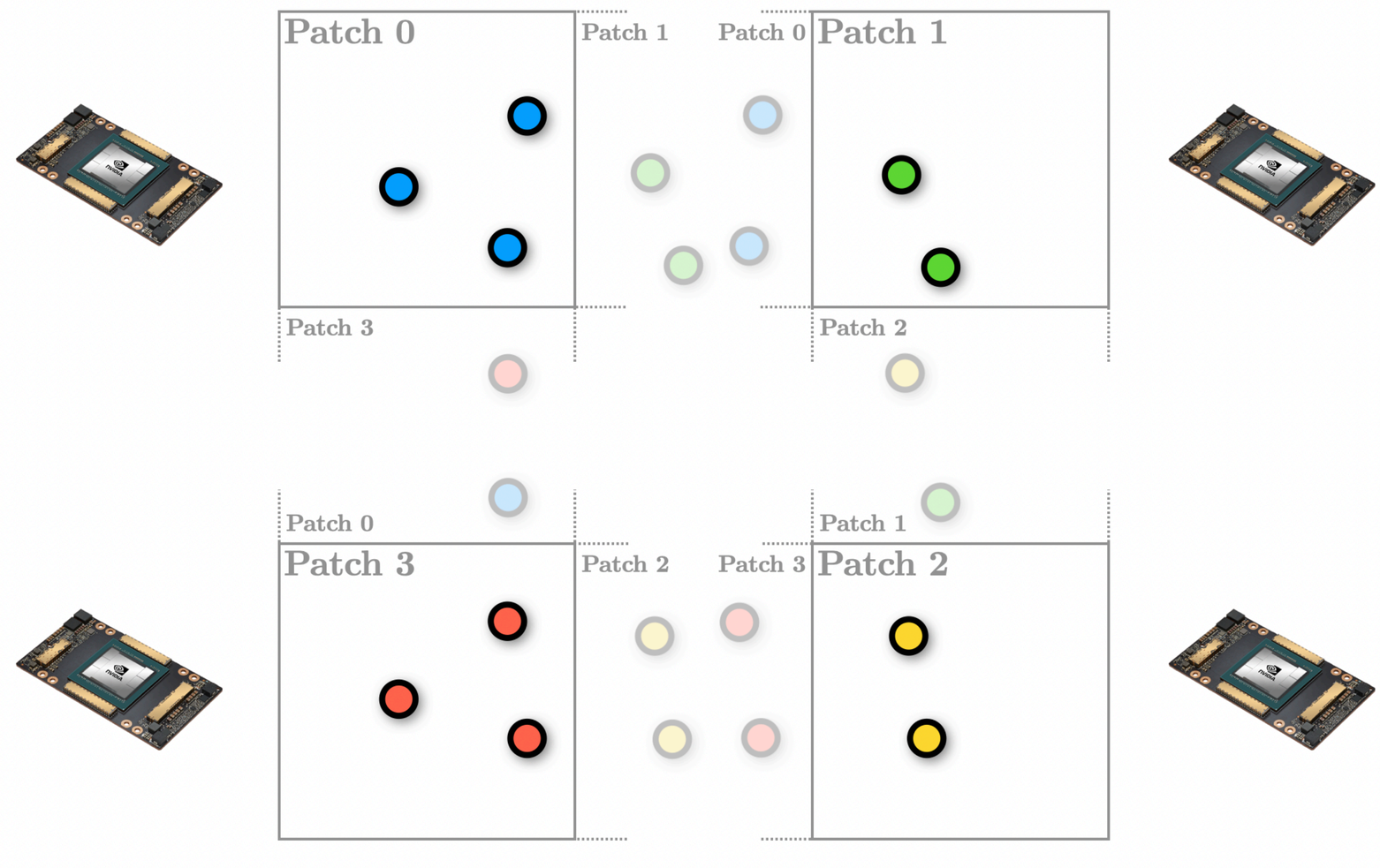






Domain decomposition

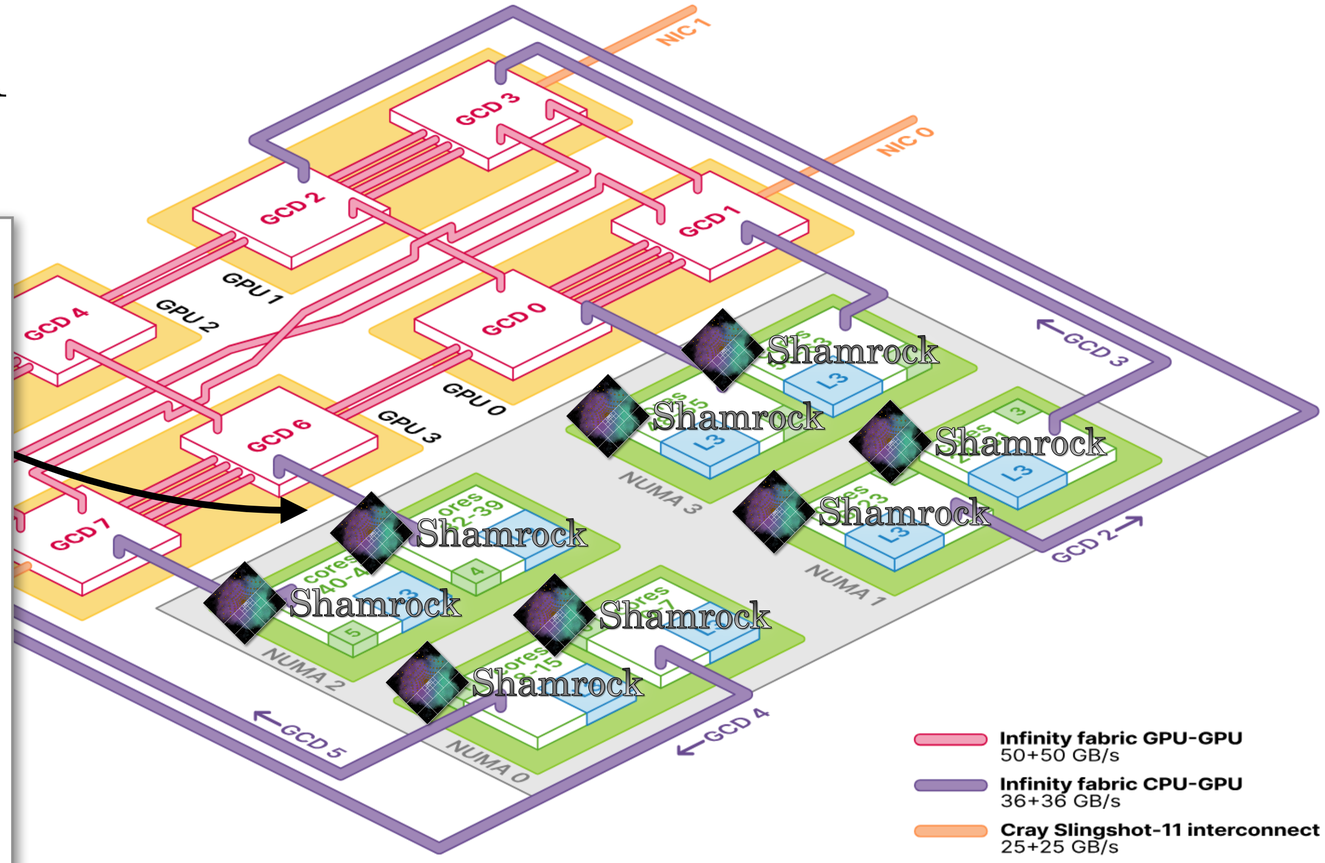
- Need to communicate ghost zones
- Shamrock uses sparse MPI communications



GPU nodes (200 - 10000)

GPU-CPU link depends on NUMA

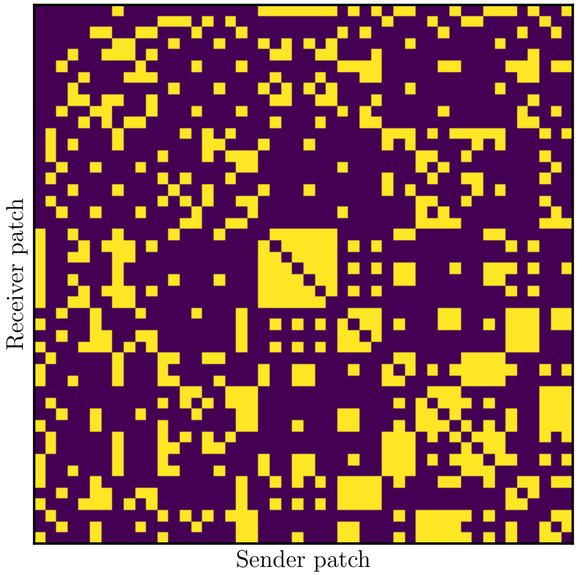
```cpp
#include <mpi.h>
#include <SYCL/sycl.hpp>

int main(int argc, char **argv){
    sycl::queue q {}; //select before MPI init

    MPI_Init(&argc, &argv);
    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    // fill with data
    int * ptr = sycl::malloc_device<int>(100, q);

    // Direct GPU comm
    if (rank == 0) {
        MPI_Send(ptr, 100, MPI_INT, 1, 0, MPI_COMM_WORLD);
    }else {
        MPI_Recv(ptr, 100, MPI_INT, 0, 0, MPI_COMM_WORLD,
            MPI_STATUS_IGNORE);
    }

    MPI_Finalize();
}
```

