Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

RAMSES GPU

Presented by:
Raffaele Pascale

Collaborators:
Francesco Calura, Claudio Gheller, Emanuele De Rubeis,
Donatella Romano, Valentina Cesare

Spoke 3 General Meeting, Elba 5-9 / 05, 2024

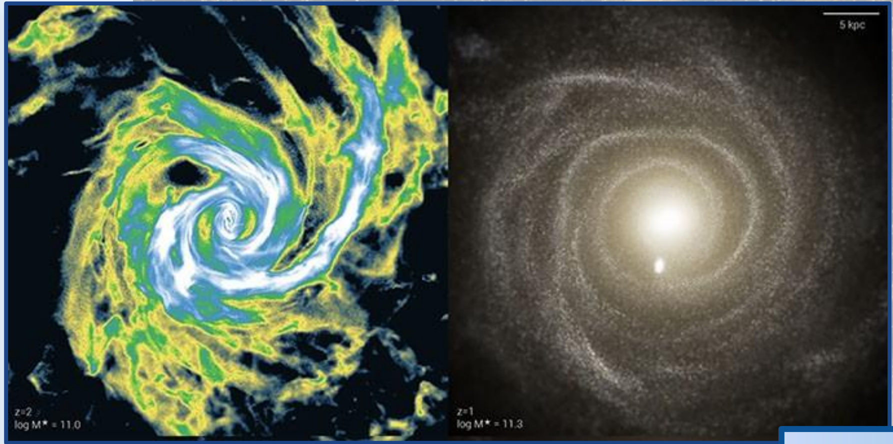
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Context

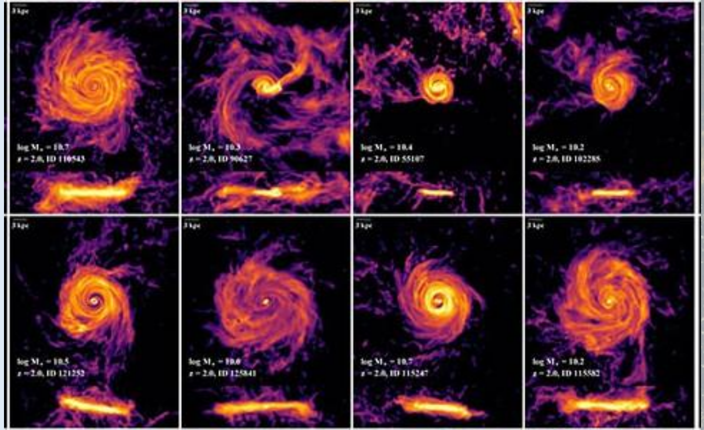
Hydrodynamical *N*-body simulations are **essential in astrophysics** since they provide tests for theories of galaxy formation and evolution.

High spatial resolutions are need to get a deeper understanding of galaxy physics.

Credits:
https://www.tng-project.org/media/



Challenges



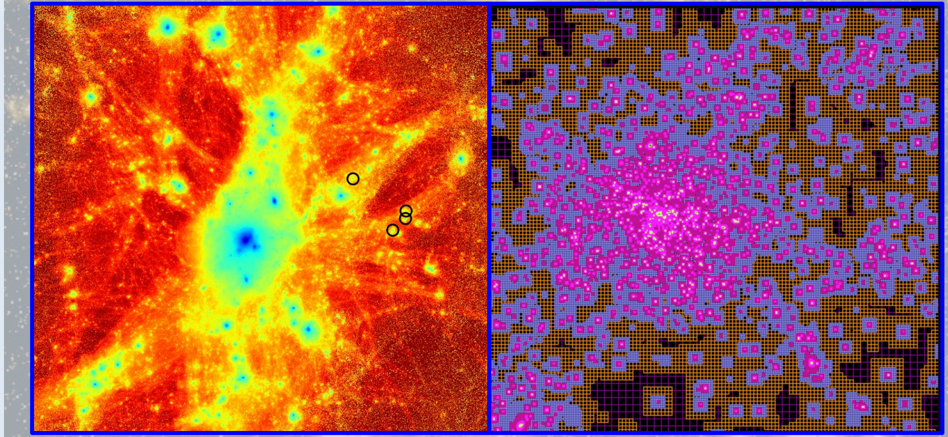
As spatial resolution increases, **computational demands escalate dramatically**

Addressing this challenge requires innovative solutions to optimize and accelerate computations.

An effective strategy involves porting hydrodynamical codes onto **GPU architecture (RAMSES)**

Application to RAMSES and MINIRAMSES

- **Ramses** and **Miniramses** are written in Fortran programming language
- **Eulerian** approach for solving compressible hydrodynamics equations
- Compatible with graphics processing units (GPUs)
- Implements adaptive mesh refinement **(AMR)** for resolving structures on different scales



MINIRAMSES is a novel version of Ramses with a more efficient grid memory management system that facilitates memory access and significantly improves the chances of an efficient GPU porting of the code

AMR

(Adaptive Mesh Refinement)

Identification of Oct Cell:

- It identifies an individual cell within the oct in the computational domain.

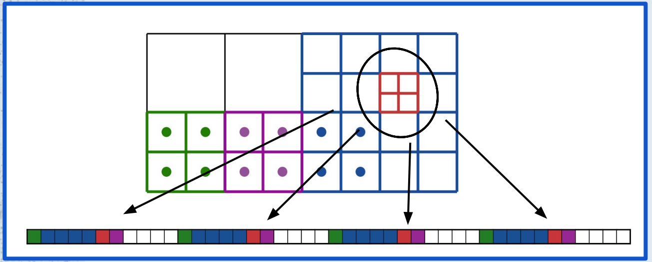
Refinement Evaluation:

- It assesses if the oct cell meets the criteria for refinement.
- Criteria may include gas density, density gradient, or other physical properties.

Cell Refinement:

- If the oct cell meets refinement criteria, it is divided into smaller cells.
- The process increases grid resolution in the region of interest.

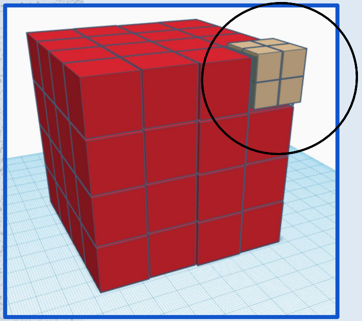
RAMSES



Example of classical AMR working

During cells refinement, new born cells belonging to the same oct are saved in non-contiguous parts of the memory.

MINIRAMSES



Introduces the new macrostructure: of super-oct in cell refinement.

ocs in super-octs are saved in contiguous memory locations. Cell adjacent in space close in memory

minimizes memory access

AMR

(Adaptive Mesh Refinement)

Identification of Oct Cell:

- It identifies an individual cell within the oct in the computational domain.

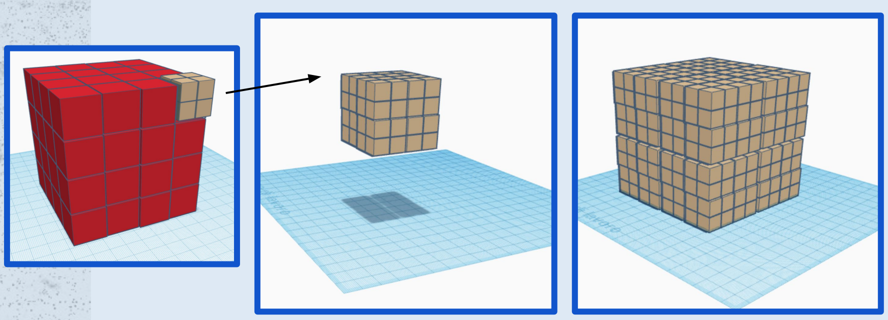
Refinement Evaluation:

- It assesses if the oct cell meets the criteria for refinement.
- Criteria may include gas density, density gradient, or other physical properties.

Cell Refinement:

- If the oct cell meets refinement criteria, it is divided into smaller cells.
- The process increases grid resolution in the region of interest.

super-oct



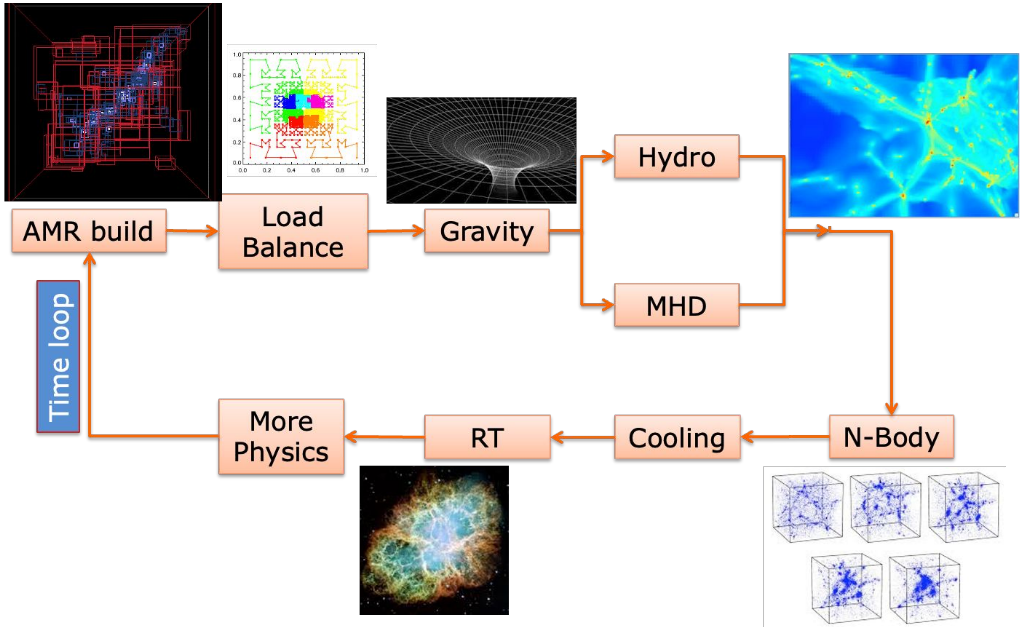
The superoct **ia a cube comprised of smaller cubes (octs)**. The superoct level operates akin to grid refinement, wherein each level increment represents a doubling factor of 2. The 'edge' of the superoct contains **twice as many octs as the previous level.**

superoct level (**n**) from 0 to 5.
In 3d, the number of octs per superoct is 8^n

n = 4 ---> octs per superoct = 4096
n = 5 ---> octs per superoct = 32768

The larger n, the better the changes for an optimal porting

Basic functioning of (MINI)RAMSES



AMR build → Load Balance → Gravity → Hydro / MHD → N-Body → Cooling → RT → More Physics → AMR build

Time loop

Adaptive Mesh Refinement (AMR):

the grid resolution is dynamically adapted to match the simulation's needs. Regions of interest are refined for higher resolution

Load Balancing:

RAMSES optimizes computational resources by distributing the workload evenly across processing units.

Gravity:

Gravity field is computed based on the matter distribution.

Hydro:

The hydrodynamic equations describing the fluid motion are solved

N-body:

the trajectories of collisionless particles (e.g., dark matter) are evolved using the leapfrog algorithm.

Cooling:

Cooling processes to account for energy loss

More physics:

Additional physics as wids, star formation etc.

