RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

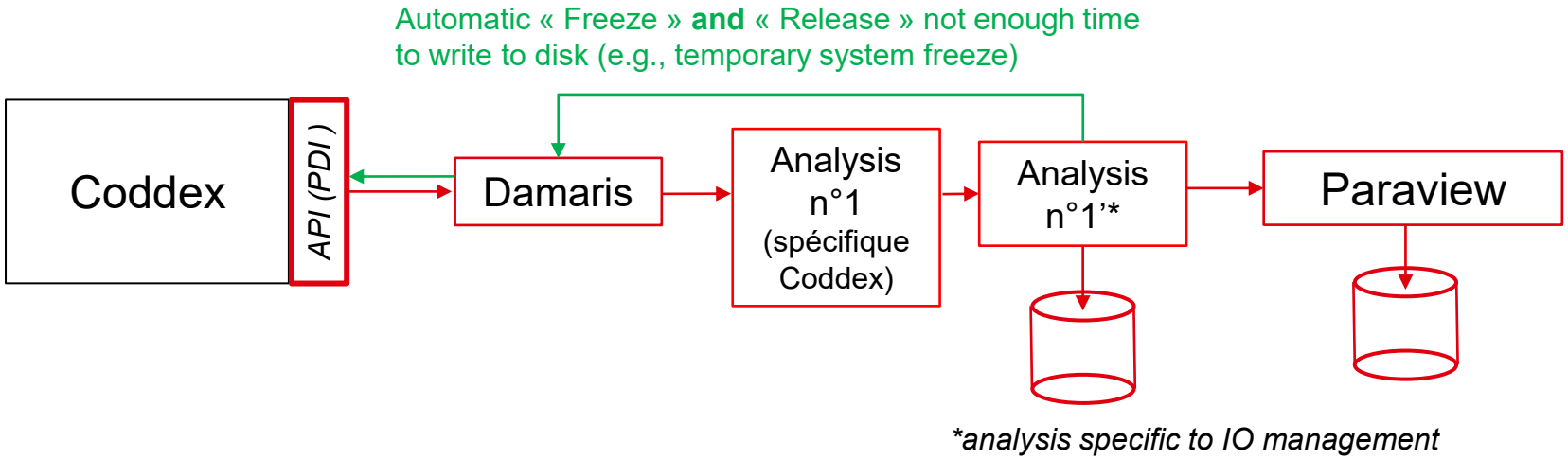
FRANCE 2030 PROGRAMME DE RECHERCHE NUMÉRIQUE POUR L'EXASCALE

ExaDoST - Work Package 1
Breakout session with Coddex & Dyablo

WP1 Leaders:
Francieli Boito (Université de Bordeaux) &
François Tessier (Inria Rennes)
Application representatives:
Laurent Colombet (CEA-DAM) and Arnaud Durocher (CEA)

Coddex

- **Develop an I/O analysis component (I/O orchestrator) to ensure data consistency and manage the in-memory data pipelining problem**
  - Anyway, the in-situ infrastructure has to be in place first
  - Unsure about who is going to do this
  - It may be related to the work with Gysela? We need to think more about this



**analysis specific to IO management*

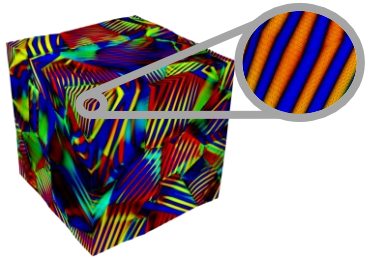
Dyablo

- **New data layout for AMR data (task 1.6/1.7 of WP1 led by CEA): Sylvain JOUBE, newly hired postdoc**
  - Efficient format for storage and fast algorithms for fast processing (slices, subdomains, etc)
  - CEA-DAM is also interested in this, they have similar AMR applications
- Currently: I/O is done to a single large HDF5 file (per time step)
  - That can change because Dyablo is very modular
  - Keeping the checkpoint independent from the domain distribution is a must (it can restart with a different number of MPI ranks)
  - Dyablo has two types of I/O: checkpoint and for post-processing, I/O is modular, it uses HDF5
- **Wish to make I/O asynchronous eventually**
- There is a trade-off to be studied between write and read performance (i.e. should we make write more complex to optimize reads?)
- **Work together on an I/O benchmark that we can distribute**

