ABSTRAO

From Real to Possible

EVALUATION OF A CONTAINERIZED MASSIVELY PARALLEL HYBRID SPH SOLVER FOR TRANSIENT SOLID DYNAMICS

Gautier Dakin, Anthony Collé, Jérôme Limido

ABSTRAO SAS

French SME created in 2007

- Software Development
- Impetus exclusive distributor in France
- Training
- Studies of materials & structures under extreme loading

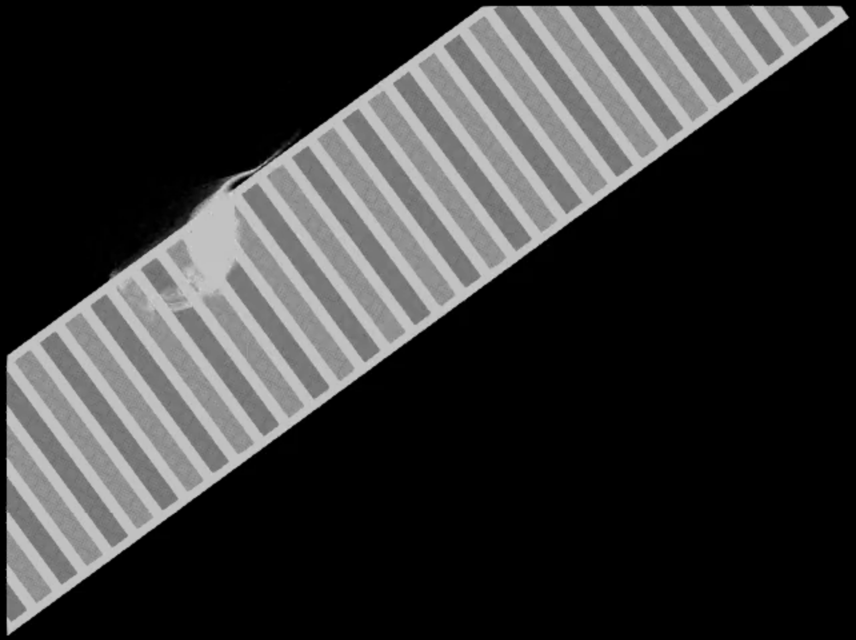
30% Growth Rate / Year

7 International Awards in the last 5 years

~50 International Publications

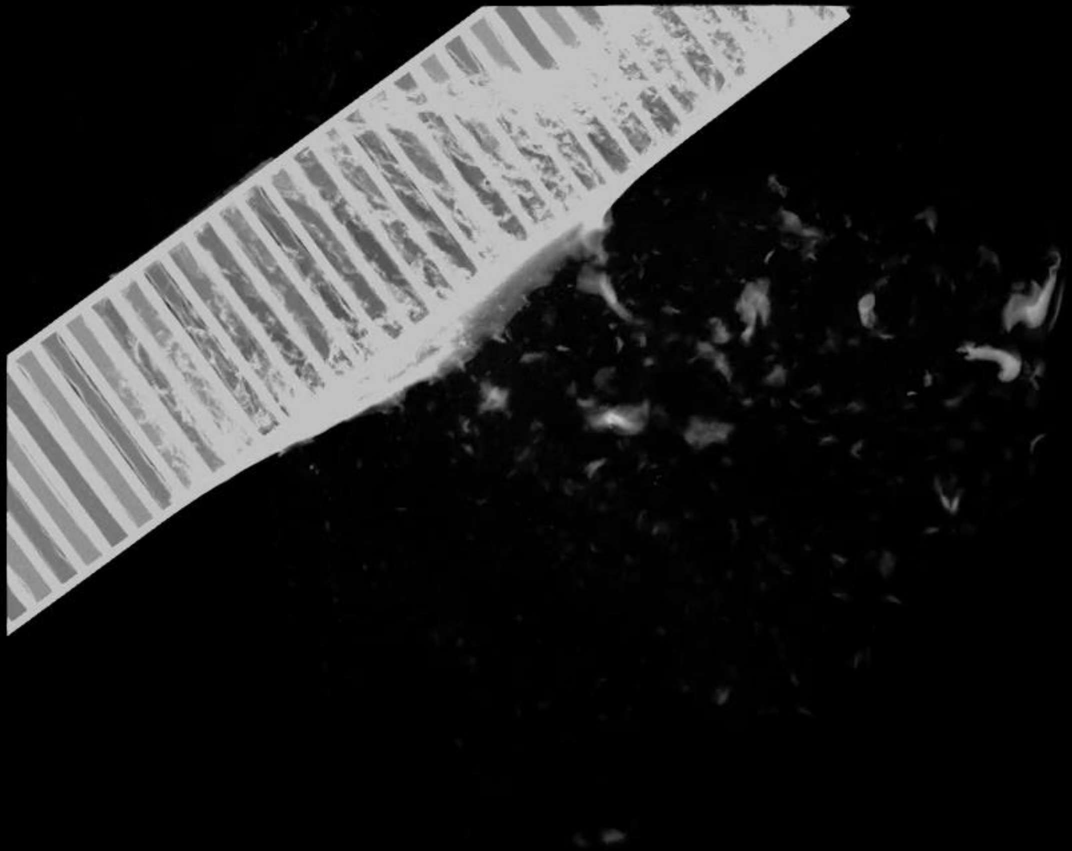
ABSTRAO Solver

First Official Release Q4 2024



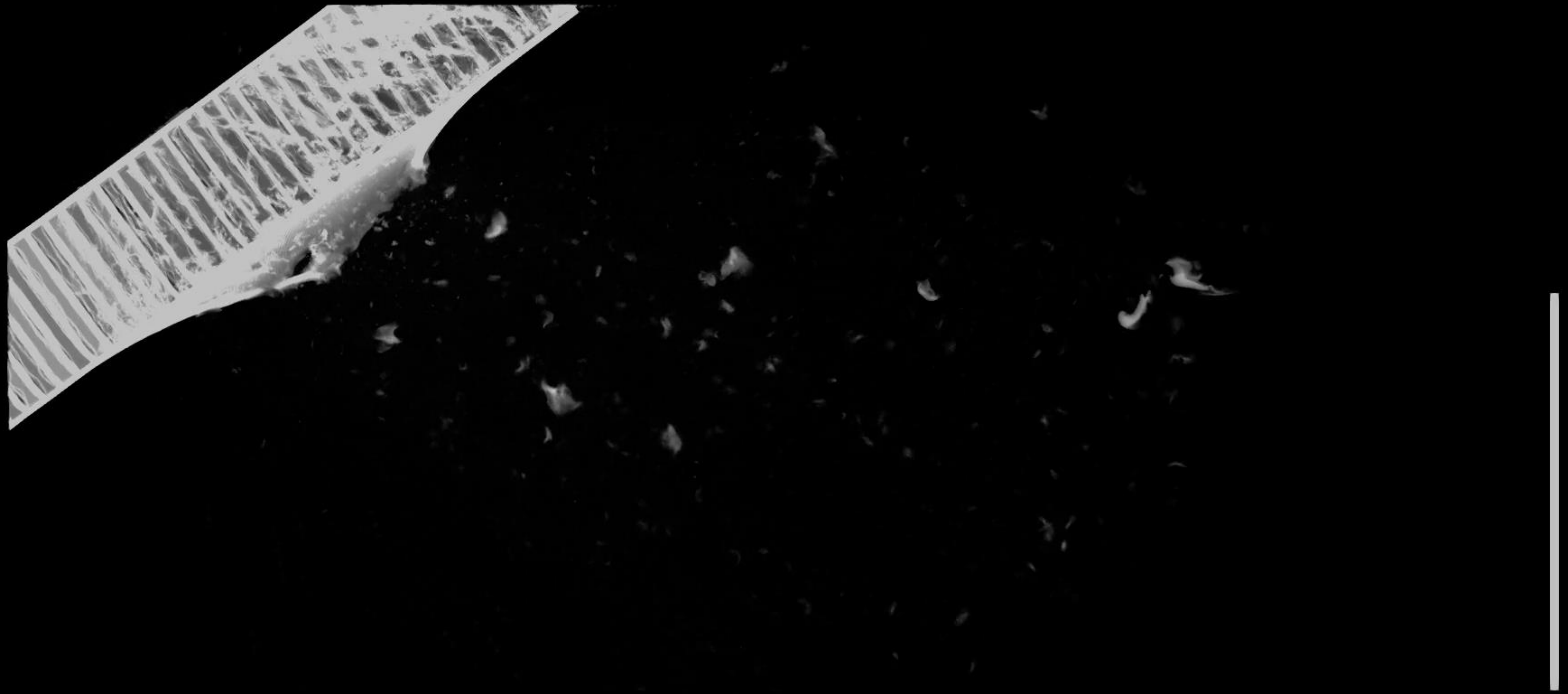
ABSTRAO Solver

First Official Release Q4 2024



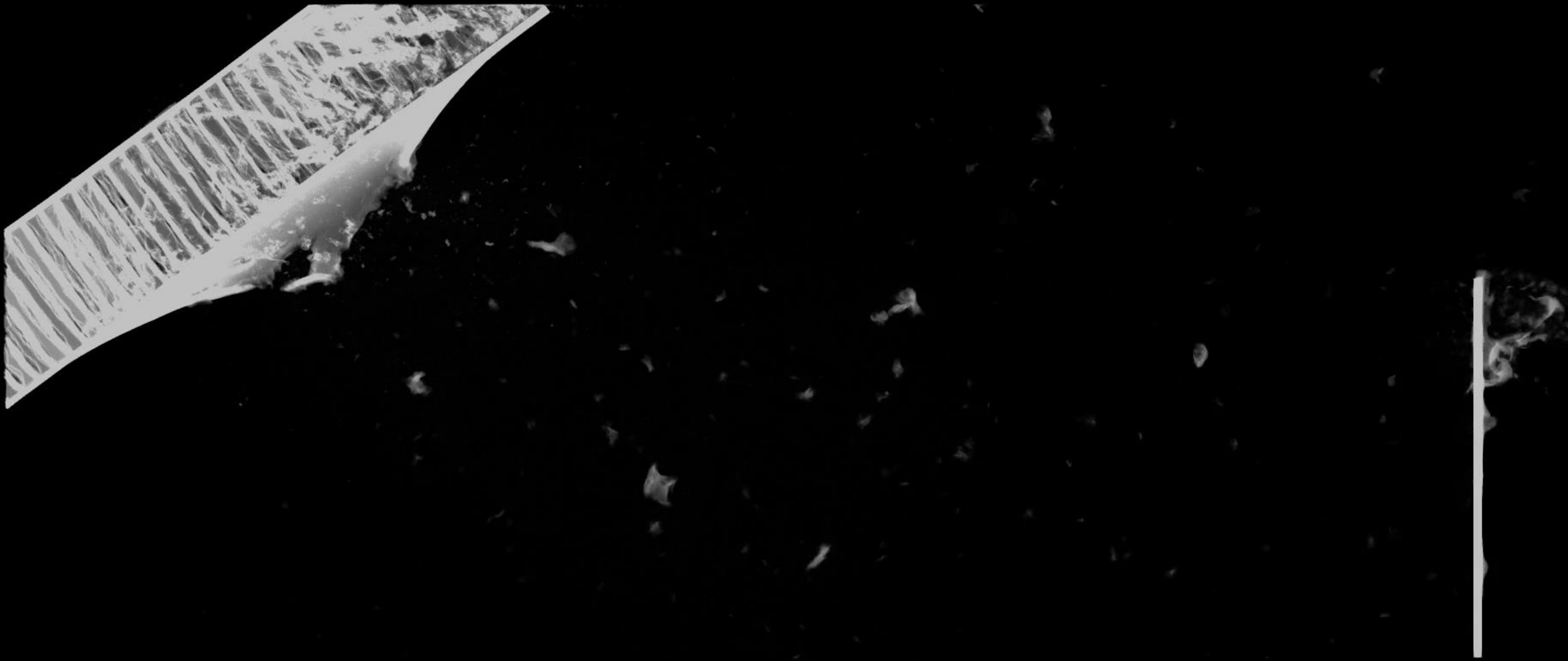
ABSTRAO Solver

First Official Release Q4 2024



ABSTRAO Solver

First Official Release Q4 2024



ABSTRAO Solver

First Official Release Q4 2024

Presentation

1. Containerization of Abstrao solver
   1. Choice of containerization technology
   2. Seamless deployment or not ?

2. Evaluation of performances of the containerized solution on workstation
   1. Linux workstation
