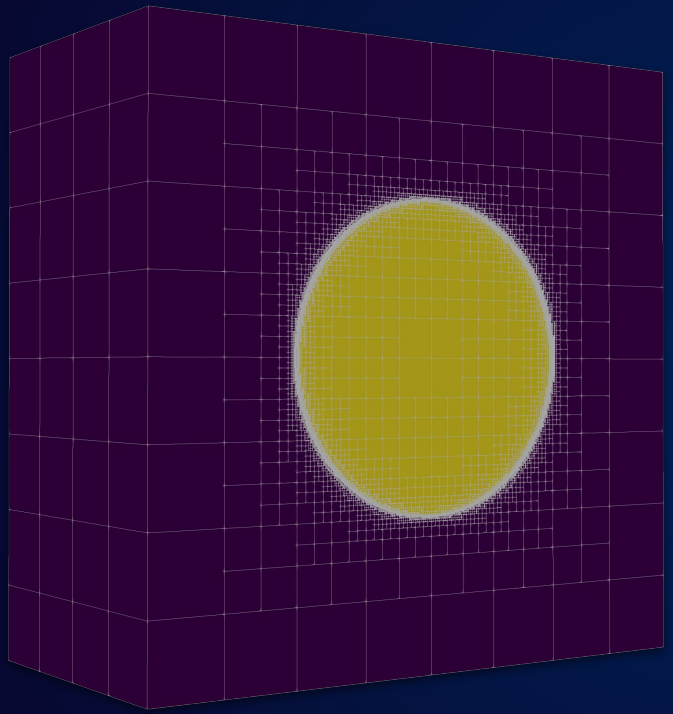


DYABLO

A new hardware-agnostic AMR code for the Ramses community

Arnaud DUROCHER
(CEA - IRFU/DEDIP/LILAS)

RUM2021 - 29/09/2021

THE GINEA GROUP

Groupe d'investigation numérique pour l'exascale en astrophysique

- **GINEA**
  Work group to find ways to overcome the barrier to Exascale that the Astro community is facing

- **Towards Exascale:**
  - *New architectures (ARM, GPUs)*
  - *Bigger machines*

- **Specificity of the Astro community:**
  - *Adaptive Mesh Refinement*
  - *Multi-physics simulations*
  - *Large data production*

Ginea / Dyablo :