Main goal

- Enhancing Efficiency and Decreasing Computational Time.
- Adapting components of MINIRAMSES for GPU architecture, resulting in a significant acceleration factor.

Adaptive Mesh Refinement (AMR):
the grid resolution is dynamically adapted to match the simulation's needs. Regions of interest are refined for higher resolution

Load Balancing:
RAMSES optimizes computational resources by distributing the workload evenly across processing units.

Gravity:
Gravity field is computed based on the matter distribution.

Hydro:
The hydrodynamic equations describing the fluid motion are solved

N-body:
the trajectories of collisionless particles (e.g., dark matter) are evolved using the leapfrog algorithm.

Cooling:
Cooling processes to account for energy loss

More physics:
Additional physics as wids, star formation etc.

Main goal

- Enhancing Efficiency and Decreasing Computational Time.
- Adapting components of MINIRAMSES for GPU architecture, resulting in a significant acceleration factor.

What and how

- Identification of two main parts of the code suitable for GPU porting: ***N*-body + Hydro**
- **OpenACC** directives to parallelize time-consuming loops and critical code regions;
- Optimization techniques for memory management, and data movement

Adaptive Mesh Refinement (AMR):

the grid resolution is dynamically adapted to match the simulation's needs. Regions of interest are refined for higher resolution

Load Balancing:

RAMSES optimizes computational resources by distributing the workload evenly across processing units.

Gravity:

Gravity field is computed based on the matter distribution.

Hydro:

The hydrodynamic equations describing the fluid motion are solved

N-body:

the trajectories of collisionless particles (e.g., dark matter) are evolved using the leapfrog algorithm.

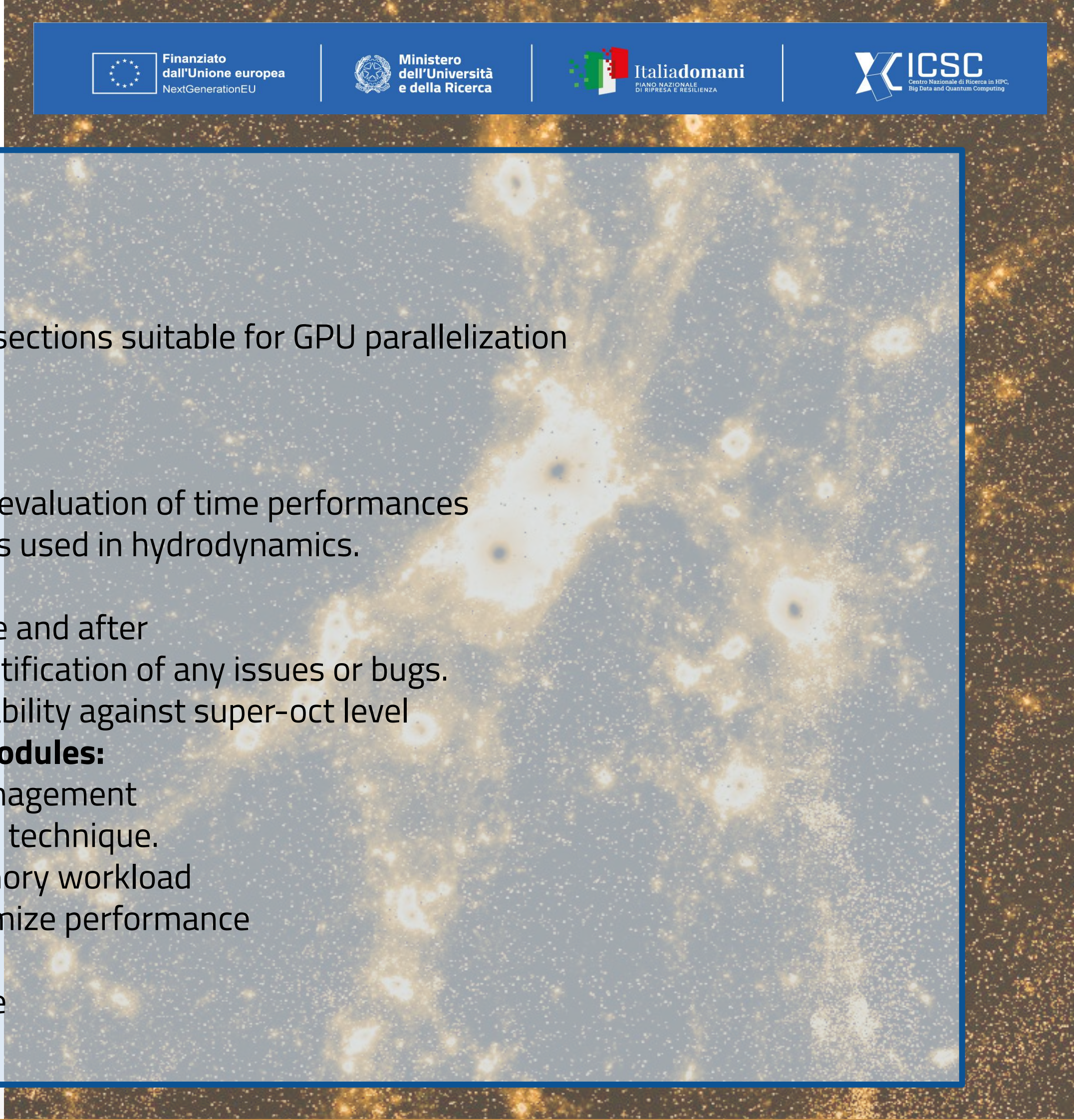
Cooling:

Cooling processes to account for energy loss

More physics:

Additional physics as wids, star formation etc.

Timescale and milestones



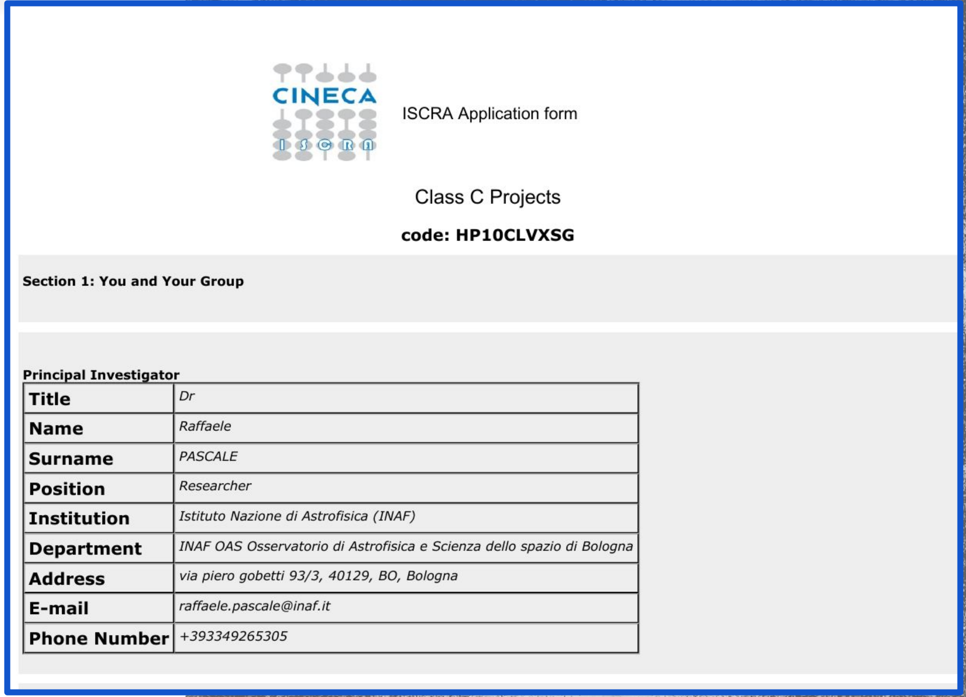
1. **M6 - Preliminary analysis:**
   Investigation of MINIRAMSES to identify sections suitable for GPU parallelization
2. **M7 - Getting GPU resources:**
   Submission o proposal @Cineca
3. **M8 - GPU porting of Hydro modules:**
   Identification of modules to port on GPU, evaluation of time performances
   Gradual GPU porting of individual modules used in hydrodynamics.
4. **M9 - Tests**
   Tests and performance evaluations before and after
   Evaluation of initial performance and identification of any issues or bugs.
   Implementation of tests to evaluate scalability against super-oct level
5. **M10 - Memory management of hydro modules:**
   Identification of strategy for memory management
   Implementation of memory management technique.
   Implementation of tests to evaluate memory workload
   Optimization of the code on GPU to maximize performance
6. **M11? - Integration:**
   Integration in principal version of the code
   Execution of tests to evaluate scalability

Accomplished work and results

- **Got GPU hours** on Leonardo with an accepted ISCRA C proposal
- Identification of modules hydrodynamical modules to port on GPU.
- **Made the code work on Leonardo** (took few months)

During the resolution of code compilation issues, with the support @ Cineca and @NVIDIA, we concluded that **offloading the Nbody component to the GPU is currently not feasible**.

The Nbody modules rely on a c_f_pointer function, a Fortran intrinsic procedure used for interoperability with C/C++ code. This function facilitates the exchange of data between Fortran and other languages by providing a Fortran pointer from a C pointer or vice versa. However, this functionality is not available for GPU offloading



ISCRA Application form

Class C Projects

code: HP10CLVXSG

Section 1: You and Your Group

Principal Investigator

Title	*Dr*
Name	*Raffaele*
Surname	*PASCALE*
Position	*Researcher*
Institution	*Istituto Nazione di Astrofisica (INAF)*
Department	*INAF OAS Osservatorio di Astrofisica e Scienza dello spazio di Bologna*
Address	*via piero gobetti 93/3, 40129, BO, Bologna*
E-mail	*raffaele.pascale@inaf.it*
Phone Number	*+393349265305*

Hydrodynamic solver

- The **Godunov** solver is a numerical technique for solving hyperbolic PDEs describing fluid flow.
- **Domain Discretization:** The spatial domain undergoes discretization into cells, constituting a 3D grid.
- **Flux Calculation Across Cell Boundaries:** For each cell, the Godunov method computes fluxes across its borders, considering fluid properties and boundary conditions.
- **State Variable Update:** State variables of the fluid get updated based on computed fluxes, adhering to flow conservation equations.
- **Temporal Iteration:** The entire process iterates over each time step until reaching a defined stopping criterion.