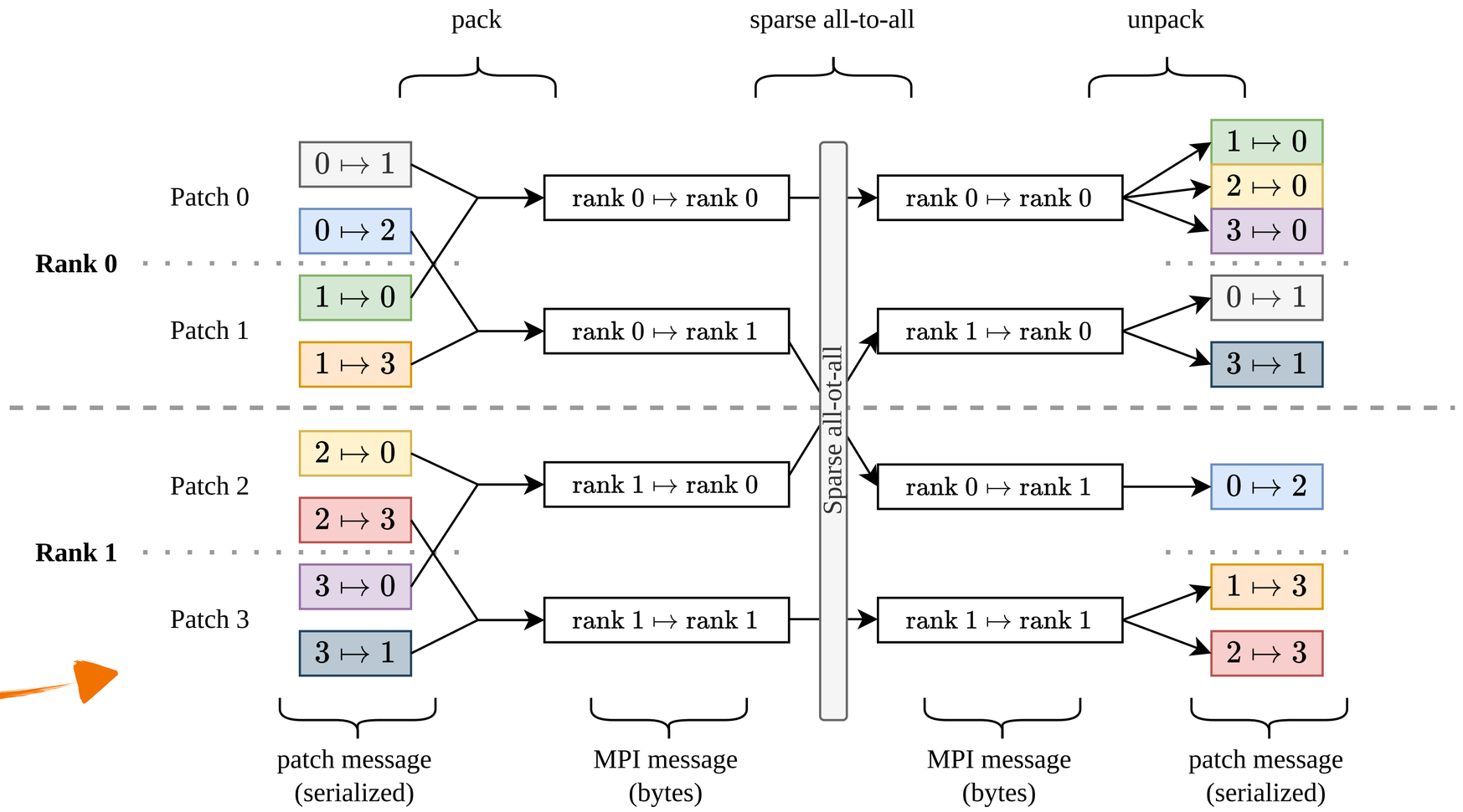


- 1 process per NUMA/GPUs
- Communications with MPI

Ghost zone graph

- Ghost zones graph = sparse matrix
- We perform sparse MPI communications



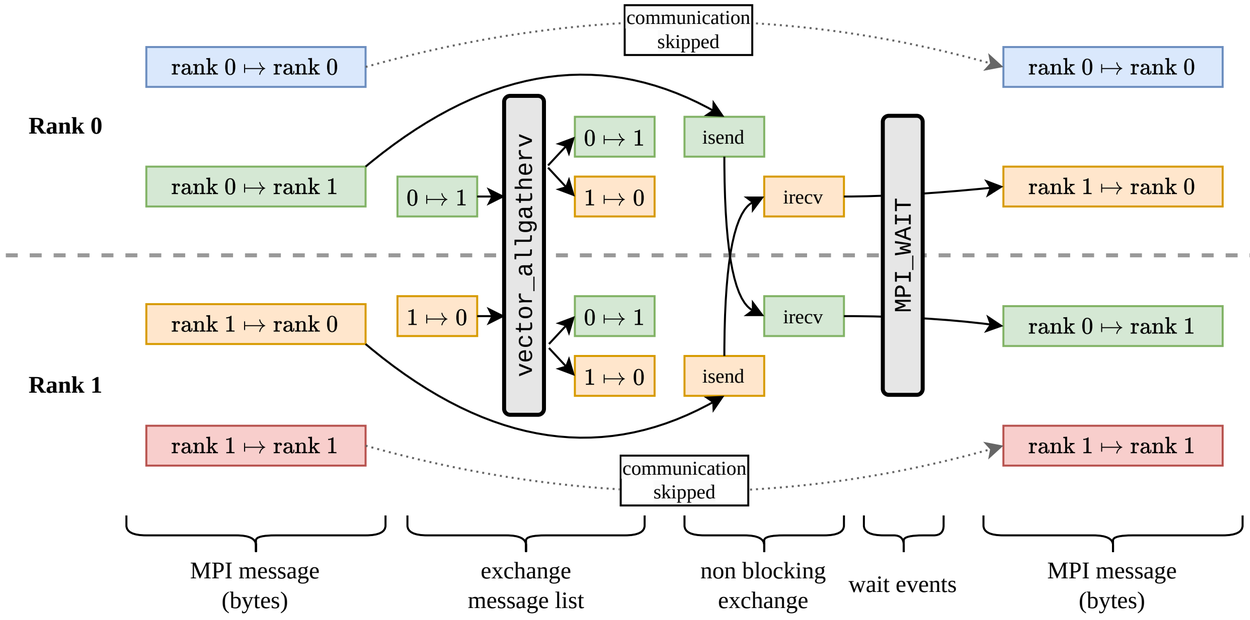


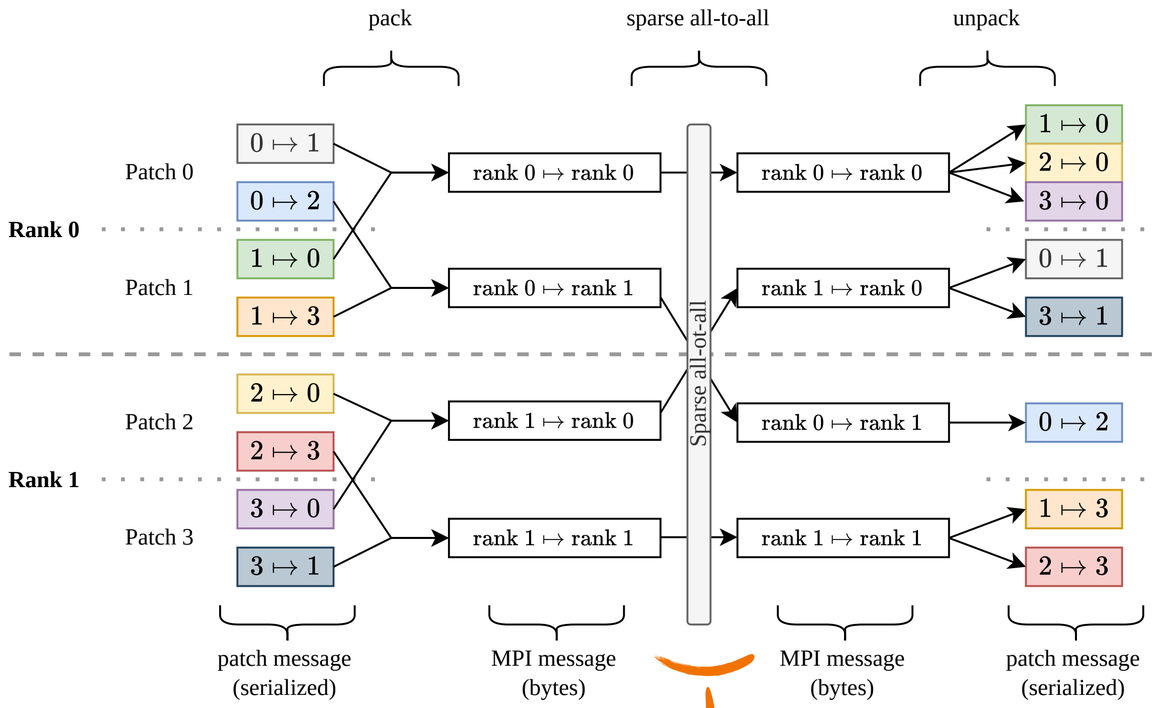
Messages independant of GPU count

Ghost zone graph

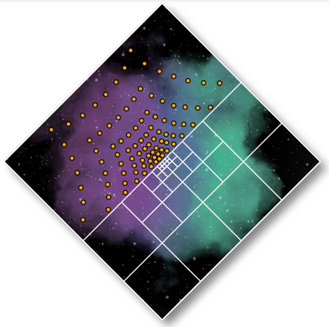
- Ghost zones graph = sparse matrix
- We perform sparse MPI communications