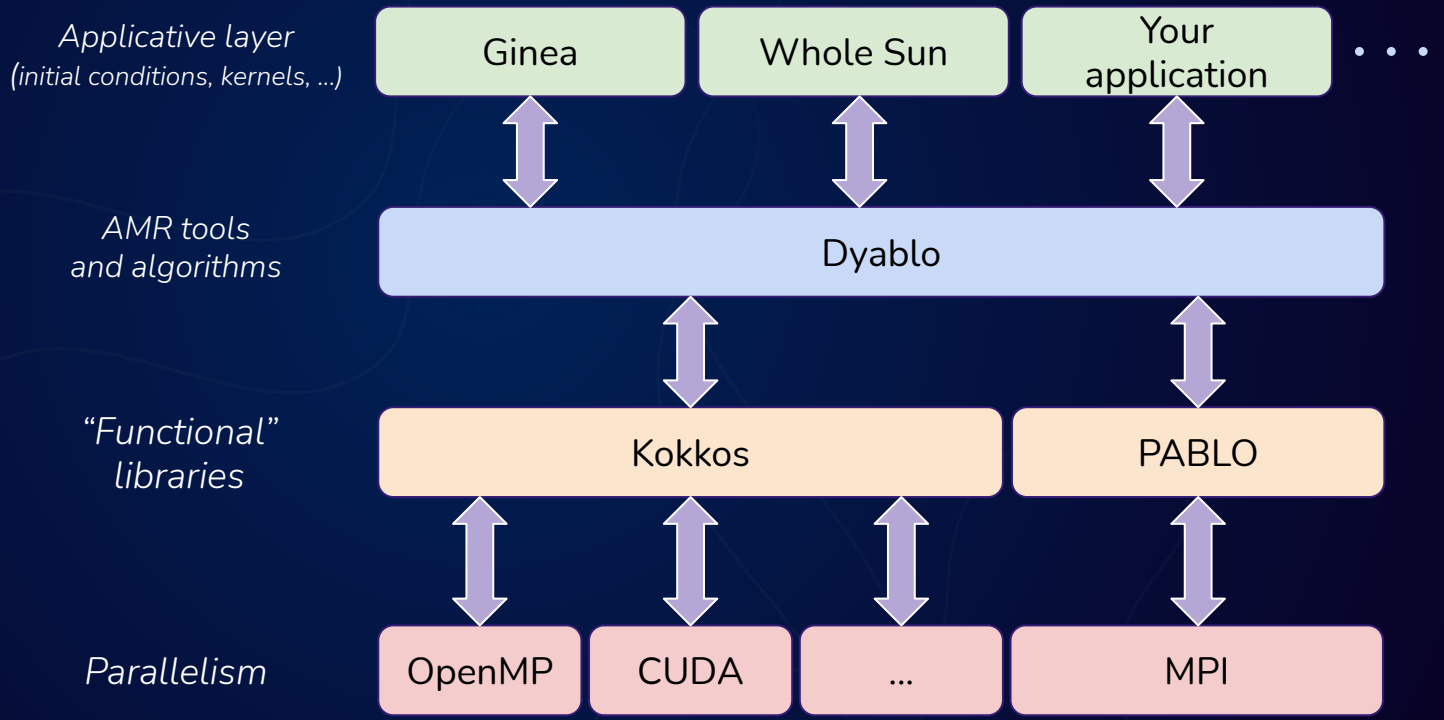
B.Commercon (CRAL),
C. Cadiou (UCL),
Y. Dubois (IAP),
F. Bournaud (CEA),
J. Fensch (CRAL),
L. Michel-Dansac (CRAL),
M. Trebitsch (Groningen),
M. Delorme (CEA),
P. Tremblin (CEA),
P. Ocvirk (Obs. Strasbourg),
Y. Rasera (Obs. Paris)
B. Thooris(CEA),
D. Chapon (CEA),
D. Aubert (Obs. Str.),
J. Blaizot (CRAL),
J. Rosdahl (CRAL),
L. Strafella (CEA),
M. Gonzalez (CEA),
O. Abramkina (IDRIS),
P. Kestener (CEA),
V. Reverdy (ENS),

DYABLO'S PHILOSOPHY

A modern and modular adaptive mesh refinement (AMR) with hardware-agnostic performance for the astrophysics community.

- **Modern and modular :**
  - *C++ (>C++14)*
  - *Applicative layer (plugins) to add physics modules*
- **Hardware-agnostic :**
  - *Kokkos library : CPU (OpenMP), GPU (CUDA) and future architectures*
- **Adaptive Mesh Refinement :**
  - *Using the PABLO library for AMR (CPU-only)*
  - *Developing our own AMR algorithms (CPU/GPU)*
- **For the astro community:**
  - *Aimed at the RAMSES community (but not exclusively)*
  - *Hydro, self-gravity, particles, etc...*
  - *Exascale project : Dyablo-Ginea / Dyablo-Wholesun*

DYABLO'S PHILOSOPHY



KOKKOS

- **C++ hardware-agnostic parallel programming model**
- **Developed at *Sandia National Laboratory*, member of the *Department of Energy's Exascale Project* :**
  - *Efficient back-ends for the architectures used in US exascale supercomputers (present and future)*
  - *OpenMP, CUDA, SYCL, (ARM, Ponte-Vecchio)*
- **Principle :**
  - `Parallel_for` *: A C++ functor (lambdas) is applied on each element in parallel*
  - `Kokkos::View:` *Abstract multidimensional arrays (as in Fortran)*
  - *Architecture selected at compile-time :*
    `Kokkos::OpenMP, Kokkos::CUDA,` *…*