Hydrodynamic solver

The **Godunov** solver is a numerical t
solving hyperbolic PDEs describing fluid f

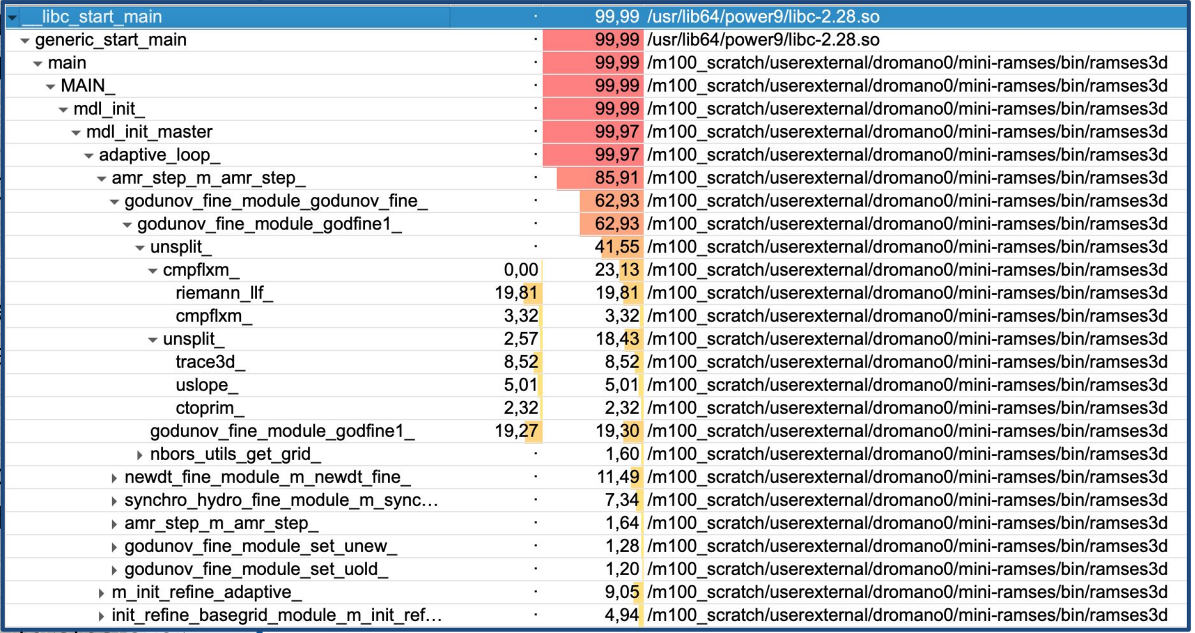
Domain Discretization: The spatial dom
discretization into cells, constituting a 3D

Flux Calculation Across Cell Boundaries
the Godunov method computes fluxe
borders, considering fluid properties a
conditions.

State Variable Update: State variables c
updated based on computed fluxes, adl
conservation equations.

Temporal Iteration: The entire process iterates over each time step until reaching a defined stopping criterion.

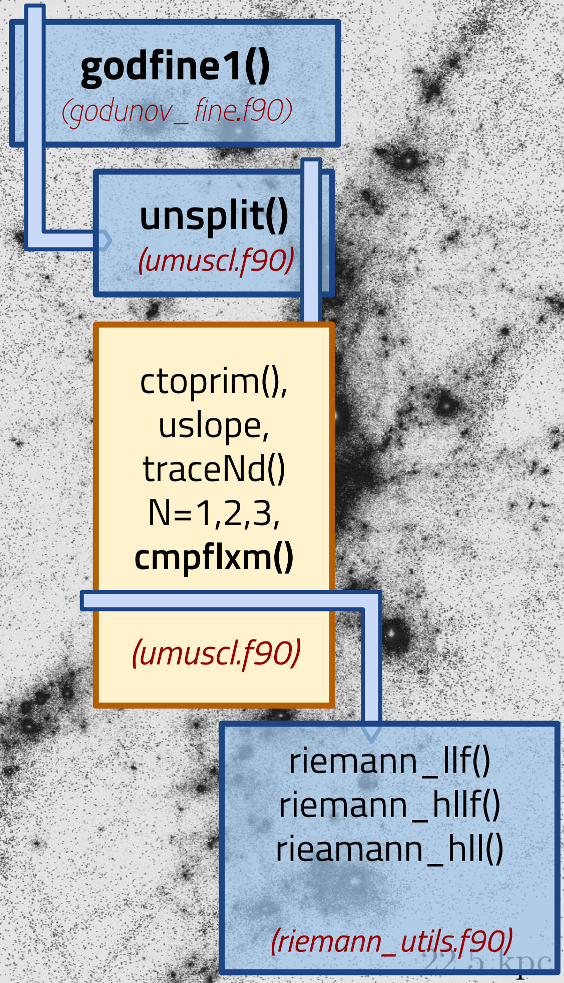
run over 1 CPU



```
▾ __libc_start_main                                  ·   99,99 /usr/lib64/power9/libc-2.28.so
 ▾ generic_start_main                                ·   99,99 /usr/lib64/power9/libc-2.28.so
  ▾ main                                             ·   99,99 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
   ▾ MAIN_                                           ·   99,99 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
    ▾ mdl_init_                                      ·   99,99 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
     ▾ mdl_init_master                               ·   99,97 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
      ▾ adaptive_loop_                               ·   99,97 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
       ▾ amr_step_m_amr_step_                        ·   85,91 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
        ▾ godunov_fine_module_godunov_fine_          ·   62,93 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
         ▾ godunov_fine_module_godfine1_             ·   62,93 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
          ▾ unsplit_                                 ·   41,55 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
           ▾ cmpflxm_                             0,00   23,13 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
              riemann_llf_                       19,81   19,81 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
              cmpflxm_                            3,32    3,32 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
           ▾ unsplit_                             2,57   18,43 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
              trace3d_                            8,52    8,52 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
              uslope_                             5,01    5,01 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
              ctoprim_                            2,32    2,32 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
          godunov_fine_module_godfine1_          19,27   19,30 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
          ▸ nbors_utils_get_grid_                    ·    1,60 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
        ▸ newdt_fine_module_m_newdt_fine_            ·   11,49 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
        ▸ synchro_hydro_fine_module_m_sync…          ·    7,34 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
        ▸ amr_step_m_amr_step_                       ·    1,64 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
        ▸ godunov_fine_module_set_unew_              ·    1,28 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
        ▸ godunov_fine_module_set_uold_              ·    1,20 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
       ▸ m_init_refine_adaptive_                     ·    9,05 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
       ▸ init_refine_basegrid_module_m_init_ref…     ·    4,94 /m100_scratch/userexternal/dromano0/mini-ramses/bin/ramses3d
```

63% of the time is spent by the hydrodynamical solver (godfine1)

Scheme of subroutines involved in the GPU hydro-porting



godfine1()
(godunov_fine.f90)

unsplit()
(umuscl.f90)

ctoprim(),
uslope,
traceNd()
N=1,2,3,
cmpflxm()

(umuscl.f90)

riemann_llf()
riemann_hllf()
rieamann_hll()

(riemann_utils.f90)

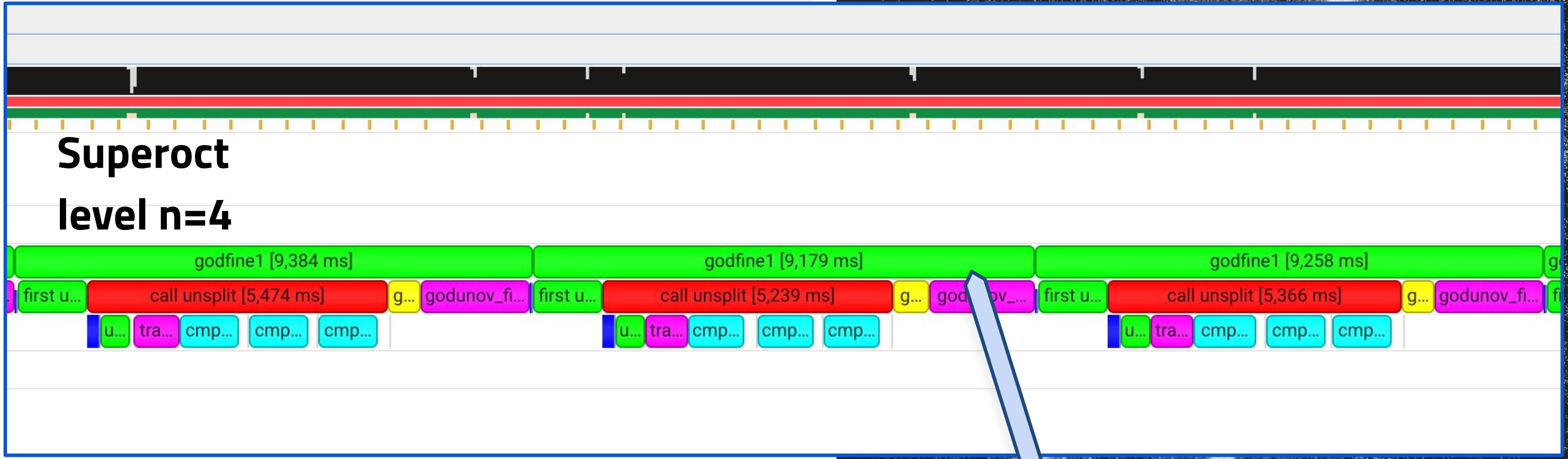
ctoprim(): The "ctoprim" subroutine converts conservative variables to primitive variables,density, momentum, energy) into primitive variables (density, velocity, pressure) .

uslope(): It is executed to compute gradients of primitive variables within each cell, providing information on slopes along cell edges.

traceNd(): the subroutine computes fluxes across cell boundaries in all directions of the domain, utilizing previously calculated gradients.

cmpflxm(): it calculates fluxes across cell boundaries based on primitive variables and cell interfaces, completing the flux calculation necessary for updating the fluid state variables.

Accomplished work



Full CPU run (1CPU)
Sedov3d test: Explosion of a supernovae in a constant medium.
Only hydro, no gravity.

Major of the computational time is spent during the **call unsplit()**

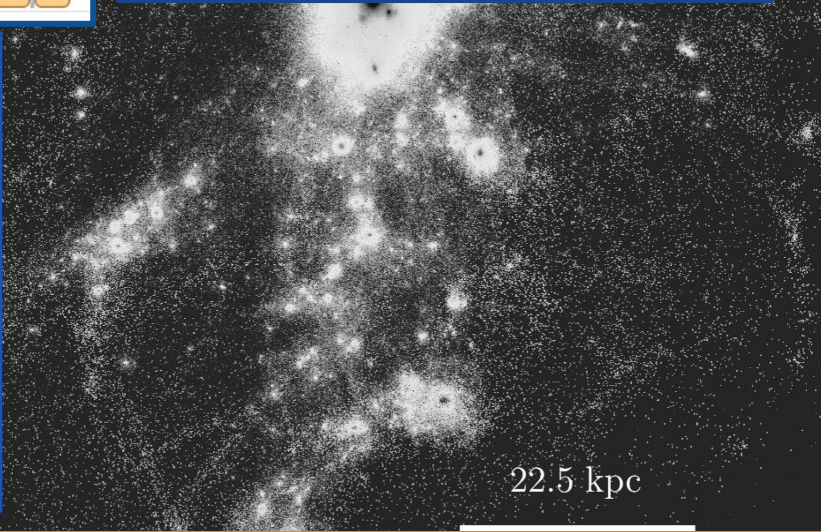
Each call to godfine1 solves hydrodynamics for one super-oct

22.5 kpc

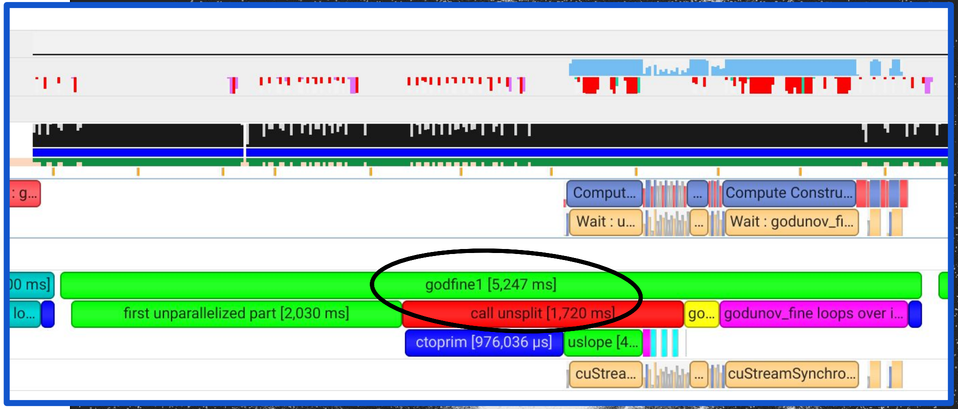
Porting the initial loops does not yield any significant speedup of the code.