- Abstracted & can also be used for other communications

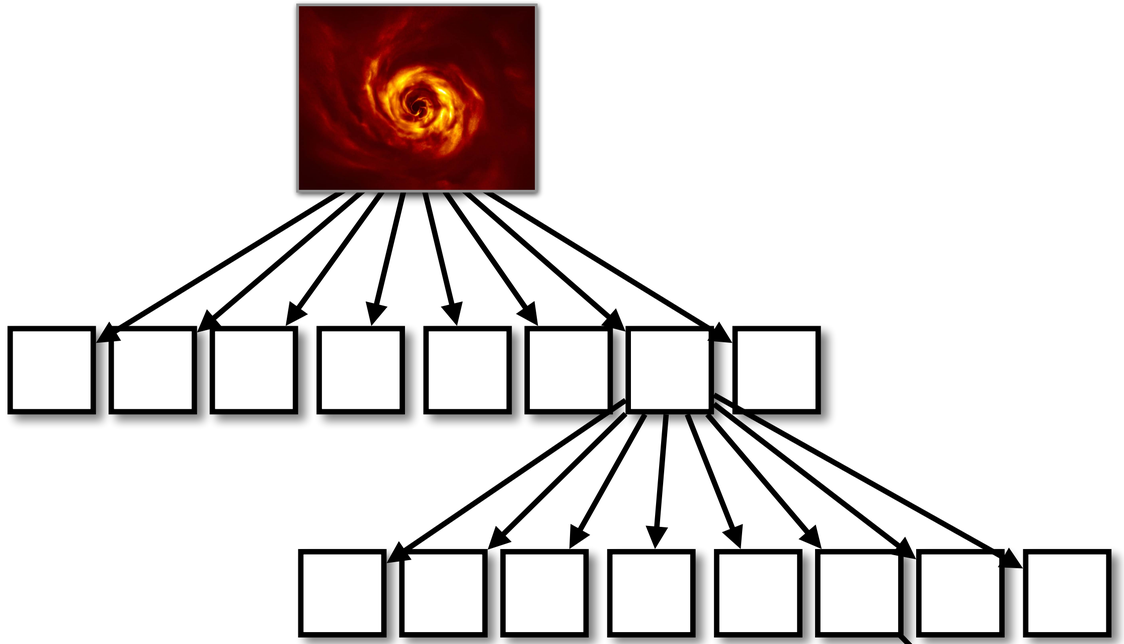




Global idea

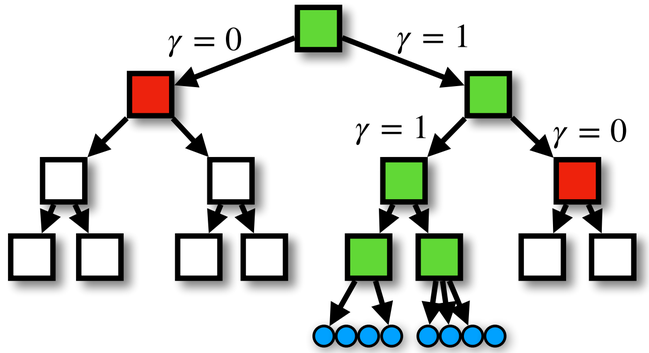


Shamrock



MPI Patch Octree :

Up to 42 levels



GPU radix tree :

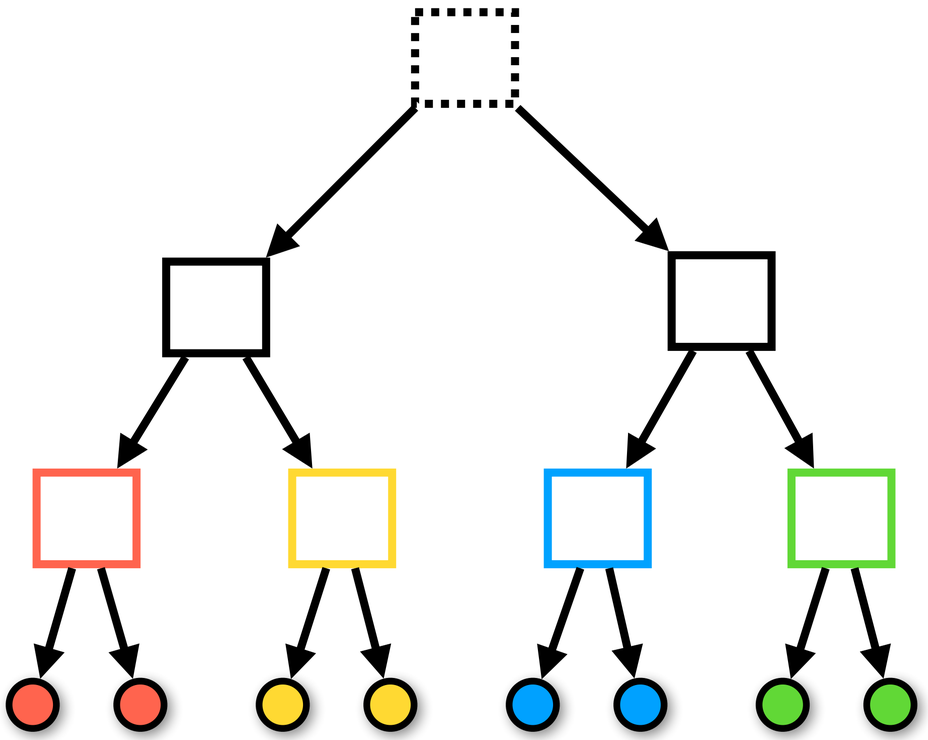
Up to 21 levels

63 levels = 2^{63} in scale resolution (2^{63} mm = 0.97 light year)

Tree building

Tree (BVH) :

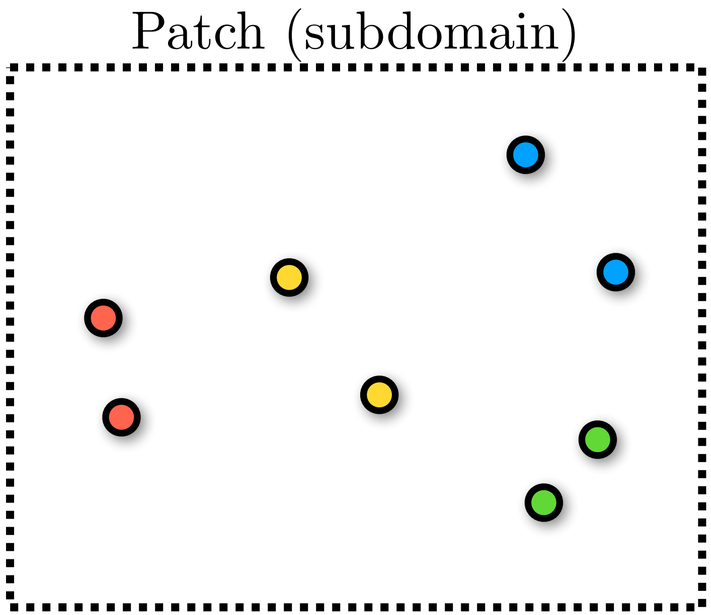
Bounding Volume Hierarchy

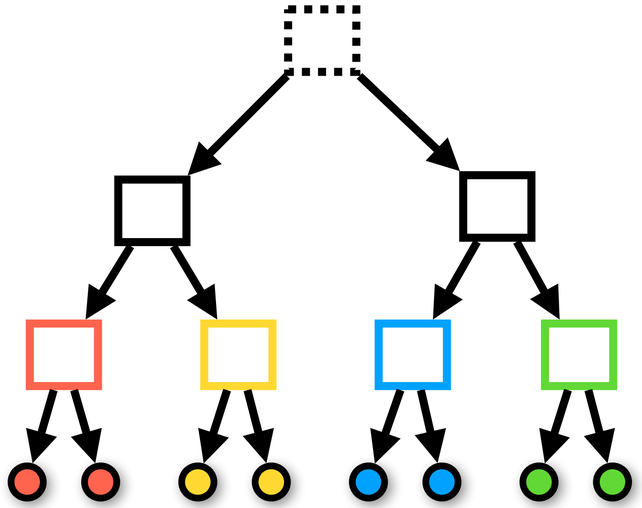


Building the tree on GPU :

- Tree update : too slow ❌
- On-the-fly parralelized tree building ✅

A novel parallel tree for SPH





Limitations :