=> The same code can be compiled for different architectures

kokkos

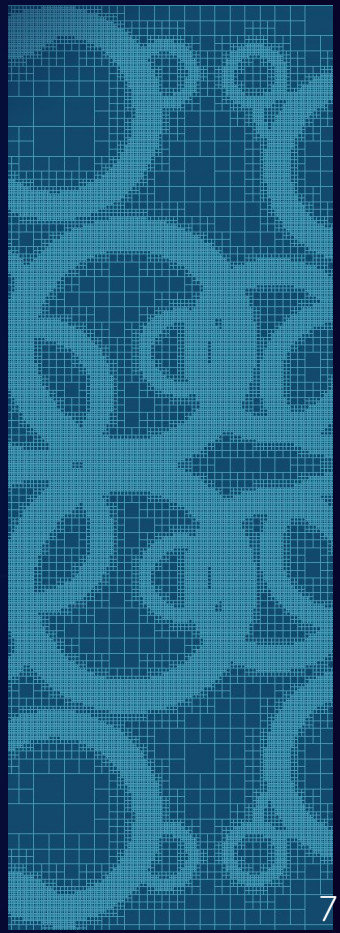
KOKKOS

```cpp
#include <Kokkos_Core.hpp>
#include <cstdio>

typedef Kokkos::View <double*[3]> view_type;

int main(int argc, char* argv[]) {
  Kokkos::initialize(argc, argv);

  {
    Kokkos::View <double**> a("A", 10, 3);

    Kokkos::parallel_for(
        10, KOKKOS_LAMBDA(const int i) {
          a(i, 0) = 1.0 * i;
          a(i, 1) = 1.0 * i * i;
          a(i, 2) = 1.0 * i * i * i;
        });

    double sum = 0;
    Kokkos::parallel_reduce(
        10,
        KOKKOS_LAMBDA(const int i, double& lsum) {
          lsum += a(i, 0) * a(i, 1) / (a(i, 2) + 0.1);
        },
        sum);
    printf("Result: %f\n", sum);
  }
  Kokkos::finalize();
}
```

PABLO

- **Bitpit : Open source C++ HPC library**

- **PArallel Balanced Linear Octree :**
  - *AMR on a cartesian grid (cube/square)*
  - *2:1 balancing*
- **MPI parallelism :**
  - *MPI domain decomposition and load-balancing*
  - *Can perform MPI communications of ghost cells*
- **Not like RAMSES**
  - *“Linear Octree” : Only leaves in the octree*



IMPLEMENTED FEATURES IN DYABLO

GEOMETRY

- **Cartesian AMR**
  - 2D / 3D
  - Block-based / Cell-based

PHYSICS

- **Hydro Demo**
  - Finite-volume
  - MUSCL-Hancock scheme
- **Gravity**
  - Static
  - WIP : Self-gravity
- **Particules** (WIP)
- Application specific (Whole Sun): (Thermal conduction, viscosity, ...)

IO

- Native **Paraview/Visit** output
- Parallel **HDF5/XDMF**
- Working on compatibility with other tools

PARALLELISM

- **MPI** (through PABLO or custom AMR)
  - Domain decomposition
  - Load-balancing
- **Compute kernels using Kokkos**
  - OpenMP / CUDA

AMR ON GPU

Challenges of AMR on GPU:

- **No GPU AMR libraries :**
  - *PABLO : CPU-only library (MPI only)*
  - *RAMSES approach not suited for GPU (contrat de progrès - Idris)*
- **Performance challenges on GPU :**
  - *Data structure*
  - *Cell neighborhood access*
  - *CPU/GPU transfers : AMR tree, Physical fields, MPI communications, ...*

Solutions in Dyablo

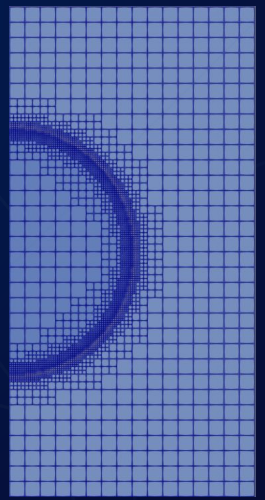
Block-based AMR:

- *Cartesian subgrids :*
  => More regular access patterns (great for GPU)
- *Reducing AMR tree depth :*
  => Reduce the AMR cycle cost

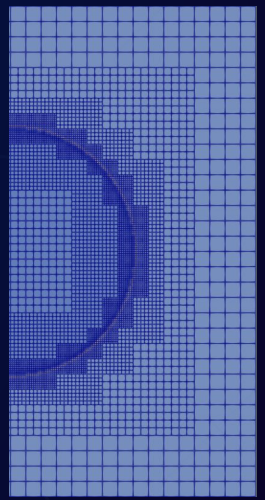
GPU 'light' AMR octree :