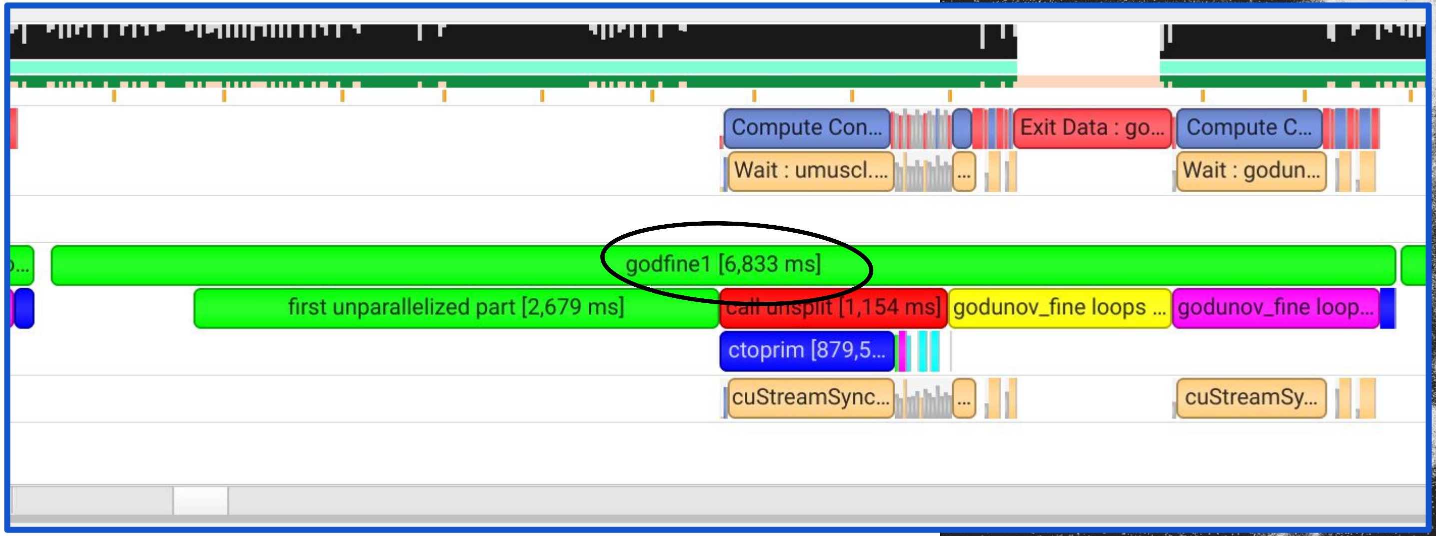
The speedup of individual parts is not notable.

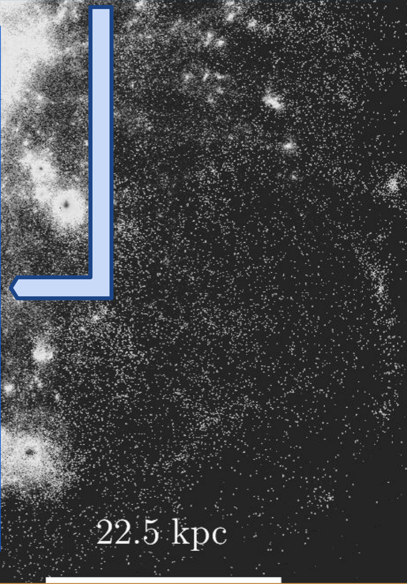
The bottleneck shifts to calls to previous subroutines.




Compute Constru...
Comput...
Wait : umuscl.f90:...
Wait : go...
godfine1 [8,393 ms]
first unparallelized part [2,060 ...
call unsplit [4,818 ms]
g...
godunov_...
godu...
firs
ctoprim [1,01...
uslope [2,247 ms]
traceNd [1,328 ms]
cuStreamSynchro...
cuStrea...

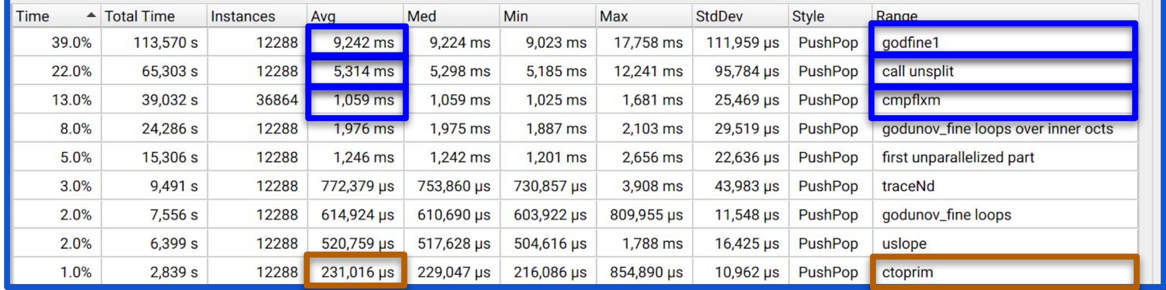

: g...
Comput...
Compute Constru...
Wait : u...
Wait : godunov_fi...
godfine1 [5,247 ms]
first unparallelized part [2,030 ms]
call unsplit [1,720 ms]
go...
godunov_fine loops over i...
ctoprim [976,036 µs]
uslope [4...
cuStrea...
cuStreamSynchro...


Compute Con...
Exit Data : go...
Compute C...
Wait : umuscl....
Wait : godun...
godfine1 [6,833 ms]
first unparallelized part [2,679 ms]
call unsplit [1,154 ms]
godunov_fine loops ...
godunov_fine loop...
ctoprim [879,5...
cuStreamSync...
cuStreamSy...


22.5 kpc

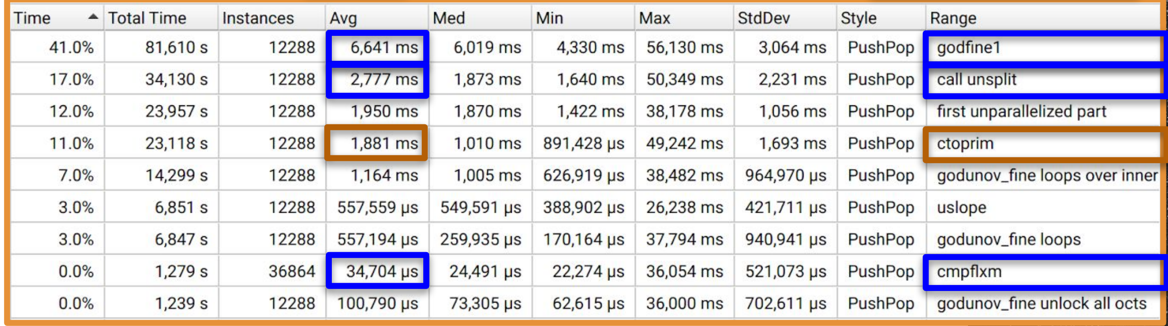
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

About time



full CPU

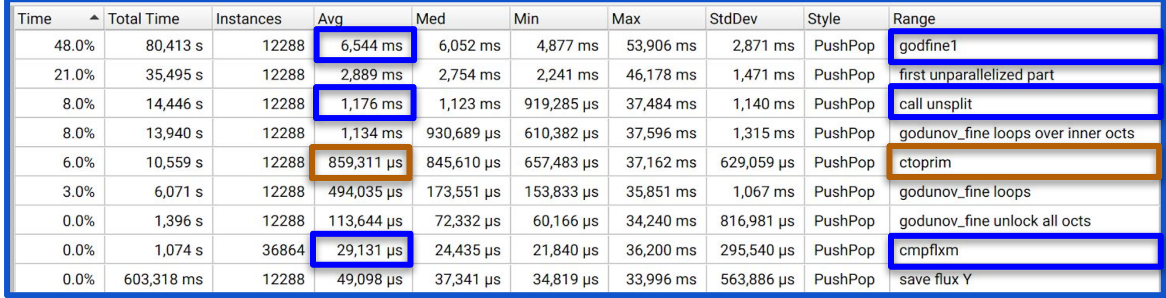
Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
39.0%	113,570 s	12288	9,242 ms	9,224 ms	9,023 ms	17,758 ms	111,959 µs	PushPop	godfine1
22.0%	65,303 s	12288	5,314 ms	5,298 ms	5,185 ms	12,241 ms	95,784 µs	PushPop	call unsplit
13.0%	39,032 s	36864	1,059 ms	1,059 ms	1,025 ms	1,681 ms	25,469 µs	PushPop	cmpflxm
8.0%	24,286 s	12288	1,976 ms	1,975 ms	1,887 ms	2,103 ms	29,519 µs	PushPop	godunov_fine loops over inner octs
5.0%	15,306 s	12288	1,246 ms	1,242 ms	1,201 ms	2,656 ms	22,636 µs	PushPop	first unparallelized part
3.0%	9,491 s	12288	772,379 µs	753,860 µs	730,857 µs	3,908 ms	43,983 µs	PushPop	traceNd
2.0%	7,556 s	12288	614,924 µs	610,690 µs	603,922 µs	809,955 µs	11,548 µs	PushPop	godunov_fine loops
2.0%	6,399 s	12288	520,759 µs	517,628 µs	504,616 µs	1,788 ms	16,425 µs	PushPop	uslope
1.0%	2,839 s	12288	231,016 µs	229,047 µs	216,086 µs	854,890 µs	10,962 µs	PushPop	ctoprim