From a practical point of view: Sylvain will start doing state of the art and familiarizing himself with the issue, once he is designing the new format we will have **WP1 meetings to discuss the I/O performance considerations.**

Detailed comments

Coddex



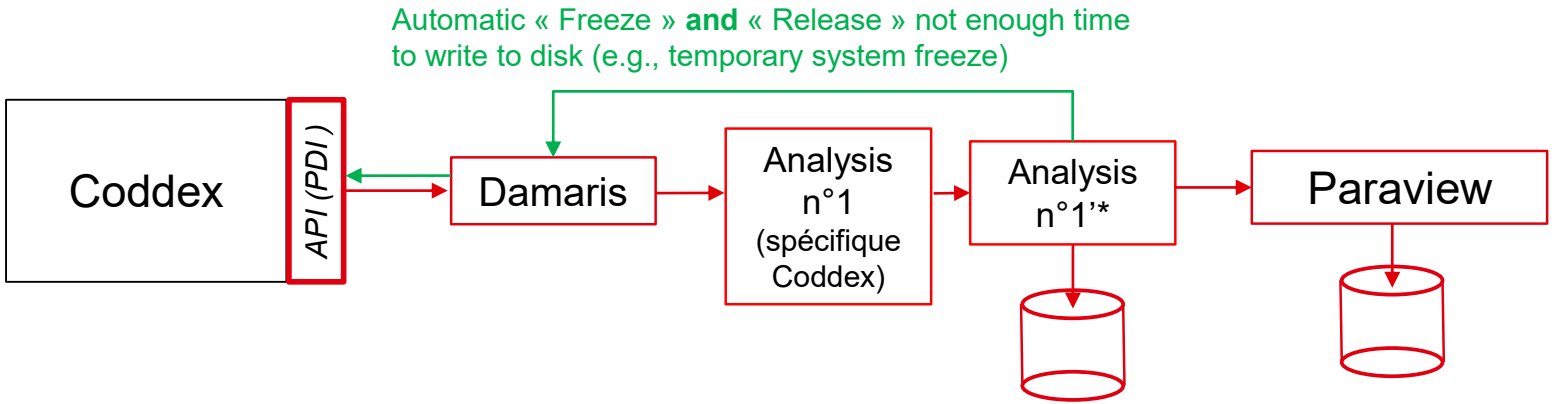
- The apps presentation: https://docs.google.com/presentation/d/1tw4MKGVqC6CPCXAAXzBgN6dhZqV-m7r0/edit?usp=sharing&ouid=108863493593436636873&rtpof=true&sd=true
- Simulation code that solves equations of continuum mechanics in dynamic hyperelasticity
  - Galerkin solver
- Legacy code
  - Started in 2000
  - Now C++17, with MPI/OpenMP. Python3 user interface
  - Use of DEISA/Damaris for in-situ viz with dedicated cores/nodes
  - te simulation uses the CPU cores only, the GPU can be used for analysis

Objectives in WP1

- **Develop an I/O analysis component (I/O orchestrator?) to ensure data consistency, trigger I/O at the right time and manage the in-memory data pipelining problem**
  - This point needs to be further discussed to better identify the requirements
  - The application sends data to staging cores/nodes (through Damaris, for example), and that data will be used for analysis. One of the consumers of the data will write it to disk as checkpoints (which are useful both for fault tolerance but also to revert the simulation with different parameters if problems are identified). The goal is to i) send data to disk, freeing memory soon enough so the next data piece can arrive. If that is not possible we have to stop Coddex until the data has been consumed. ii) verify that data is not corrupted.
  - a checkpoint could be skipped if required, but sometimes we need to save data for post-processing because that is where the interesting phenomena are