- *Paradigm change in PABLO vs RAMSES :*
  "Linear octree" (leaves only) => better for GPUs
- *Hashmap AMR :*
  - *Parallel, GPU compatible*
  - `Kokkos::UnorderedMap`

Cell-based
600k AMR octs
600k Cells



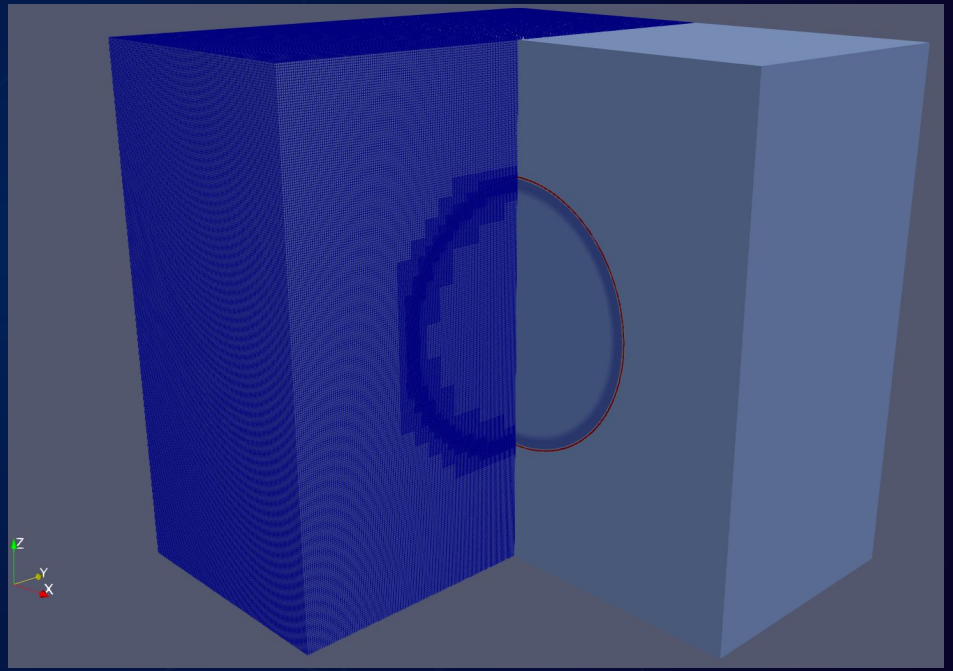
Block-based
18k AMR octs
1100k Cells



HYDRO BENCHMARK
Single-node performance

Testcase:

- 3D Sedov Blast
- MUSCL-Hancock
- **3 active AMR levels**
- 1024^3 cells at finest level
- **Time-to-solution t=5ms**
- Ramses :
  - NVECTOR = 32
  - nremap=10
  - nsubcycle=2*1,20*2
  - **AMR levels 8 -> 10**
- Dyablo
  - **Block-based**
  - AMR levels ~ Ramses (same cell size)
  - **Hybrid MPI+OpenMP or MPI + CUDA**



HYDRO BENCHMARK : CPU

Single-node performance

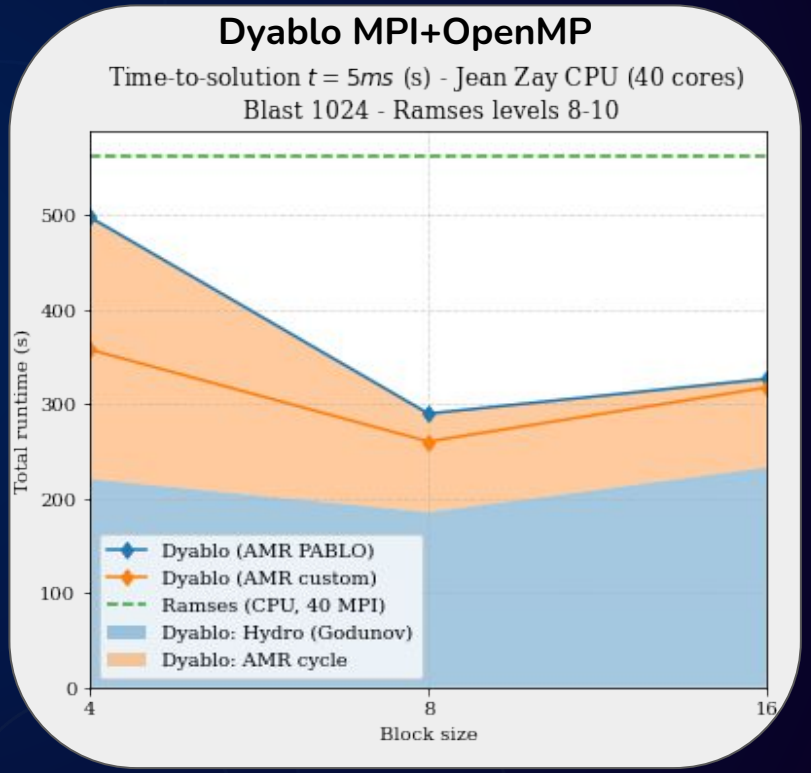
Ramses
1 Jean Zay node - 40 cores / 40 MPI