intermediate

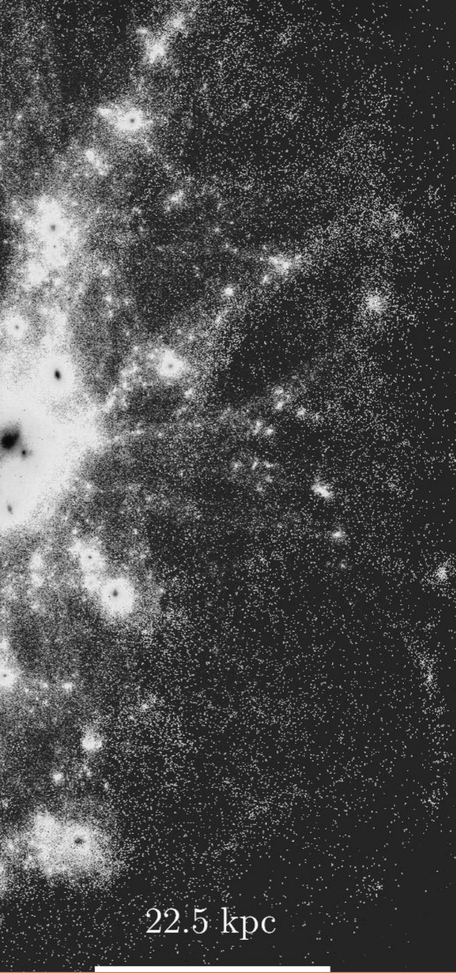
Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
41.0%	81,610 s	12288	6,641 ms	6,019 ms	4,330 ms	56,130 ms	3,064 ms	PushPop	godfine1
17.0%	34,130 s	12288	2,777 ms	1,873 ms	1,640 ms	50,349 ms	2,231 ms	PushPop	call unsplit
12.0%	23,957 s	12288	1,950 ms	1,870 ms	1,422 ms	38,178 ms	1,056 ms	PushPop	first unparallelized part
11.0%	23,118 s	12288	1,881 ms	1,010 ms	891,428 µs	49,242 ms	1,693 ms	PushPop	ctoprim
7.0%	14,299 s	12288	1,164 ms	1,005 ms	626,919 µs	38,482 ms	964,970 µs	PushPop	godunov_fine loops over inner
3.0%	6,851 s	12288	557,559 µs	549,591 µs	388,902 µs	26,238 ms	421,711 µs	PushPop	uslope
3.0%	6,847 s	12288	557,194 µs	259,935 µs	170,164 µs	37,794 ms	940,941 µs	PushPop	godunov_fine loops
0.0%	1,279 s	36864	34,704 µs	24,491 µs	22,274 µs	36,054 ms	521,073 µs	PushPop	cmpflxm
0.0%	1,239 s	12288	100,790 µs	73,305 µs	62,615 µs	36,000 ms	702,611 µs	PushPop	godunov_fine unlock all octs

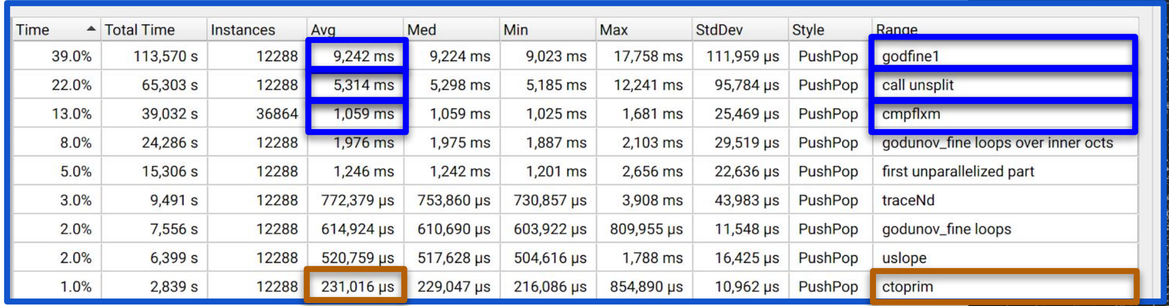
superoct level 4



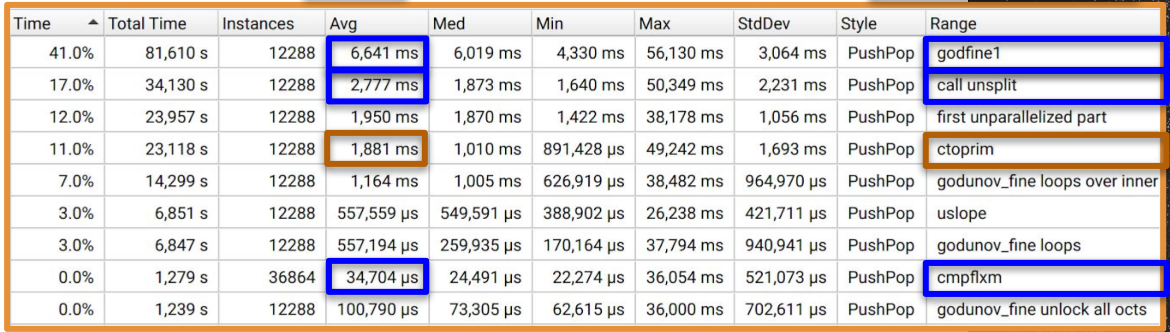
full GPU

Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
48.0%	80,413 s	12288	6,544 ms	6,052 ms	4,877 ms	53,906 ms	2,871 ms	PushPop	godfine1
21.0%	35,495 s	12288	2,889 ms	2,754 ms	2,241 ms	46,178 ms	1,471 ms	PushPop	first unparallelized part
8.0%	14,446 s	12288	1,176 ms	1,123 ms	919,285 µs	37,484 ms	1,140 ms	PushPop	call unsplit
8.0%	13,940 s	12288	1,134 ms	930,689 µs	610,382 µs	37,596 ms	1,315 ms	PushPop	godunov_fine loops over inner octs
6.0%	10,559 s	12288	859,311 µs	845,610 µs	657,483 µs	37,162 ms	629,059 µs	PushPop	ctoprim
3.0%	6,071 s	12288	494,035 µs	173,551 µs	153,833 µs	35,851 ms	1,067 ms	PushPop	godunov_fine loops
0.0%	1,396 s	12288	113,644 µs	72,332 µs	60,166 µs	34,240 ms	816,981 µs	PushPop	godunov_fine unlock all octs
0.0%	1,074 s	36864	29,131 µs	24,435 µs	21,840 µs	36,200 ms	295,540 µs	PushPop	cmpflxm
0.0%	603,318 ms	12288	49,098 µs	37,341 µs	34,819 µs	33,996 ms	563,886 µs	PushPop	save flux Y

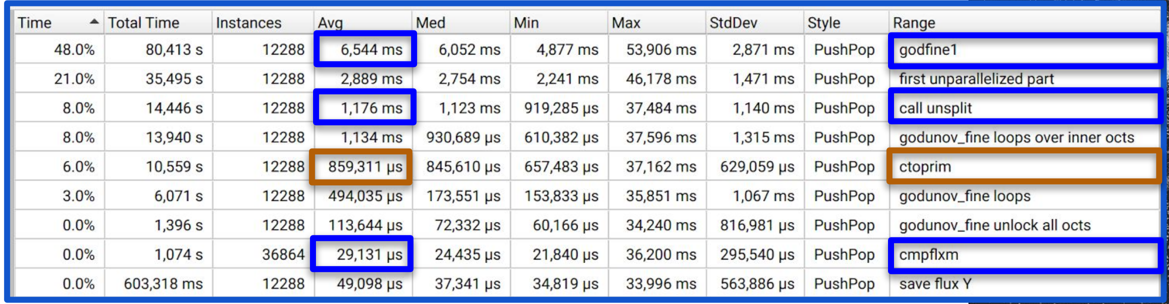




Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
39.0%	113,570 s	12288	9,242 ms	9,224 ms	9,023 ms	17,758 ms	111,959 µs	PushPop	godfine1
22.0%	65,303 s	12288	5,314 ms	5,298 ms	5,185 ms	12,241 ms	95,784 µs	PushPop	call unsplit
13.0%	39,032 s	36864	1,059 ms	1,059 ms	1,025 ms	1,681 ms	25,469 µs	PushPop	cmpflxm
8.0%	24,286 s	12288	1,976 ms	1,975 ms	1,887 ms	2,103 ms	29,519 µs	PushPop	godunov_fine loops over inner octs
5.0%	15,306 s	12288	1,246 ms	1,242 ms	1,201 ms	2,656 ms	22,636 µs	PushPop	first unparallelized part
3.0%	9,491 s	12288	772,379 µs	753,860 µs	730,857 µs	3,908 ms	43,983 µs	PushPop	traceNd
2.0%	7,556 s	12288	614,924 µs	610,690 µs	603,922 µs	809,955 µs	11,548 µs	PushPop	godunov_fine loops
2.0%	6,399 s	12288	520,759 µs	517,628 µs	504,616 µs	1,788 ms	16,425 µs	PushPop	uslope
1.0%	2,839 s	12288	231,016 µs	229,047 µs	216,086 µs	854,890 µs	10,962 µs	PushPop	ctoprim



Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
41.0%	81,610 s	12288	6,641 ms	6,019 ms	4,330 ms	56,130 ms	3,064 ms	PushPop	godfine1
17.0%	34,130 s	12288	2,777 ms	1,873 ms	1,640 ms	50,349 ms	2,231 ms	PushPop	call unsplit
12.0%	23,957 s	12288	1,950 ms	1,870 ms	1,422 ms	38,178 ms	1,056 ms	PushPop	first unparallelized part
11.0%	23,118 s	12288	1,881 ms	1,010 ms	891,428 µs	49,242 ms	1,693 ms	PushPop	ctoprim
7.0%	14,299 s	12288	1,164 ms	1,005 ms	626,919 µs	38,482 ms	964,970 µs	PushPop	godunov_fine loops over inner
3.0%	6,851 s	12288	557,559 µs	549,591 µs	388,902 µs	26,238 ms	421,711 µs	PushPop	uslope
3.0%	6,847 s	12288	557,194 µs	259,935 µs	170,164 µs	37,794 ms	940,941 µs	PushPop	godunov_fine loops
0.0%	1,279 s	36864	34,704 µs	24,491 µs	22,274 µs	36,054 ms	521,073 µs	PushPop	cmpflxm
0.0%	1,239 s	12288	100,790 µs	73,305 µs	62,615 µs	36,000 ms	702,611 µs	PushPop	godunov_fine unlock all octs



Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
48.0%	80,413 s	12288	6,544 ms	6,052 ms	4,877 ms	53,906 ms	2,871 ms	PushPop	godfine1
21.0%	35,495 s	12288	2,889 ms	2,754 ms	2,241 ms	46,178 ms	1,471 ms	PushPop	first unparallelized part
8.0%	14,446 s	12288	1,176 ms	1,123 ms	919,285 µs	37,484 ms	1,140 ms	PushPop	call unsplit
8.0%	13,940 s	12288	1,134 ms	930,689 µs	610,382 µs	37,596 ms	1,315 ms	PushPop	godunov_fine loops over inner octs
6.0%	10,559 s	12288	859,311 µs	845,610 µs	657,483 µs	37,162 ms	629,059 µs	PushPop	ctoprim
3.0%	6,071 s	12288	494,035 µs	173,551 µs	153,833 µs	35,851 ms	1,067 ms	PushPop	godunov_fine loops
0.0%	1,396 s	12288	113,644 µs	72,332 µs	60,166 µs	34,240 ms	816,981 µs	PushPop	godunov_fine unlock all octs
0.0%	1,074 s	36864	29,131 µs	24,435 µs	21,840 µs	36,200 ms	295,540 µs	PushPop	cmpflxm
0.0%	603,318 ms	12288	49,098 µs	37,341 µs	34,819 µs	33,996 ms	563,886 µs	PushPop	save flux Y

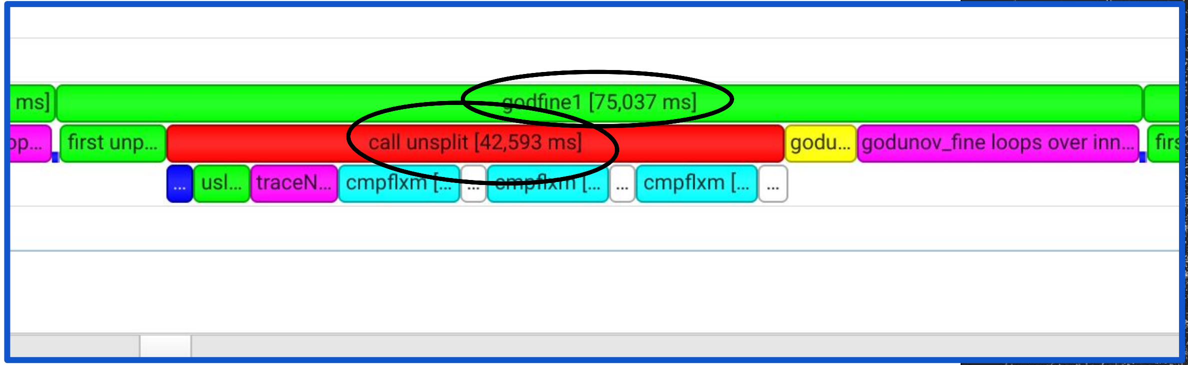
Improvements?