   2. Windows workstation

3. Evaluation of performances of the containerized solution on cluster
   1. Performance comparisons
   2. Weak scalability

CONTAINERIZATION OF ABSTRAO SOLVER

Containerization of Abstrao solver

Choice has been made to use Apptainer:

- Security: Supports running unprivileged containers, reducing security risks.
- Portability: Allows easy movement of applications across different environments (laptops, clusters, clouds).
- Compatibility: Supports native integration with high-performance computing (HPC) workloads.
- Reproducibility: Ensures consistency across different systems and deployments.
- Performance: Minimal overhead for running applications close to native speed.

But…

Some questions remain in pratice:

- GPU: Does the container handle well GPUs ?
- Parallel environment: How does the container handle parallel environment (mpi, slurm, etc…) ?
- Windows OS : Does it work on Windows workstation/server ?
- Performance: What does "minimal overhead" really mean ?

Container environment

- Start with Ubuntu 22.04
- Install g++, cmake, …
- Install CUDA 11.8 for GPU version
- Compile and install from source OpenMPI 4.1.4 with CUDA support activated
- At the end, remove and purge every CUDA related libs inside the container

EVALUATION OF THE CONTAINERIZED VERSION ON WORKSTATION

Evaluation of performances on linux workstation

- Test are performed on a RHEL 8 workstation with 4xP6000

- List of the 6 versions tested:
  - Locally compiled without GPU direct for MPI transfers LNGD
  - Locally compiled with GPU direct for MPI transfers LGD
  - Apptainer solution without GPU direct for MPI transfers
    - Using MPI inside the container CNGDI
    - Using MPI outside the container CNGDO
  - Apptainer solution with GPU direct for MPI transfers
    - Using MPI inside the container CGDI
    - Using MPI outside the container CGDO

Apptainer usage

- With MPI inside
  - apptainer –quiet exec --nv --env OMP_NUM_THREADS=$NB_THREADS --bind $PWD $PATH_TO_CONTAINER mpirun –np $NMPI $EXE_INSIDE_CONTAINER
  - --nv allows usage of GPU cards inside the container
  - --env allows passing of environment variable like the number of threads
  - --bind mounts the desired path to the singularity
- With MPI outside
  - mpirun –np $NMPI apptainer –quiet exec --nv --env OMP_NUM_THREADS=$NB_THREADS --bind $PWD $PATH_TO_CONTAINER $EXE_INSIDE_CONTAINER

Performances on workstation

- No GPU-Direct

	LNGD	CNGDI	CNGDO
Normalized time	1	1.03	1.03

- With GPU-Direct

	LGD	CGDI	CGDO
Normalized time	1	1.02	1.02

- Very slight overhead for the containerized solution
- GPU Direct is completely functional
- No real MPI testing as everything is on a workstation

Apptainer usage on Windows OS

- Apptainer is used inside WSL2 (Windows subsystem Linux)
- Always with MPI inside
  - apptainer –quiet exec --nvccli --env OMP_NUM_THREADS=$NB_THREADS --bind $PWD $PATH_TO_CONTAINER mpirun –np $NMPI $EXE_INSIDE_CONTAINER
  - --nvccli allows usage of GPU cards inside the container
  - --env allows passing of environment variable like the number of threads
  - --bind mounts the desired path to the singularity
- Apptainer container is otherwise exactly the same as the one distributed on Linux.
- Usage of WSL2 add a slight overhead compared to the same machine on RHEL distribution.