Block size	Godunov (s)	Total (s)
(1)	256.3	561.5

Dyablo (Custom AMR) MPI+OpenMP
1 Jean Zay node - 4MPI x 10cores

Block size	Godunov (s)	Total (s)
4	221	357
8	183	260
16	233	317

Dyablo MPI+OpenMP



HYDRO BENCHMARK : GPU

Single-node performance

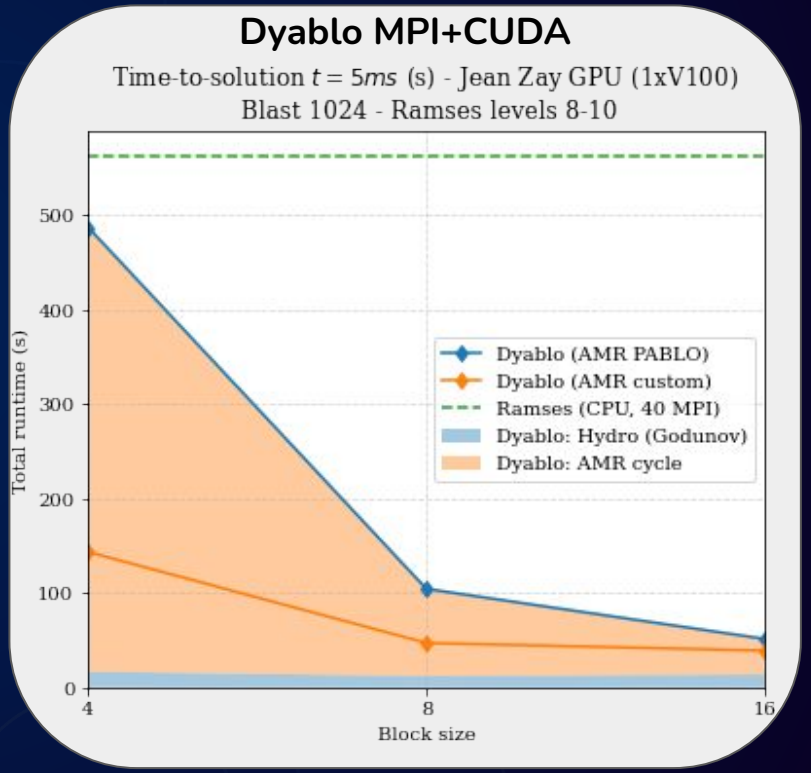
Ramses
1 Jean Zay node - 40 cores / 40 MPI