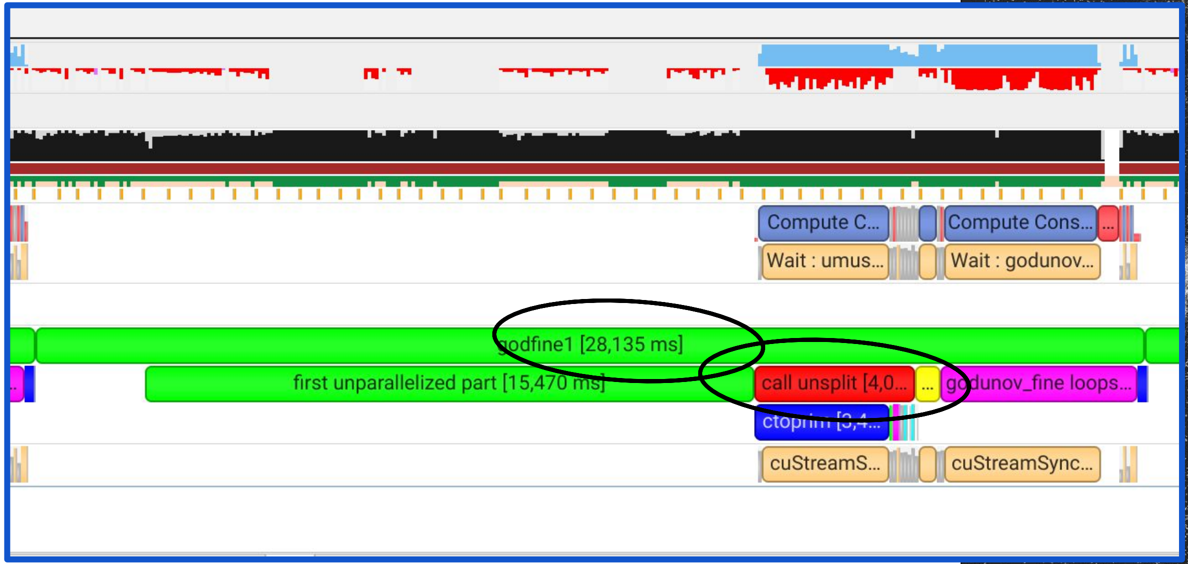
Each call to the godfine1 subroutine results in a speedup of approximately **1.5 times (low).**

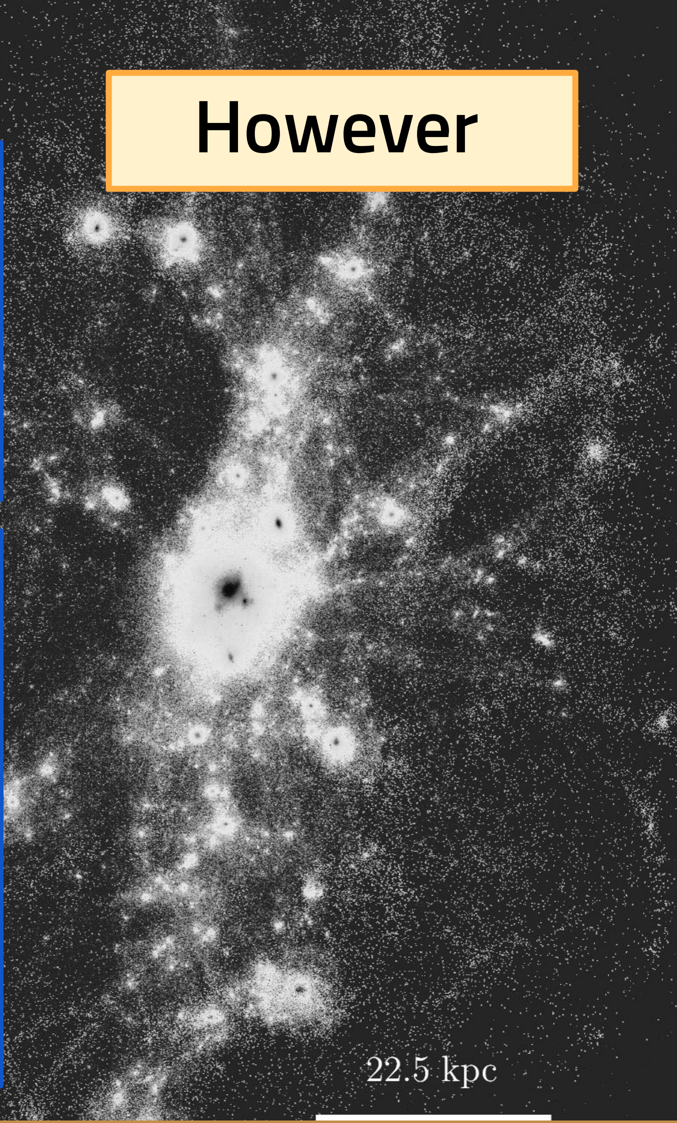
The primary reason for the limited gain is the **overhead associated with memory management and communication** between the CPU and GPU.

These tasks consume a significant portion of the processing time, offsetting the potential performance improvements.

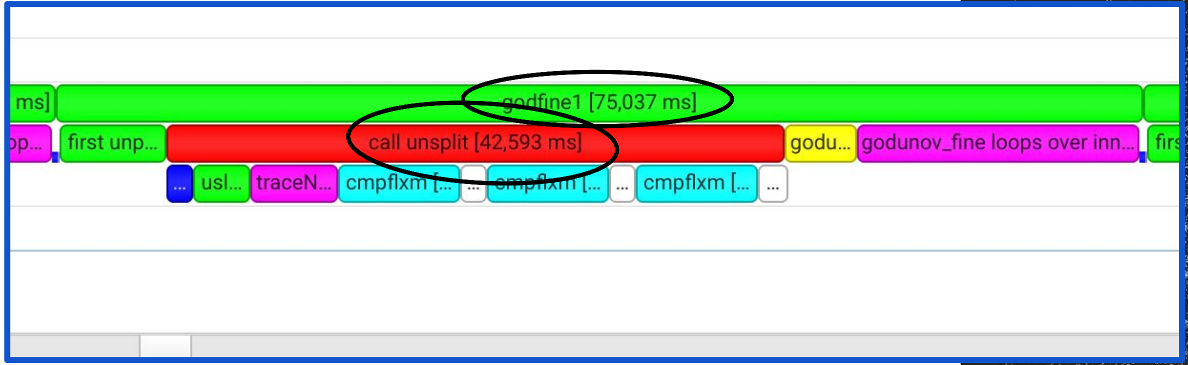
However

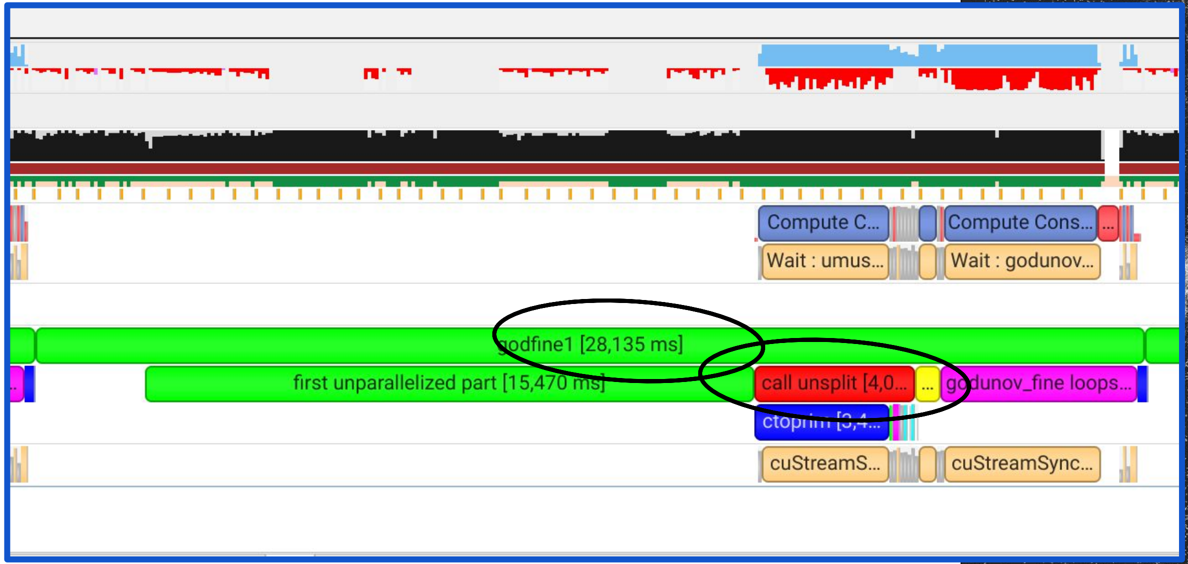

ms]
godfine1 [75,037 ms]
op...
first unp...
call unsplit [42,593 ms]
godu...
godunov_fine loops over inn...
firs
...
usl...
traceN...
cmpflxm [...
...
cmpflxm [...
...
cmpflxm [...
...


Compute C...
Compute Cons...
...
Wait : umus...
Wait : godunov...
godfine1 [28,135 ms]
first unparallelized part [15,470 ms]
call unsplit [4,0...
...
godunov_fine loops...
ctoprim [3,4...
cuStreamS...
cuStreamSync...