- Only 1 particles per tree leaf
- Can not handle duplicates
  (2 particles with same morton code)

Map on fractal curve
(Z-order)

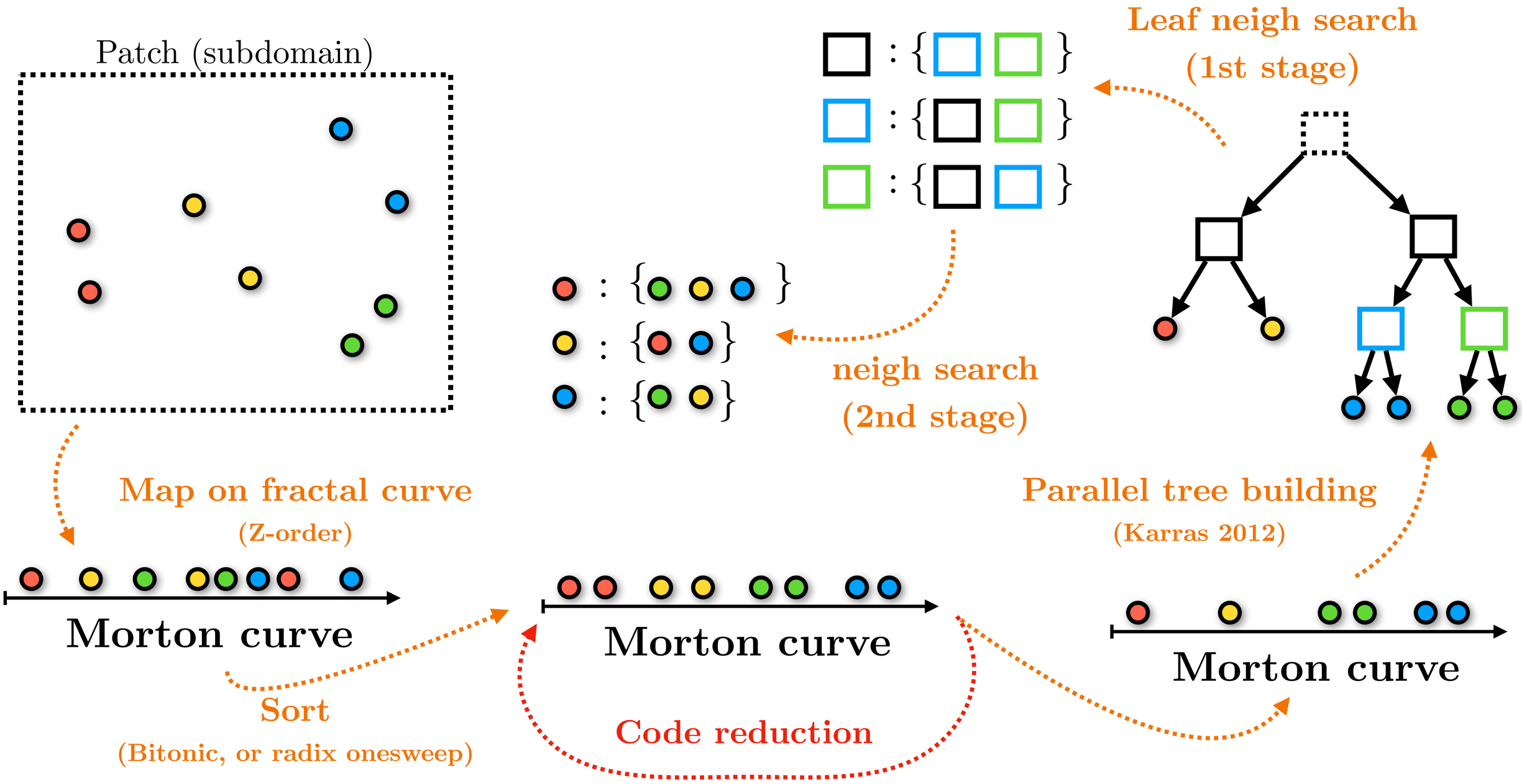
Parallel tree building
(Karras 2012)

Code reduction

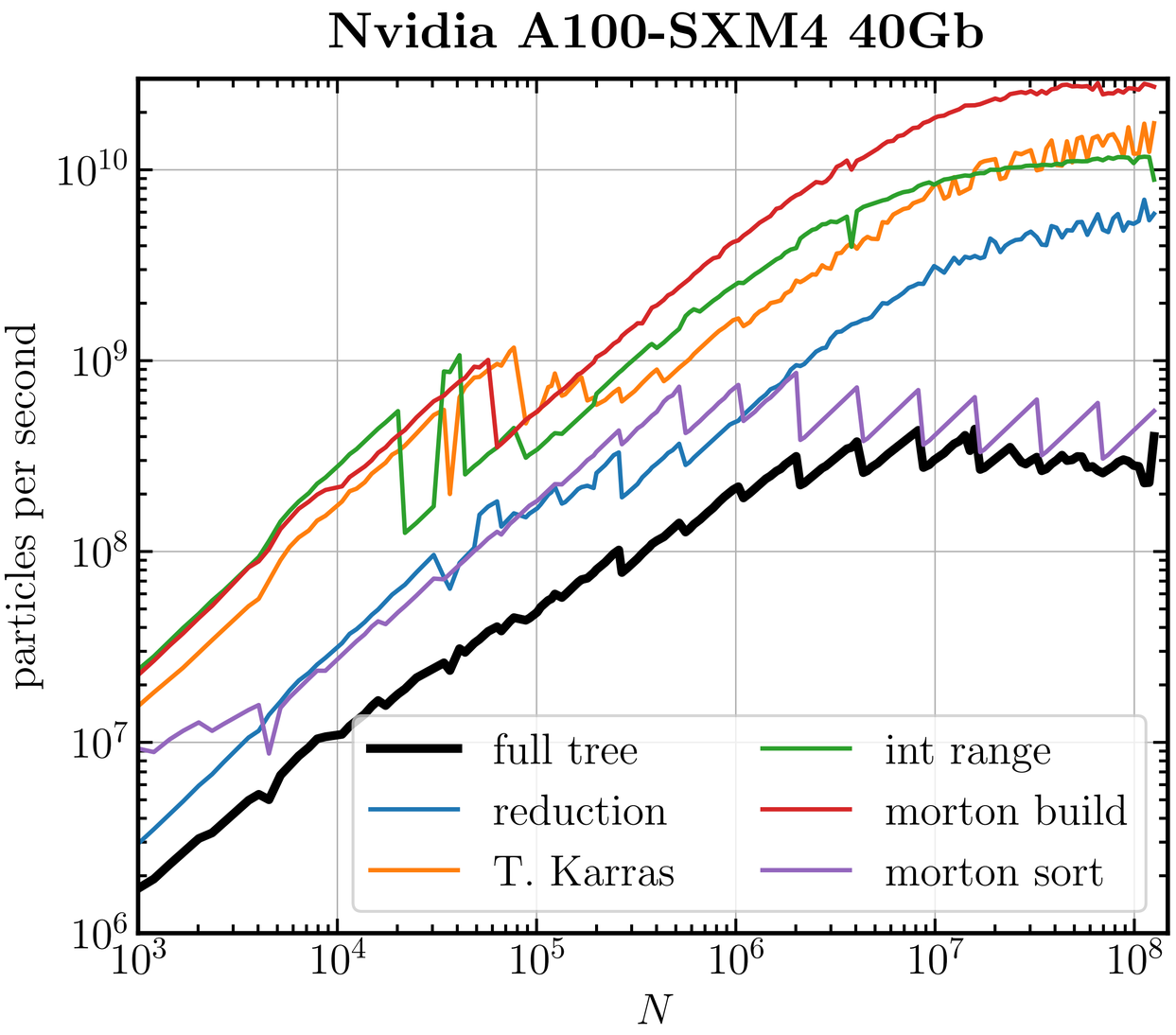
Solution :

- Reduction algorithm
  - Exploit binary arithmetics
  - Remove duplicates
  - Reduce the number of leaf before tree building

A novel parallel tree for SPH



Parallel tree building performance



Tree building in SHAMROCK :

- **Fully parallel on single-GPU**
- **$>$200M parts/second** ($\simeq$ 5% timestep)