Block size	Godunov (s)	Total (s)
(1)	256.3	561.5

Dyablo (Custom AMR)
MPI+CUDA
Jean Zay - 1xV100

Block size	Godunov (s)	Total (s)
4	16.5	144
8	12.0	47.1
16	13.9	39.0

Dyablo MPI+CUDA

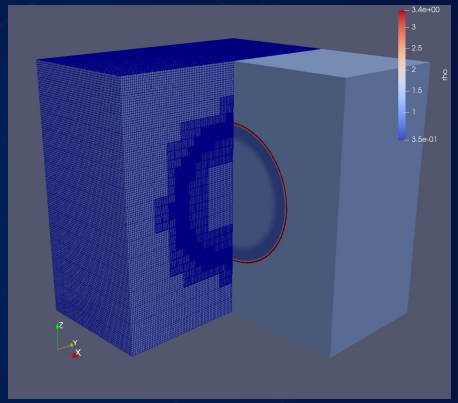


HYDRO BENCHMARK
MPI scalability

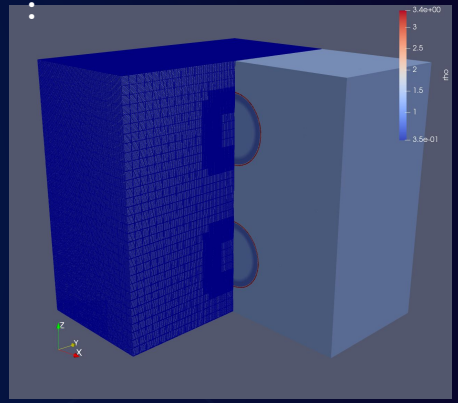
Cas test :

- Replicated 3D Sedov blast
- MUSCL-Hancock
- **4 active AMR levels**
- **512^3** at finest level **per MPI process**
- **256 itérations**
- Ramses :
  levels 5-9 for 1 blast
  levels 8-12 for 512 blasts
- Dyablo
  - Block-based (size = 8)
  - AMR levels ~ Ramses (same cell size)

1 MPI = 1x1x1 :



2 MPI = 1x1x2 :



Weak scaling :

- Resolution growing with hardware resources
  - 1x 512^3 blast per ¼ compute node
  - AMR tree depth grows with hardware resources
- Fixed number of timesteps
  => runtime should stay constant

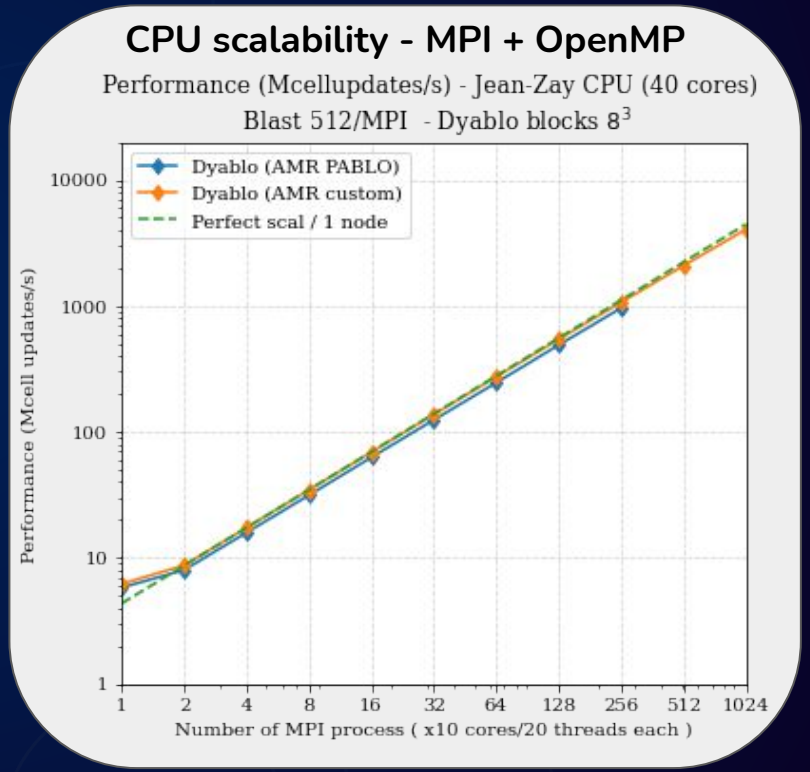
Caveats :

- Slight variations in number of cells/MPI
- Load-balanced by construction
- 4 active AMR levels :
  coarse level increases with resolution