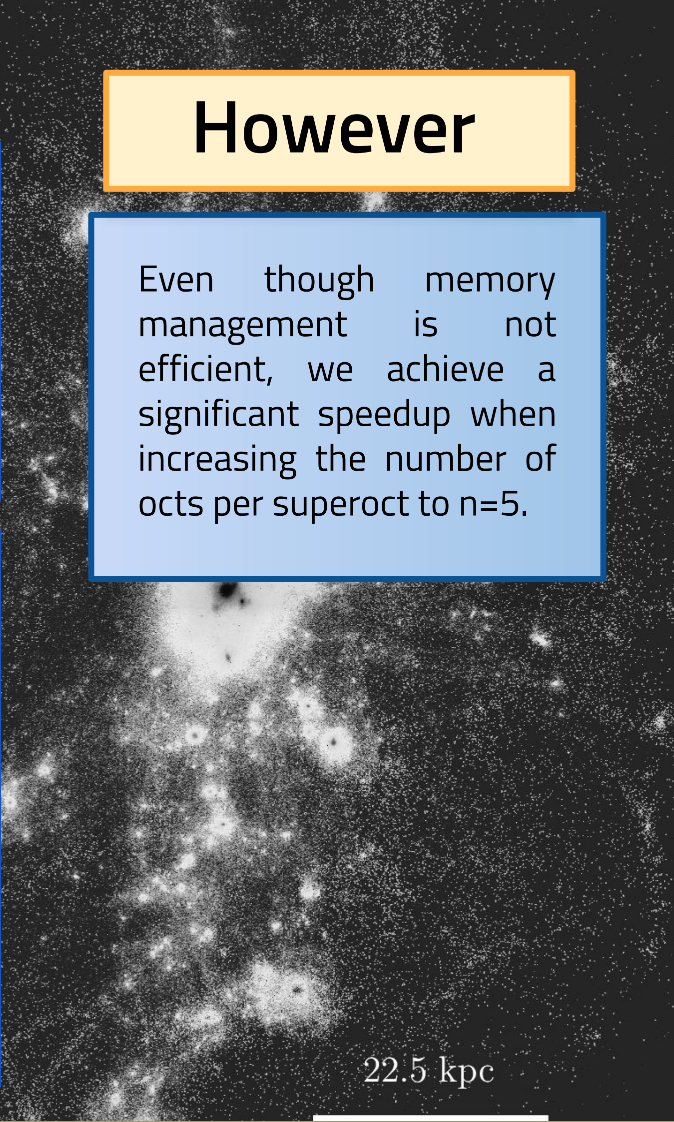

22.5 kpc

However



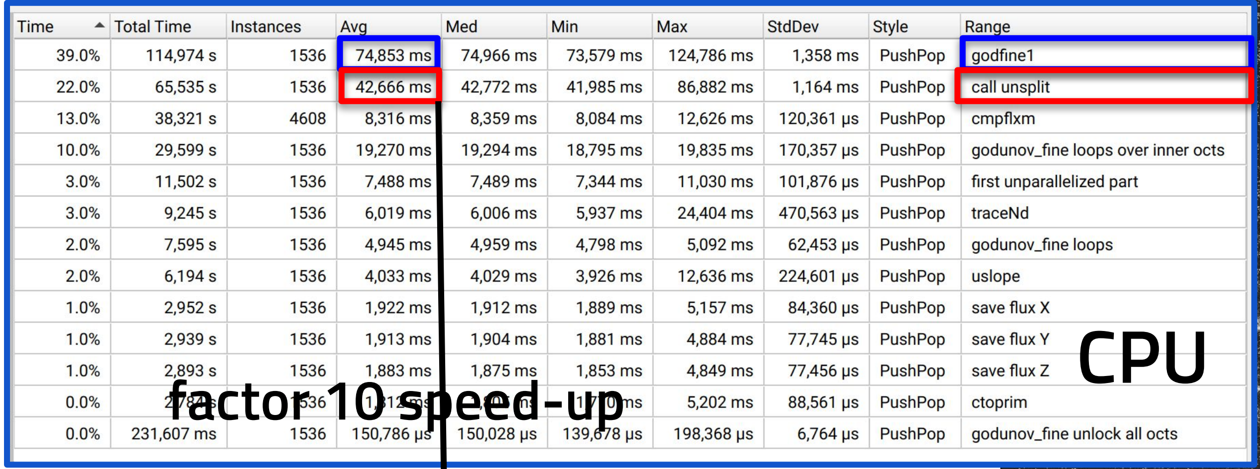


Even though memory management is not efficient, we achieve a significant speedup when increasing the number of octs per superoct to n=5.



However

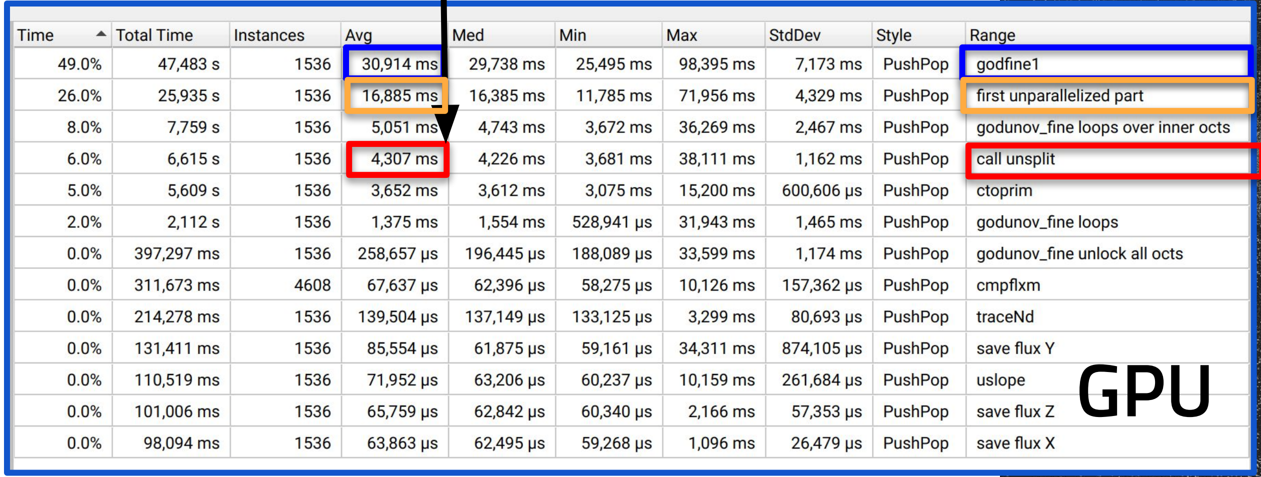
Even though memory management is not efficient, we achieve a significant speedup when increasing the number of octs per superoct to n=5.



CPU

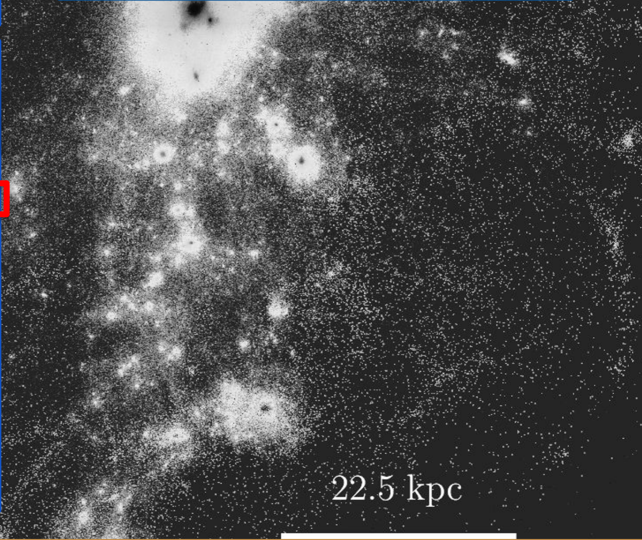
Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
39.0%	114,974 s	1536	74,853 ms	74,966 ms	73,579 ms	124,786 ms	1,358 ms	PushPop	godfine1
22.0%	65,535 s	1536	42,666 ms	42,772 ms	41,985 ms	86,882 ms	1,164 ms	PushPop	call unsplit
13.0%	38,321 s	4608	8,316 ms	8,359 ms	8,084 ms	12,626 ms	120,361 μs	PushPop	cmpflxm
10.0%	29,599 s	1536	19,270 ms	19,294 ms	18,795 ms	19,835 ms	170,357 μs	PushPop	godunov_fine loops over inner octs
3.0%	11,502 s	1536	7,488 ms	7,489 ms	7,344 ms	11,030 ms	101,876 μs	PushPop	first unparallelized part
3.0%	9,245 s	1536	6,019 ms	6,006 ms	5,937 ms	24,404 ms	470,563 μs	PushPop	traceNd
2.0%	7,595 s	1536	4,945 ms	4,959 ms	4,798 ms	5,092 ms	62,453 μs	PushPop	godunov_fine loops
2.0%	6,194 s	1536	4,033 ms	4,029 ms	3,926 ms	12,636 ms	224,601 μs	PushPop	uslope
1.0%	2,952 s	1536	1,922 ms	1,912 ms	1,889 ms	5,157 ms	84,360 μs	PushPop	save flux X
1.0%	2,939 s	1536	1,913 ms	1,904 ms	1,881 ms	4,884 ms	77,745 μs	PushPop	save flux Y
1.0%	2,893 s	1536	1,883 ms	1,875 ms	1,853 ms	4,849 ms	77,456 μs	PushPop	save flux Z
0.0%	2,784 s	1536	1,812 ms	[illegible]	[illegible]	5,202 ms	88,561 μs	PushPop	ctoprim
0.0%	231,607 ms	1536	150,786 μs	150,028 μs	139,678 μs	198,368 μs	6,764 μs	PushPop	godunov_fine unlock all octs