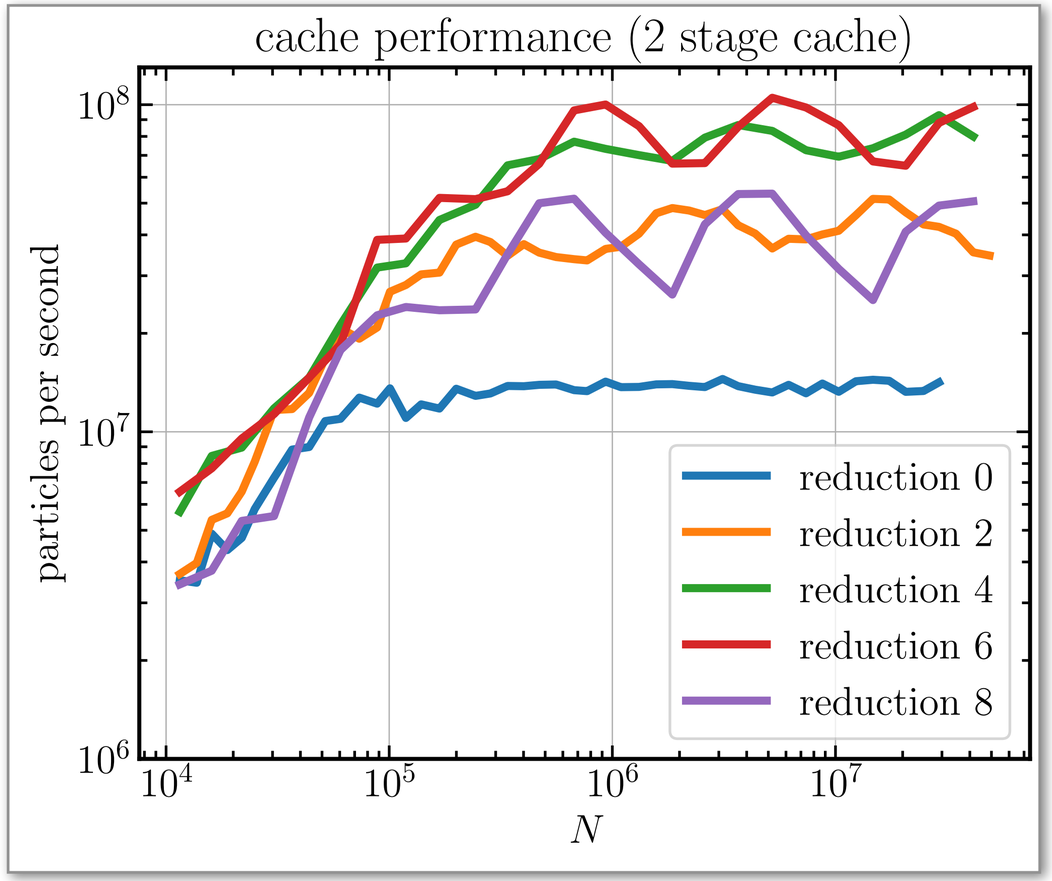
$\Rightarrow$ **Tree building is very cheap**

(No need to update it)

But :

- **Could reduce allocation latency**
- **Could increase perf of the sort**

Shamrock tree traversal



Ideal case :

- **Similar performance to tree build**

Tree traversal in SHAMROCK :

- **Fit in registers (maximum occupancy)**
- **2-stages optimised by reduction**
- **100M parts/second (SPH criterion is complex)**

But :

- **Could exploit RT units**
- **Could have better memory access patterns**
- **Could have better task balancing/scheduling**

Tree with reduction algorithm : $10^7 \rightarrow 10^8$ (parts/sec)

Performance !!!!

Reference code:

Phantom SPH code

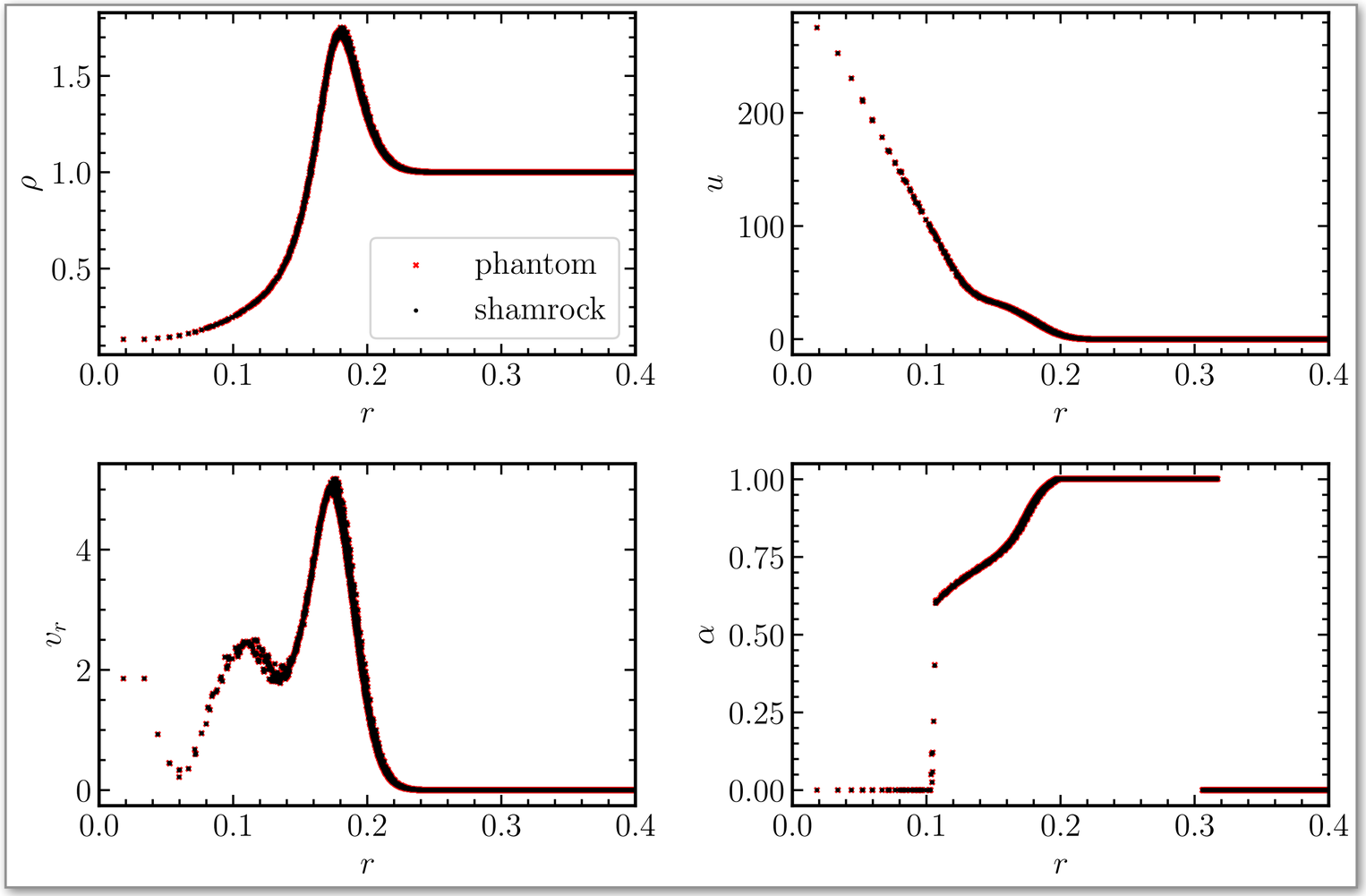
Used for 15 years, 450 citations

But:

- No MPI scalability
- No GPUs

Comparison with Phantom

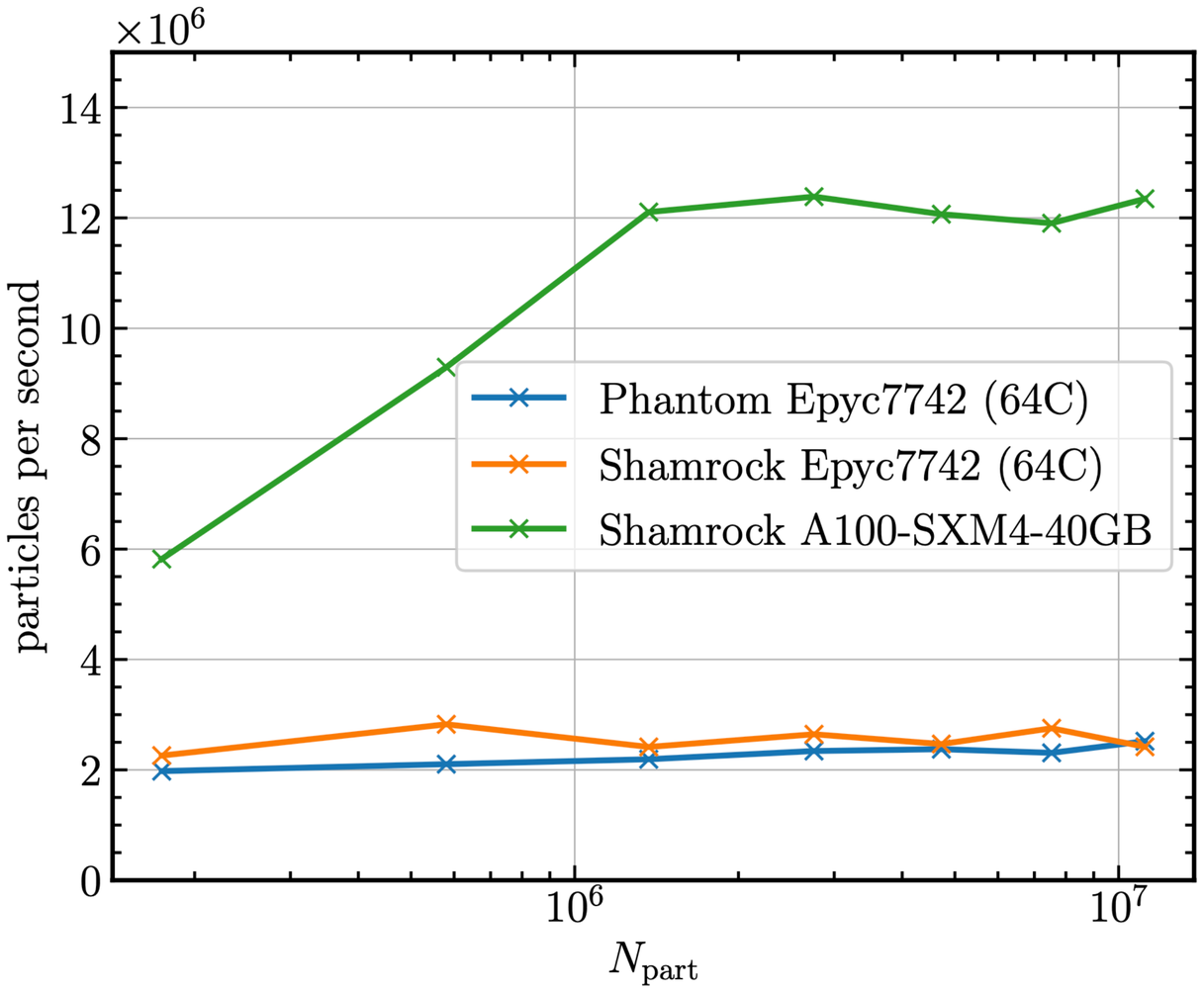
Sedov-blast wave:



Reproduce Phantom solver !

