Time series AMR data representation for out-of-core interactive visualization

W. ALEXANDRE-BARFF[1], H. DELEAU[1], J. Sarton[3], F. LEDOUX[2] and L. LUCAS[1]

1. Université de Reims Champagne-Ardenne, LICIIS, LRC DIGIT, France
2. CEA, DAM, DIF, LRC DIGIT, F-91297 Arpajon, France
3. Université de Strasbourg, ICube, UMR-CNRS 7357, France

PROBLEM

- 3D Numerical simulation → massive AMR time series
- data size >>>> current GPUs memory
- Interactive visualization of AMR data is not trivial
- Ad hoc decomposition of input data in ordered blocks
- Blocks management between ROM / RAM / VRAM

RELATED WORK

Adaptive Mesh Refinement (**AMR**): Introduced by [1] it seeks to combine the simplicity of structured grids with the advantages of local refinement to obtain a multi-resolution hierarchy of cells called *Block-Structured AMR* (**BS-AMR**).

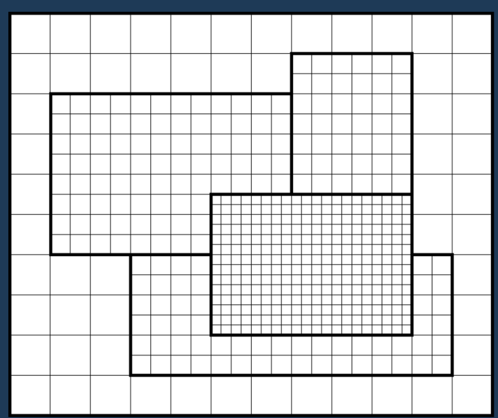


Figure 1: Three refinement level BS-AMR

Out-of-core approaches which address entire massive datasets, like Sarton et al. Proposition [2], introduce a data structure based on a caching strategy with a virtual memory addressing system coupled to efficient parallel management on GPU to provide efficient access to data in interactive time.

Our method proposes to extend this approach in a general-purpose framework allowing us to visualize and process interactively time-dependent AMR datasets on the GPU.

OVERVIEW

We propose in this poster – illustrated in figure 2 – a 3D time-dependent **AMR** data representation for out-of-core ready approach interactive volume visualization.

Our preprocessing step converts the time series of regular voxel grid into indexed Hilbert's curve path-based **BS-AMR** blocks.

We also introduce an integrating method of our data representation into virtual addressing data structure on GPU.

CONCLUSION

Our preprocessing shows potential for in-situ visualization as it averages a 30s time for a single step from - our biggest – dataset 2, as well as interactive volume visualization of an entire time series dataset. (See Fig. 8)