EVALUATION OF THE CONTAINERIZED VERSION ON CLUSTER

Evaluation of performances on cluster

- Test are performed Supercomputer Turpan (ARM-accelerated Machine operated by Mesocenter CALMIP, Mesonet Project)
  - Total perfomance peak of 613,5 TF/s (1200 ARM core and 30 GPU A100).
  - Node contains :
    - 1 Ampere Altra Q80-30, 80 cores ARM v8.2, 3 Ghz : 1,9 TF/s Peak per socket
    - 2 GPU Nvidia A100-80 (80 GB HBM), 19,5 TF/s (FP64 TC)
    - 2 HDR Infiniband link (2 x 200 Gb/s)
    - 40,9 TF/s Peak Performance at node level
  - 15 Nodes in total

- We test the same 6 versions

- Container is built using cross compilation to handle arm64 environment on a amd64 machine

Performances on cluster – single node

- No GPU-Direct

	LNGD	CNGDI	CNGDO
Normalized time	1	1.03	1.03

- With GPU-Direct

	LGD	CGDI	CGDO
Normalized time	1	1.02	1.02

- Very slight overhead for the containerized solution
- No real MPI testing as everything is on a single node

Performances on cluster – multiple nodes

- No GPU-Direct

	LNGD	CNGDI	CNGDO
Normalized time	1	-------	1.02

- With GPU-Direct

	LGD	CGDI	CGDO
Normalized time	1	-------	1.02

- Very slight overhead for the containerized solution
- Embedded MPI does not support multiple nodes as expected

Weak scalability – full industrial case – Turpan + Apptainer

#Nodes	#MPI	#GPUs	Ideal scalability	Multi-nodes scalability
1	2	2	100%	
2	4	4	89%	100%
4	8	8	86%	97%
8	16	16	85%	95%
15	30	30	83%	93%

Weak scalability – full industrial case – Jean-Zay

#Nodes	#MPI	#GPUs	Ideal scalability	Multi-block scalability
1	8	8	100%	
2	16	16	103%	
4	32	32	93%	
8	64	64	74%	100%
16	128	128	74%	99%
32	256	256	59%	79%

CONCLUSION

Concluding remarks

- We developed a full transient solid dynamics code GPU + MPI which run from 4x to 100x faster than our competitors, with about 93% scalability on 32GPUs.
- Apptainer is well suited for GPU + MPI application
- Very slight overhead, pretty easy deployment and apptainer creation.
- The same container can be used on linux workstation, windows workstation through WSL2 and clusters.
  - On workstation, using embedded MPI is OK.
  - On clusters, MPI installed on it are to be used (through mpirun or srun)

- Acknowledgments

  - This work was supported by a French government grant managed by the Agence Nationale de la Recherche under the "Investissements d'avenir" program (reference "ANR-21-ESRE-0051"). This work was granted access to the MesoNET resources center and the MesoNET Project.
  - This work was performed using HPC resources from GENCI-IDRIS (Grant 2024-[AD010615633]).

We are hiring Interns, PhD Students or recent PhD graduates!

Exciting Opportunities: Join cutting-edge projects in high-performance computing.

Innovative Environment: Work on advanced with real-world impact.

Growth & Development: Enhance your skills in CUDA, MPI, and HPC technologies.

Collaborative Team: Be part of a dynamic team of experts and researchers.

Flexible Roles: Positions available for internships, thesis collaborations or full-time.

Location: Work with us in Plaisance du Touch, near Toulouse !

Apply Now at contact@abstrao.com and push the boundaries of technology with us!

THANK YOU FOR YOUR ATTENTION