Single GPU performance

Sedov-blast wave:



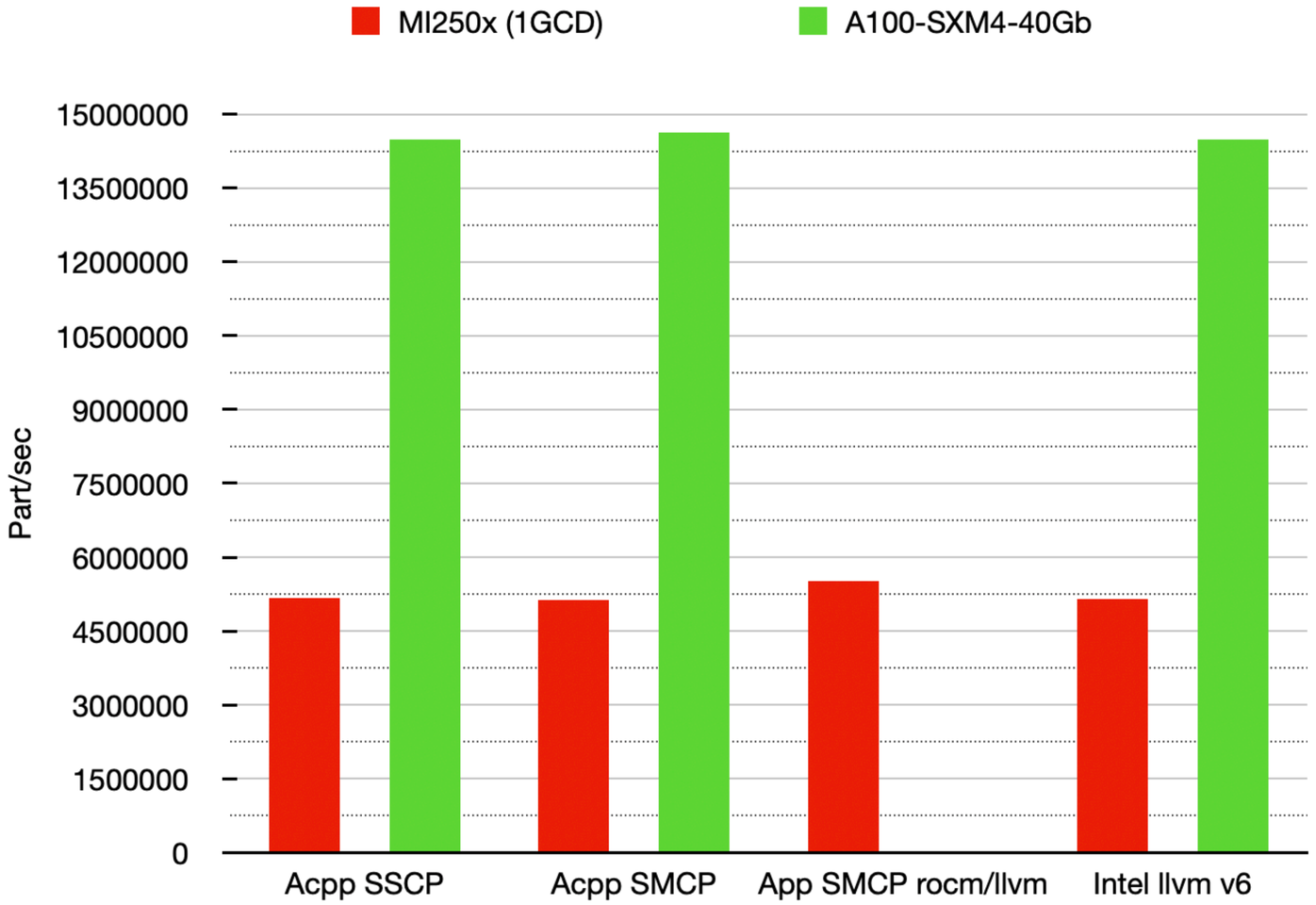
Intel/llvm
CUDA backend

AdaptiveCpp
Omp backend

⇒ **Acceleration by a factor 6 CPU⇒GPU**

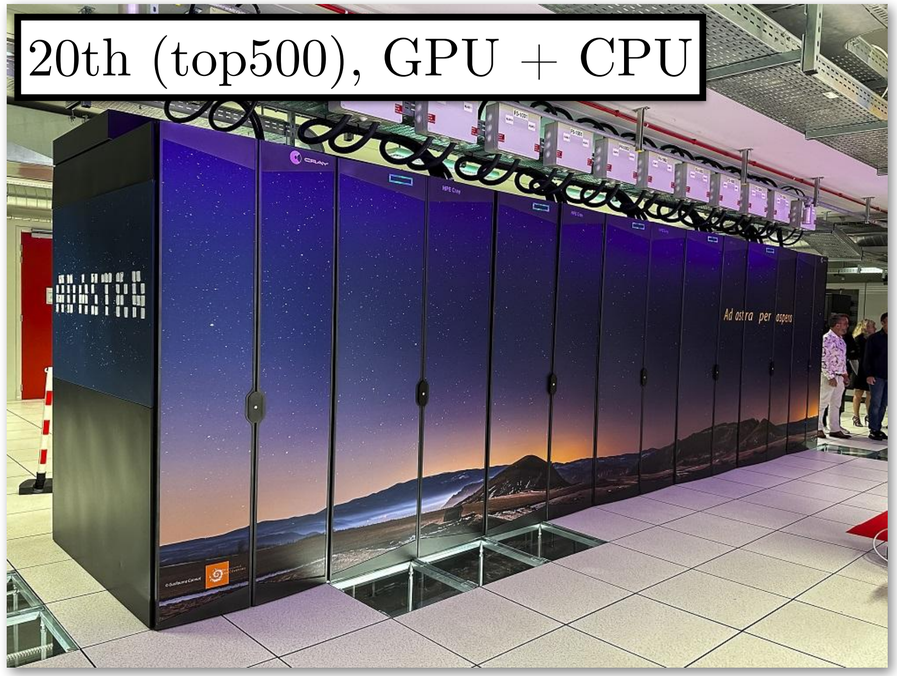
Single GPU performance

Sedov-blast wave:



Similar performance across implementations/backends

SPH on Multi-GPU (Adastra)



338 nodes, 4 MI250x / Nodes

921kW (max)

61.6 PFlop/s

RANKING

List	Rank	System					
11/2023	17	HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11	HPE	319,072	46.10	61.61	921.48

HPC wire

Since 1987 - Covering the Fastest Computers in the World and the People Who Run Them