factor 10 speed-up

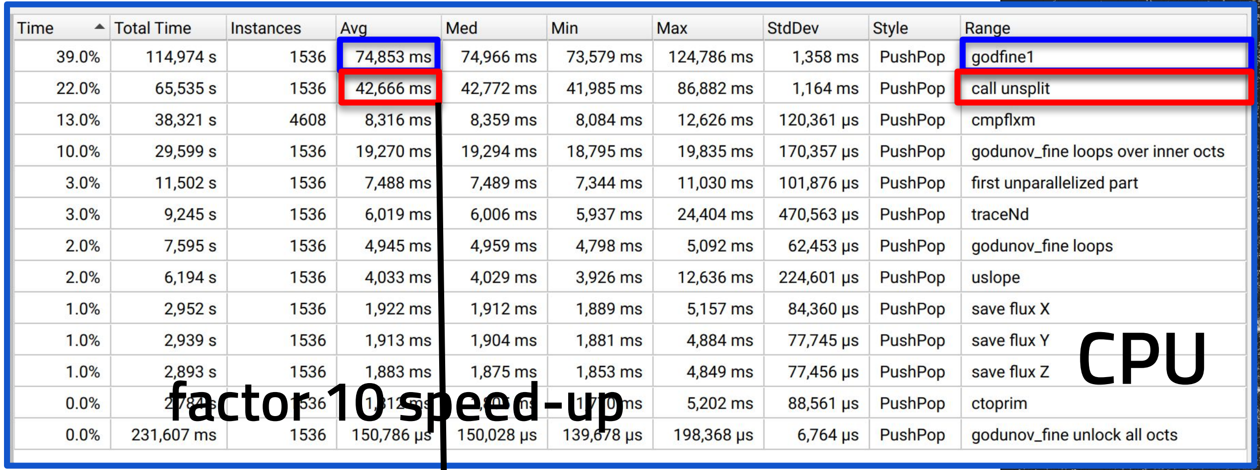


GPU

Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
49.0%	47,483 s	1536	30,914 ms	29,738 ms	25,495 ms	98,395 ms	7,173 ms	PushPop	godfine1
26.0%	25,935 s	1536	16,885 ms	16,385 ms	11,785 ms	71,956 ms	4,329 ms	PushPop	first unparallelized part
8.0%	7,759 s	1536	5,051 ms	4,743 ms	3,672 ms	36,269 ms	2,467 ms	PushPop	godunov_fine loops over inner octs
6.0%	6,615 s	1536	4,307 ms	4,226 ms	3,681 ms	38,111 ms	1,162 ms	PushPop	call unsplit
5.0%	5,609 s	1536	3,652 ms	3,612 ms	3,075 ms	15,200 ms	600,606 μs	PushPop	ctoprim
2.0%	2,112 s	1536	1,375 ms	1,554 ms	528,941 μs	31,943 ms	1,465 ms	PushPop	godunov_fine loops
0.0%	397,297 ms	1536	258,657 μs	196,445 μs	188,089 μs	33,599 ms	1,174 ms	PushPop	godunov_fine unlock all octs
0.0%	311,673 ms	4608	67,637 μs	62,396 μs	58,275 μs	10,126 ms	157,362 μs	PushPop	cmpflxm
0.0%	214,278 ms	1536	139,504 μs	137,149 μs	133,125 μs	3,299 ms	80,693 μs	PushPop	traceNd
0.0%	131,411 ms	1536	85,554 μs	61,875 μs	59,161 μs	34,311 ms	874,105 μs	PushPop	save flux Y
0.0%	110,519 ms	1536	71,952 μs	63,206 μs	60,237 μs	10,159 ms	261,684 μs	PushPop	uslope
0.0%	101,006 ms	1536	65,759 μs	62,842 μs	60,340 μs	2,166 ms	57,353 μs	PushPop	save flux Z
0.0%	98,094 ms	1536	63,863 μs	62,495 μs	59,268 μs	1,096 ms	26,479 μs	PushPop	save flux X



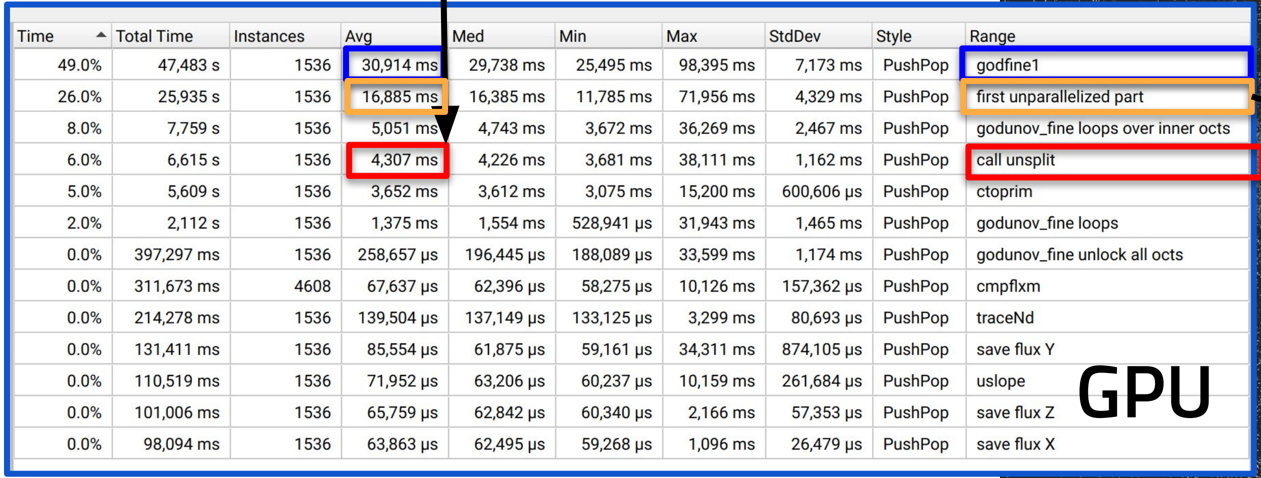
However



Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
39.0%	114,974 s	1536	74,853 ms	74,966 ms	73,579 ms	124,786 ms	1,358 ms	PushPop	godfine1
22.0%	65,535 s	1536	42,666 ms	42,772 ms	41,985 ms	86,882 ms	1,164 ms	PushPop	call unsplit
13.0%	38,321 s	4608	8,316 ms	8,359 ms	8,084 ms	12,626 ms	120,361 μs	PushPop	cmpflxm
10.0%	29,599 s	1536	19,270 ms	19,294 ms	18,795 ms	19,835 ms	170,357 μs	PushPop	godunov_fine loops over inner octs
3.0%	11,502 s	1536	7,488 ms	7,489 ms	7,344 ms	11,030 ms	101,876 μs	PushPop	first unparallelized part
3.0%	9,245 s	1536	6,019 ms	6,006 ms	5,937 ms	24,404 ms	470,563 μs	PushPop	traceNd
2.0%	7,595 s	1536	4,945 ms	4,959 ms	4,798 ms	5,092 ms	62,453 μs	PushPop	godunov_fine loops
2.0%	6,194 s	1536	4,033 ms	4,029 ms	3,926 ms	12,636 ms	224,601 μs	PushPop	uslope
1.0%	2,952 s	1536	1,922 ms	1,912 ms	1,889 ms	5,157 ms	84,360 μs	PushPop	save flux X
1.0%	2,939 s	1536	1,913 ms	1,904 ms	1,881 ms	4,884 ms	77,745 μs	PushPop	save flux Y
1.0%	2,893 s	1536	1,883 ms	1,875 ms	1,853 ms	4,849 ms	77,456 μs	PushPop	save flux Z
0.0%	2,784 s	1536	1,812 ms	[illegible]	[illegible]	5,202 ms	88,561 μs	PushPop	ctoprim
0.0%	231,607 ms	1536	150,786 μs	150,028 μs	139,678 μs	198,368 μs	6,764 μs	PushPop	godunov_fine unlock all octs

CPU

factor 10 speed-up

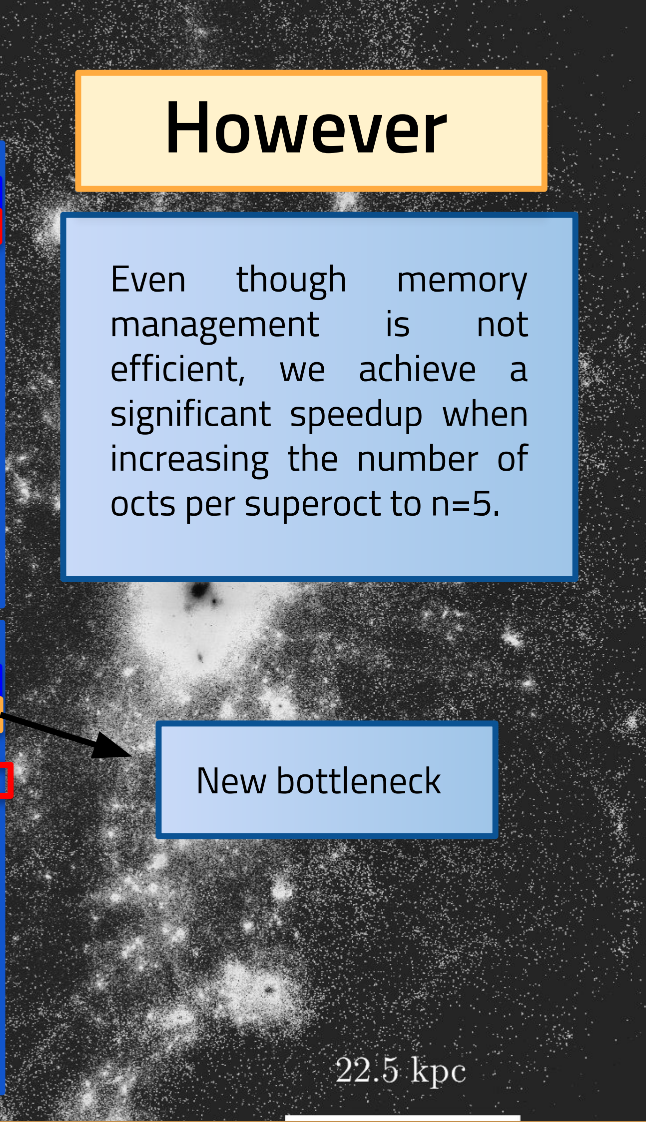


Time	Total Time	Instances	Avg	Med	Min	Max	StdDev	Style	Range
49.0%	47,483 s	1536	30,914 ms	29,738 ms	25,495 ms	98,395 ms	7,173 ms	PushPop	godfine1
26.0%	25,935 s	1536	16,885 ms	16,385 ms	11,785 ms	71,956 ms	4,329 ms	PushPop	first unparallelized part
8.0%	7,759 s	1536	5,051 ms	4,743 ms	3,672 ms	36,269 ms	2,467 ms	PushPop	godunov_fine loops over inner octs
6.0%	6,615 s	1536	4,307 ms	4,226 ms	3,681 ms	38,111 ms	1,162 ms	PushPop	call unsplit
5.0%	5,609 s	1536	3,652 ms	3,612 ms	3,075 ms	15,200 ms	600,606 μs	PushPop	ctoprim
2.0%	2,112 s	1536	1,375 ms	1,554 ms	528,941 μs	31,943 ms	1,465 ms	PushPop	godunov_fine loops
0.0%	397,297 ms	1536	258,657 μs	196,445 μs	188,089 μs	33,599 ms	1,174 ms	PushPop	godunov_fine unlock all octs
0.0%	311,673 ms	4608	67,637 μs	62,396 μs	58,275 μs	10,126 ms	157,362 μs	PushPop	cmpflxm
0.0%	214,278 ms	1536	139,504 μs	137,149 μs	133,125 μs	3,299 ms	80,693 μs	PushPop	traceNd
0.0%	131,411 ms	1536	85,554 μs	61,875 μs	59,161 μs	34,311 ms	874,105 μs	PushPop	save flux Y
0.0%	110,519 ms	1536	71,952 μs	63,206 μs	60,237 μs	10,159 ms	261,684 μs	PushPop	uslope
0.0%	101,006 ms	1536	65,759 μs	62,842 μs	60,340 μs	2,166 ms	57,353 μs	PushPop	save flux Z
0.0%	98,094 ms	1536	63,863 μs	62,495 μs	59,268 μs	1,096 ms	26,479 μs	PushPop	save flux X

GPU

Even though memory management is not efficient, we achieve a significant speedup when increasing the number of octs per superoct to n=5.

New bottleneck



Conclusions and Next steps

Impossible to complete to porting of Nbody component as long as the NVIDIA compiler is updated. Complete focus on hydrodynamics

We were able to port on GPU the majority of the subroutines associated with hydrodynamical component.

The code has a significant speed up in case of superoct level 5, but not superoct level 4

Optimizing memory management

Currently, each critical loops of the code is separately offloaded to the GPU using OpenACC directives, without any specific selection of variables to b used on the GPU. This leads to inefficient memory usage and continuous communication between host and device.

Initial attempts to employ OpenACC for GPU memory management have not yielded the desired results.

Improving memory movement could result in significant speed-ups, particularly in scenarios where superoct level 4.