Objectives in WP1

- **Develop an I/O analysis component (I/O orchestrator?) to ensure data consistency, trigger I/O at the right time and manage the in-memory data pipelining problem**
  - This point needs to be further discussed to better identify the requirements
  - The application sends data to staging cores/nodes (through Damaris, for example), and that data will be used for analysis. One of the consumers of the data will write it to disk as checkpoints (which are useful both for fault tolerance but also to revert the simulation with different parameters if problems are identified). The goal is to i) send data to disk, freeing memory soon enough so the next data piece can arrive. If that is not possible we have to stop Coddex until the data has been consumed. ii) verify that data is not corrupted.
  - a checkpoint could be skipped if required, but sometimes we need to save data for post-processing because that is where the interesting phenomena are



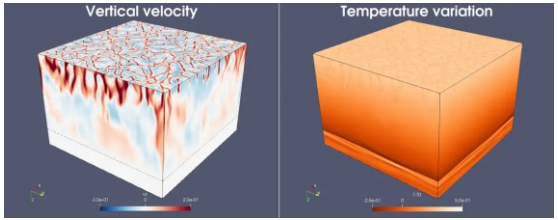
**analysis specific to IO management (wp1) ?*

do we have the human resources?

anyway, first the in-situ infrastructure has to be in place so we can proceed

possible link with the gysela work (Méline Trochon's thesis)? We have to think about it

Dyablo 1/3

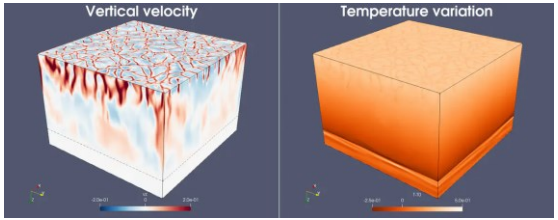


- Hardware-agnostic AMR code for astrophysics at Exascale
  - Cosmology, galaxies, solar/stellar
- Successor of RAMSES (failed to port on GPU)
- Written in C++, using external lib (HDF5, PABLO, …)
- Kokkos + MPI parallelism
- Supports exascale hardware
- Octree-based AMR (specific to Dyablo)
- Accessing the neighbors leads to random accesses
- Storage in 4D view (just a sequence of blocks, and then the geometry)
  - in one big HDF5 file, uses paraview unstructured mesh
- The AMR structure is dynamic, it may change every time step (it depends on the application)
- Is load imbalance a problem? Not really for now, because they have a coarse parallelism with a limited number of ranks and relying on shared memory
  - but maybe it could be an issue with new features (?)
  - load balancing is part of the loop, but load balancing for computation could lead to a load imbalance for I/O
    - but cells don't really vary that much in computation cost due to the AMR (we increase resolution where we need it)

I/O in Dyablo

- Checkpoint/restart of all the data. Dyablo-specific format
- Synthetic benchmarks available!

Dyablo 2/3



I/O in Dyablo

- Checkpoint/restart of all the data. Dyablo-specific format
- I/O is modular: new plugins can be added
- Synthetic benchmarks available!
- HDF5 format used
- Output files are independent from MPI distribution -> this is a requirement for a new data format

I/O benchmark

- there is a github, and slack, since it is modular we could make an I/O benchmark
- they have very simple benchmarks yet, without load imbalance problems

Dyablo uses GPU only

Dyablo - 3/3

Objectives in WP1

- **New data layout for AMR data (task 1.6/1.7 of WP1 led by CEA): Sylvain JOUBE, newly hired postdoc**
  - efficient format for storage and fast algorithms for fast processing (slices, subdomains, etc)
  - "we want HDF5 for AMR data" (Julien B.)
  - CEA-DAM is also working on this, they have similar AMR applications
- Dyablo has two types of I/O: checkpoint and for post-processing, I/O is modular, it uses HDF5
  - it is a single file per write step
  - files are independent from domain distribution (it can be restarted with a different number of MPI ranks)
- **They would like to make I/O asynchronous eventually**
- Write vs read performance trade-off
- They can generate large amounts of data for benchmarking
- **Work together on a I/O benchmark that we can distribute**