Home

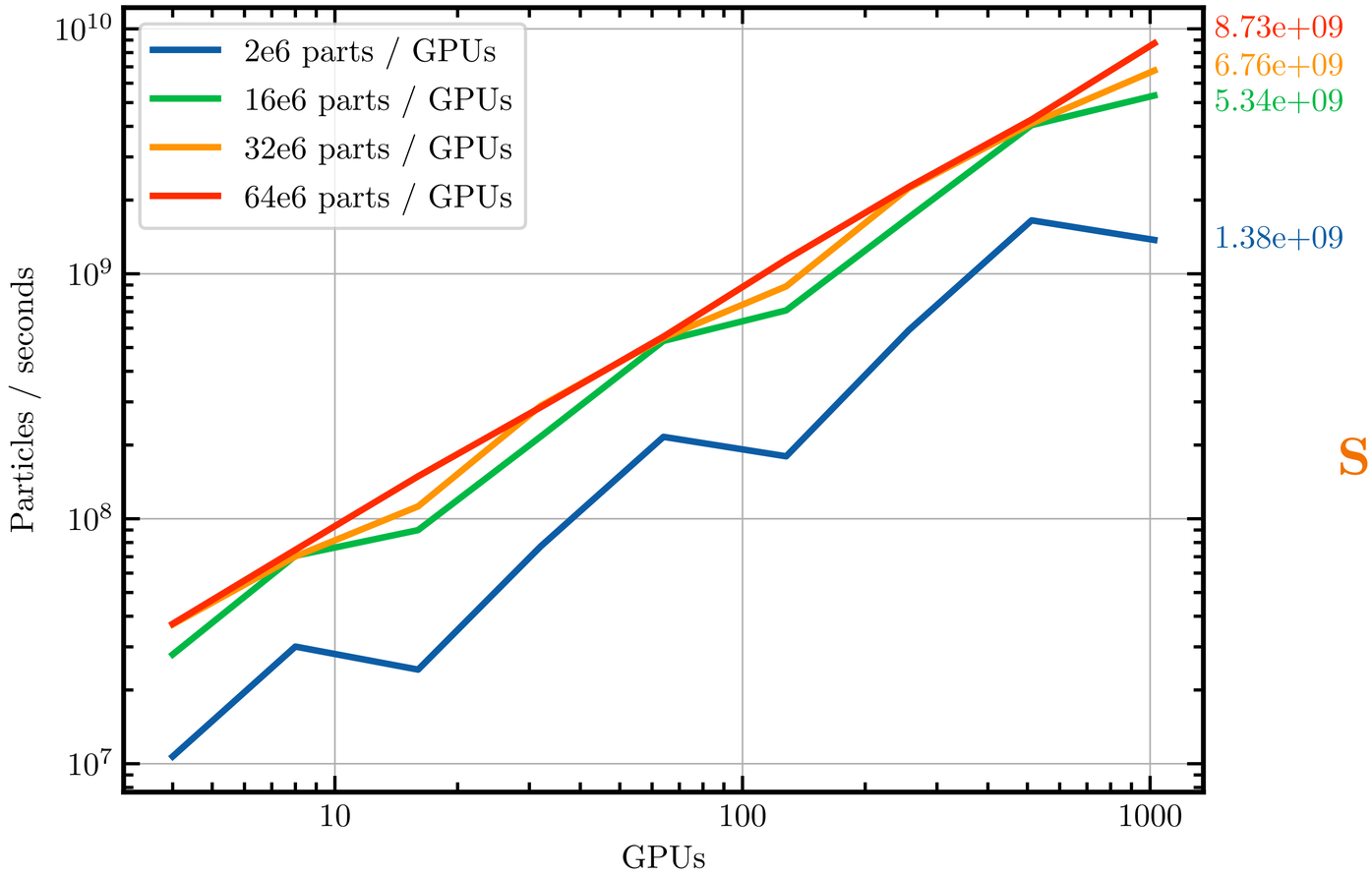
Topics

GENCI's Adastra Marks a New Step Towards a More Sustainable HPC

November 25, 2022

Nov. 25, 2022 — The energy efficiency of GENCI (Grand Equipement National de Calcul Intensif)'s new supercomputer Adastra has been improved to reach now 58.2 GF/W, ranking Adastra at #3 position on the new Green500 list announced during SC22. Adastra is ranking #11 in the November 2022 Top500 list with 46.10 PFlops measured performance.

Weak scaling (Sedov-blast)



Shamrock :

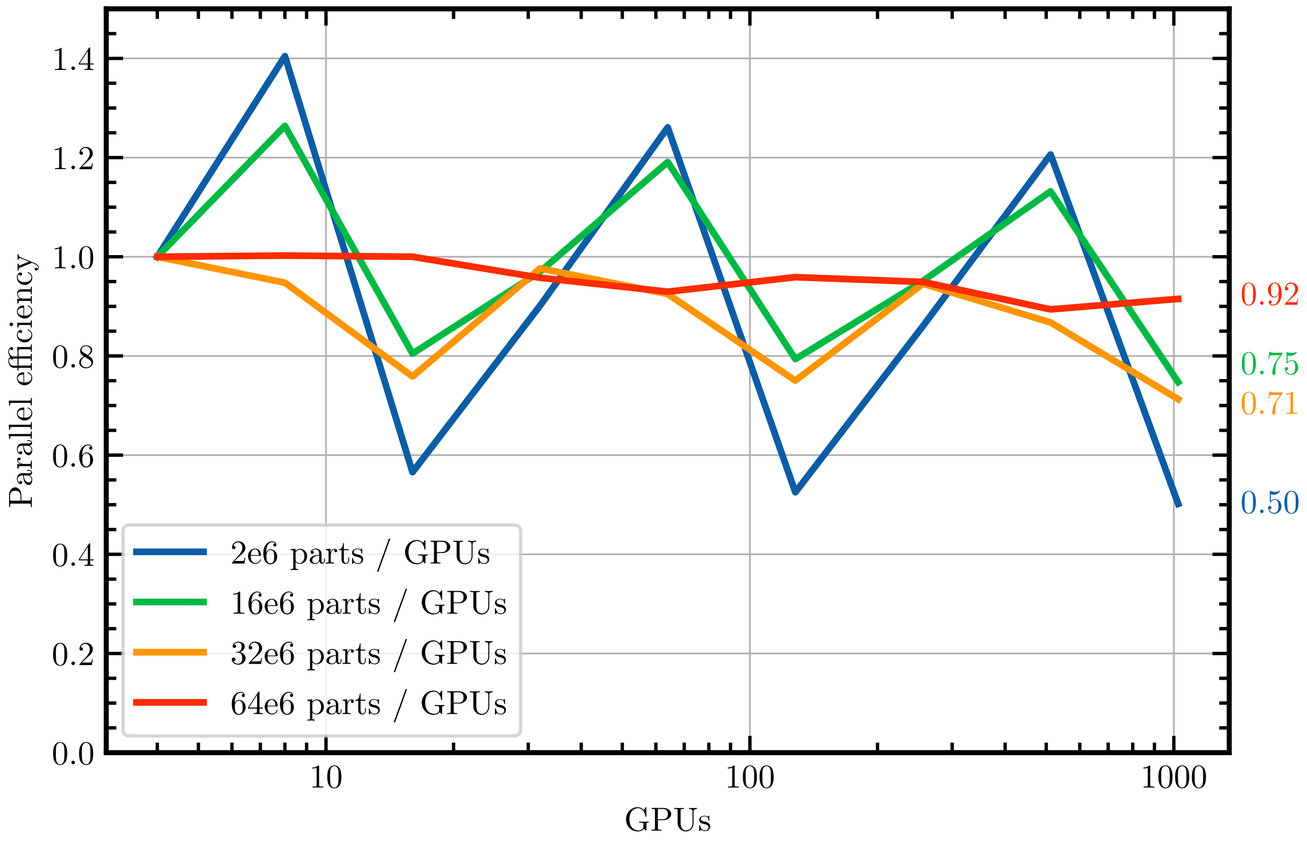
- **65.000.000.000 particles** (4000^3)
- **9G part / seconds**
- **7 sec / iterations**
- **1024 GPU (MI250x)**

Reference :

Same test with Phantom SPH = 2e6 part / seconds

$\longrightarrow$ **x4500 speedup !**

Weak scaling (Sedov-blast)