HYDRO BENCHMARK : CPU

MPI scalability

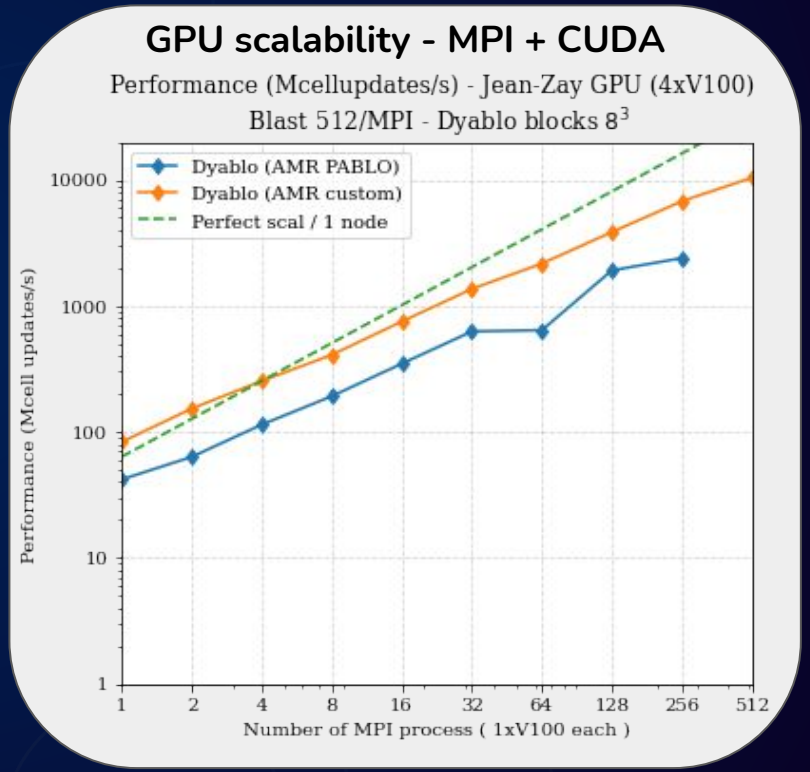
Nodes x cores	Hydro (s)	Total (s)	Mcell updates/s
1x10	424	579	6.2
1x20	579	579	8.7
1x40	570	781	17.4
2x40	568	779	34.6
4x40	579	796	69.0
8x40	571	790	137
16x40	569	787	274
32x40	579	809	543
64x40	570	813	1068
128x40	567	827	2085
256x40	579	868	4052



HYDRO BENCHMARK : GPU

MPI scalability

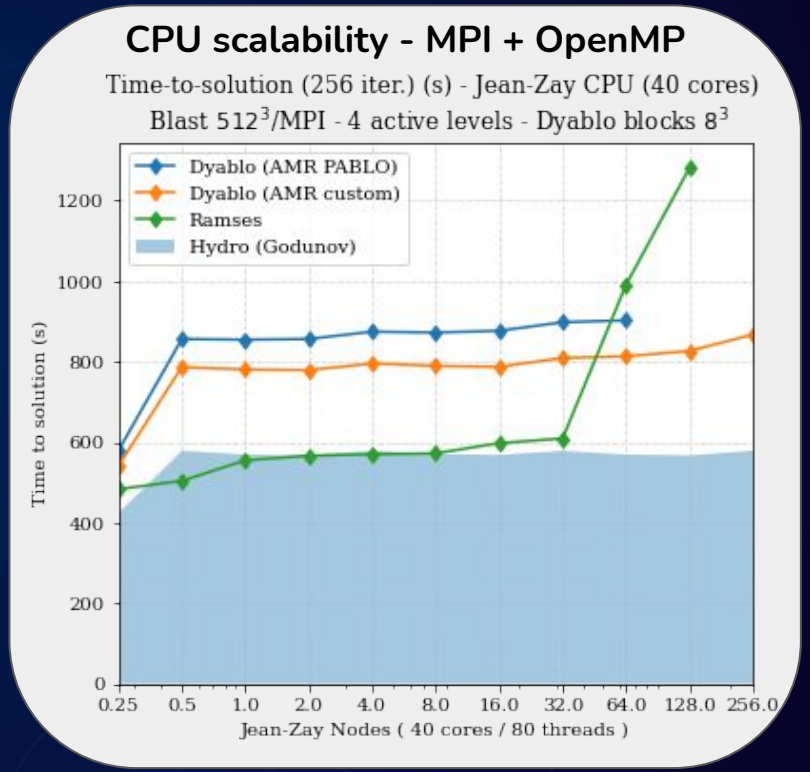
Nodes x GPUs	Hydro (s)	Total (s)	Mcell updates/s
1x1	8.98	40.5	83.1
1x2	8.98	44.9	153
1x4	8.79	53.5	253
2x4	8.72	66.0	408
4x4	8.90	73.1	751
8x4	8.79	79.6	1 364
16x4	8.72	79.6	2 168
32x4	8.90	114	3 860
64x4	8.87	128	6 773
128x4	8.81	147	10 411