From a practical point of view: Sylvain will start doing state of the art and familiarizing himself with the issue, once he is designing the new format we will have **WP1 meetings to discuss the I/O performance considerations.**

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

cea cnrs Inria

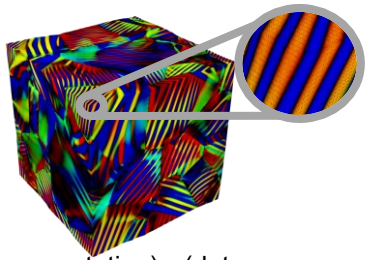
FRANCE 2030
PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

ExaDoST - Work Package 1
Breakout session with "other applications"
From 2024

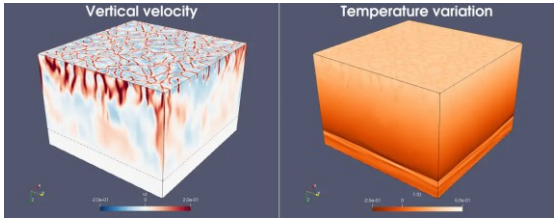
WP Leaders:
Francieli Boito (Université de Bordeaux) &
François Tessier (Inria Rennes)

Coddex (from our previous AG)

- Coddex is a FEM simulation (Smoothed Element Galerkin solver) - multi-physics simulation
- **50*10^6 elements .. may go up to 50-200 million**
- C++ code with Python interface (MPI + OpenMP) - 110k LOC, 40 regression tests + 11 other tests
  - has a end-to-end miniapp (input files, some demonstration of elasticity, and output files)
- MPI Parallelism
  - only MPI parallelism is in production
  - voronoi for domain decomposition that is balanced across nodes
- there is multiple high frequency visualisation pipelines
  - (legacy?, fastest) with postscript (“2D” reduced data projection of cube, multiple grid cells per visual representation) - (data volumes?)
  - vtk (which require full volume - (data volumes?)- 2x 10`6 elements (4096 cores, 64 MPI rank)
  - **simulated X-ray diffraction (4k cores, 64 mpi proces, 10 hours, 2 million elements) - Paraview - read IO is a bottleneck - currently post-mortem in situ candidate**
  - raw slow for large simulation, writes to single file(?) - 3.5 million (4k cores, 64 MPI ranks,, 10 hours) - read IO bottleneck - long to process, no grid? maybe does FFT
- Plan to move different analysis to in situ to bypass storage and reduce checkpoint frequency that need to be written - desire elasticity for additional analysis that may be triggered based on some condition during simulation runtime



Dyablo (from discussions with CEA)



- Dyablo is a C++ code for the simulation of astrophysical fluids with adaptive mesh refinement on heterogeneous architectures
- Two types of I/O: checkpoint and snapshots (for visu)
- Different backends to plug into the code

Snapshots

- code AMR
- snapshot en format pour paraview, il n'y a pas de format qui permet de représenter un AMR, donc c'est non-structuré
- pour représenter cette structure il faut stocker plus de données (de connectivité, la géométrie), encore plus que les choses à stocker (champ physique). format adapté: divisera par deux (plus ou moins): au moins 6 entiers par cellule, qui est un double
- données très compressibles
- synchrone "pour le moment", exécution plutôt gpu
- several TB of uncompressed snapshot data
- lecture qui pourrait charger une solution très dégradée, puis zoomer sur une partie intéressante (example: cosmologie pour voir les galaxies).
- données viennent de tous les processus, load-balancing marche plutôt bien pour le moment mais cela peut changer avec des pas de temps asynchrones.

Checkpoint

- ???

In-situ

- Dyablo n'est pas encore assez mature, pas encore de vraies runs de production