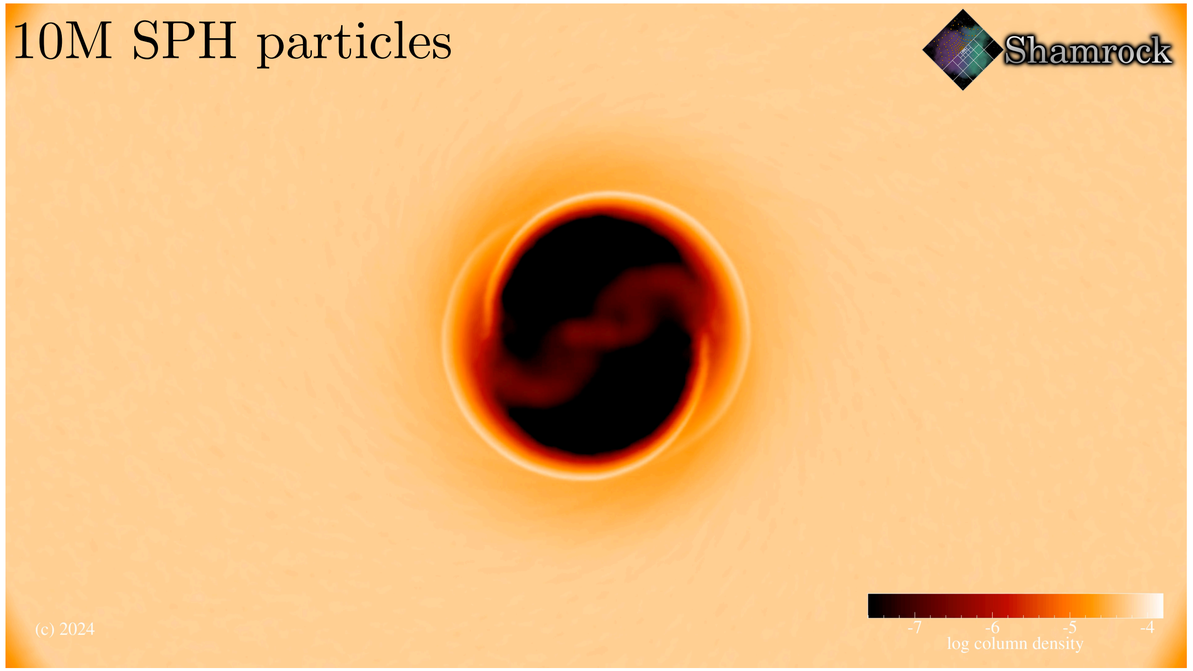


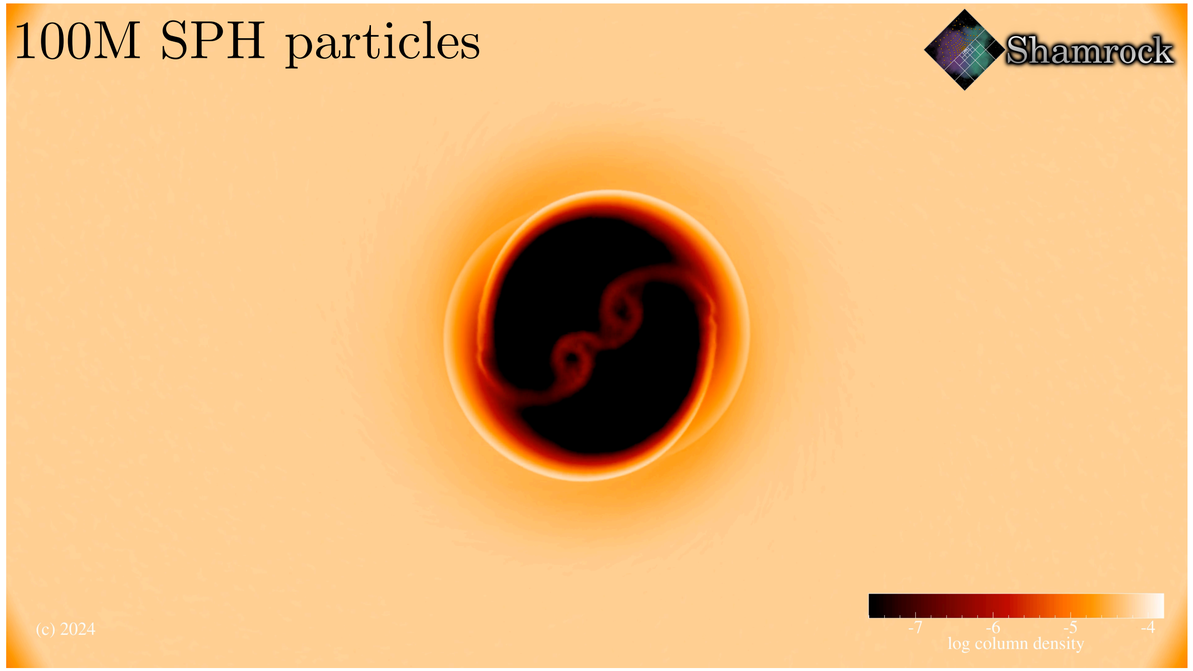
Weak scaling of Shamrock SPH solver :

92% parallel efficiency

(Large simulations)

Exemples : circum-binary discs

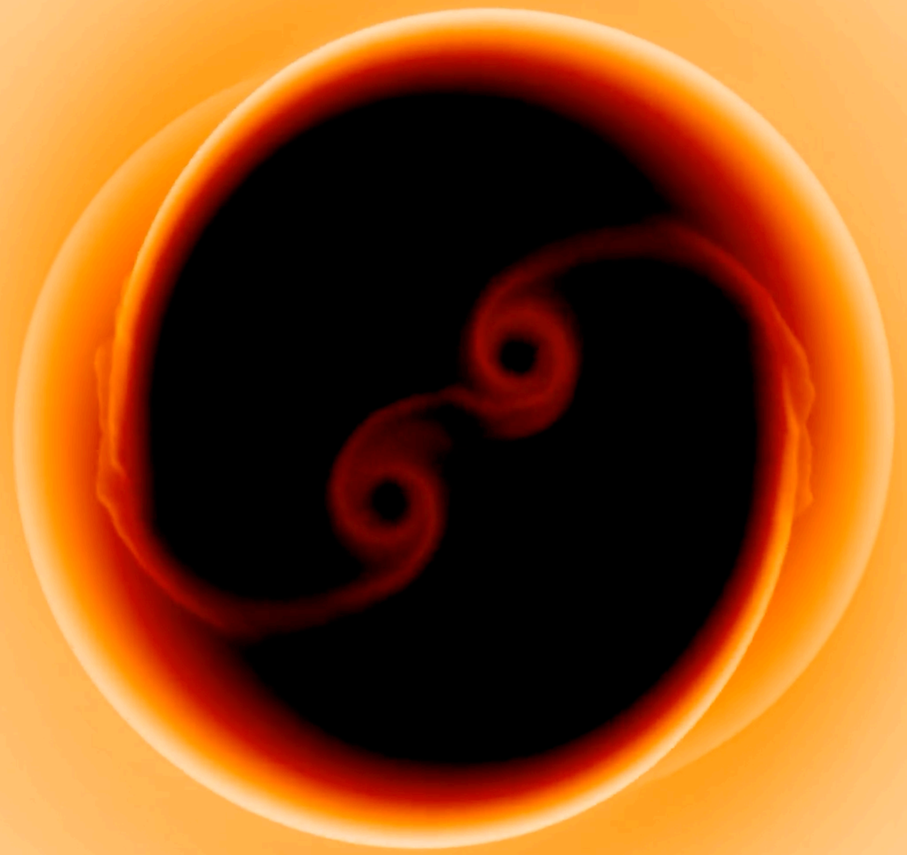




Ok but on many GPUs ???

1G SPH particles

8 nodes (32 MI250X)



Shamrock

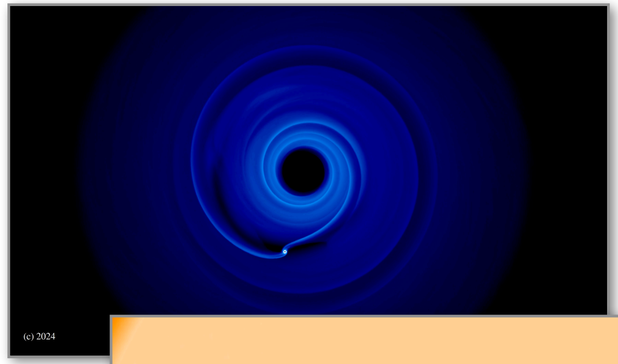
Conclusion

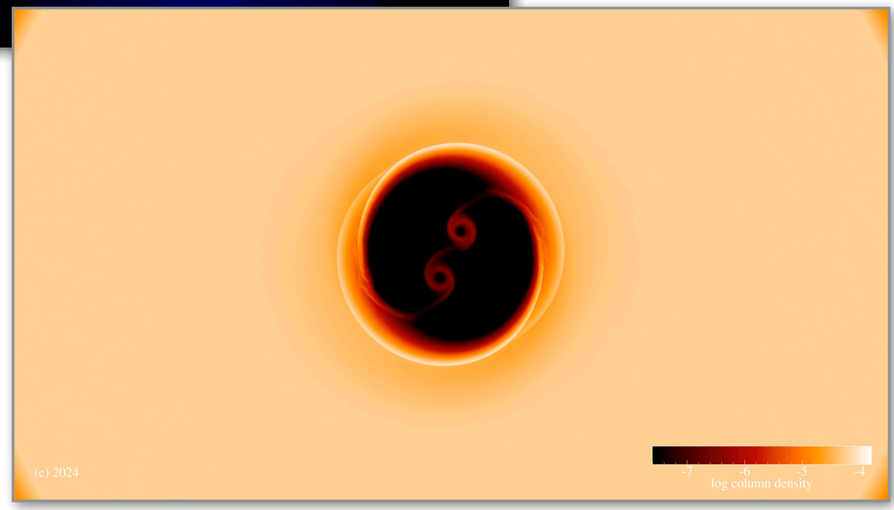
Currently :

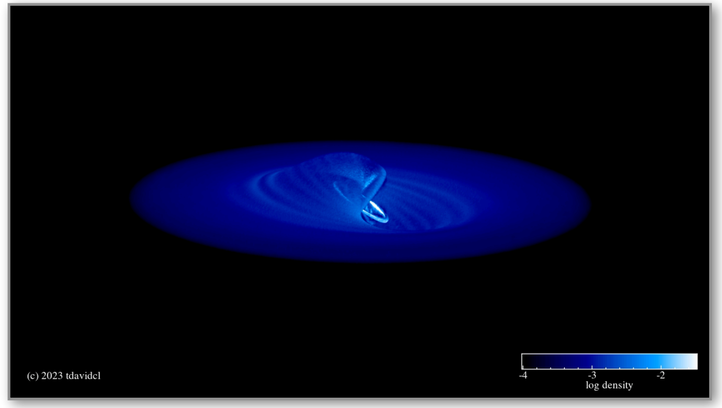
- On the fly built trees
- Patch based decomposition
- SYCL allow us to target all GPUs
- 92% SPH weak scalability on uniform tests

Next :

- Exascale disc simulations
- More optimizations (e.g. performance, latency)
- More physics
- Load modelling







Questions ?

No. 101053020 (Dust2Planets)