HYDRO BENCHMARK : CPU

MPI scalability

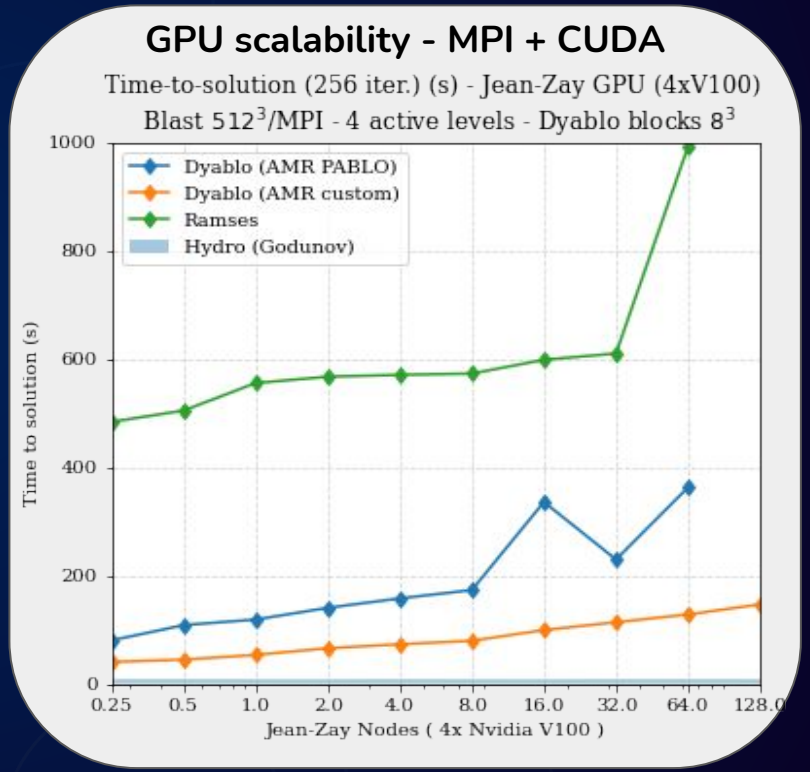
Nodes x cores	Hydro (s)	Total (s)	Mcell updates/s
1x10	424	579	6.2
1x20	579	579	8.7
1x40	570	781	17.4
2x40	568	779	34.6
4x40	579	796	69.0
8x40	571	790	137
16x40	569	787	274
32x40	579	809	543
64x40	570	813	1068
128x40	567	827	2085
256x40	579	868	4052



HYDRO BENCHMARK : GPU

MPI scalability

Nodes x GPUs	Hydro (s)	Total (s)	Mcell updates/s
1x1	8.98	40.5	83.1
1x2	8.98	44.9	153
1x4	8.79	53.5	253
2x4	8.72	66.0	408
4x4	8.90	73.1	751
8x4	8.79	79.6	1 364
16x4	8.72	79.6	2 168
32x4	8.90	114	3 860
64x4	8.87	128	6 773
128x4	8.81	147	10 411



CONCLUSION

Dyablo's philosophy :

- *Modern and modular*
- *CPU/GPU Compatibility (Kokkos)*
- *AMR tailored for GPU*

=> A solution for exascale and GPUs for the RAMSES community

Hydro demo :

- Hardware-agnostic hydro solver :
  - *AMR Block / Cell , 2D / 3D*
  - *Kokkos : OpenMP / CUDA*
- Performance :
  - *Competitive with RAMSES*
  - *Scalable : Tested up to **512 GPUs** and **10 000 CPU cores**.*

FUTURE WORK

Improve Dyablo :

- Self-gravity
  - *Conjugate Gradient (WIP)*
  - *Multigrid poisson solver*
- Particles
- Optimisations :
  - *Hydro solver (Vectorisation, multi-timestep, ...)*
  - *Optimize AMR cycle on GPU (2:1 balance, load-balancing)*

Collaborate with the community:

- **Ginea** : *Frequent discussions to set the priorities for Dyablo*
- **Whole-Sun** : *Convection benchmark for the surface of the sun (M. Delorme)*
- **You** : *Contact us if you want to try Dyablo.*

*mais restons prudents