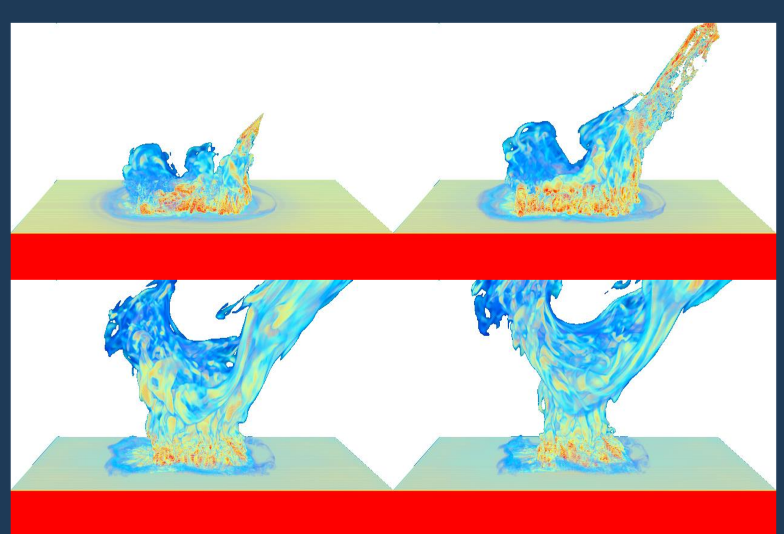


Figure 8: Time series AMR interactive volume visualization

METHODOLOGY

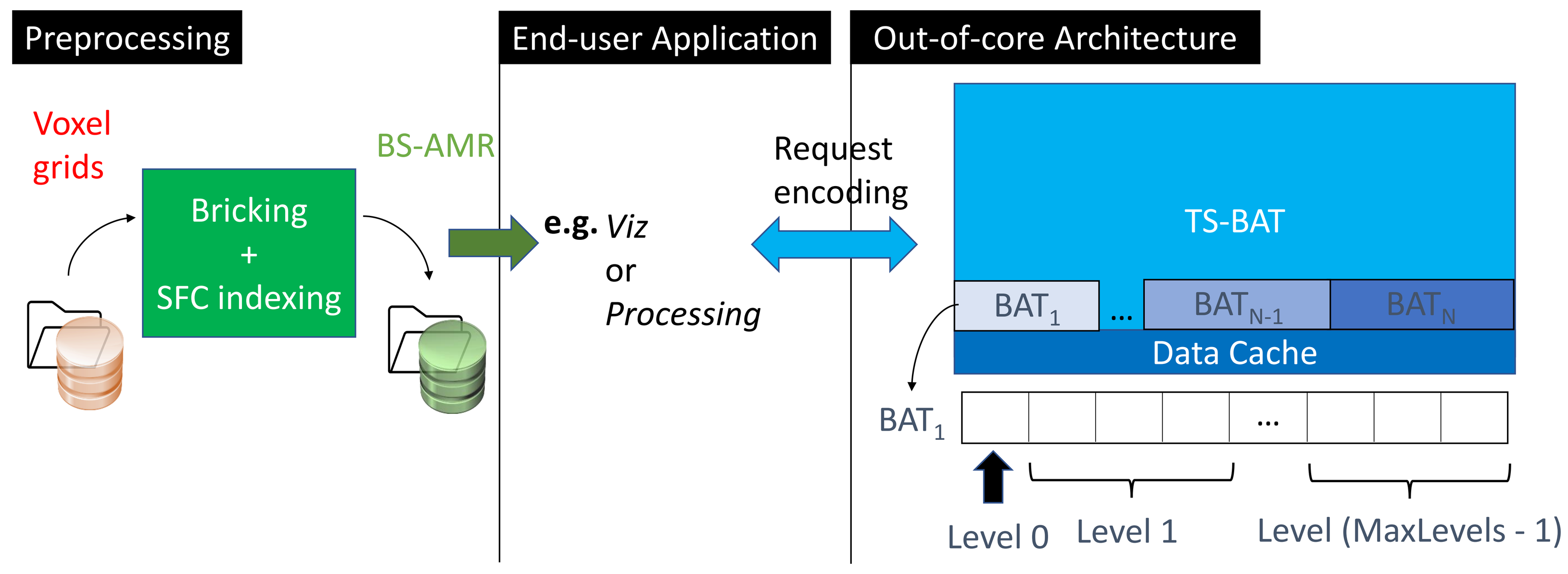


Figure 2: Overview time series AMR data pipeline

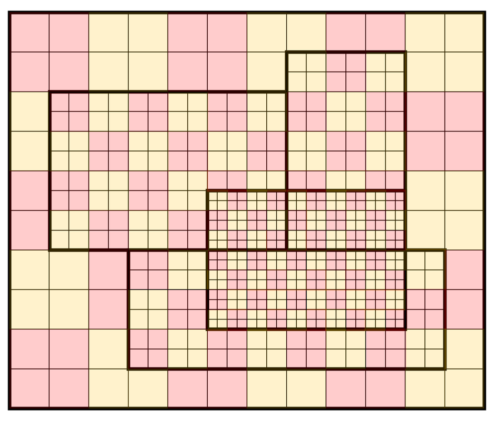
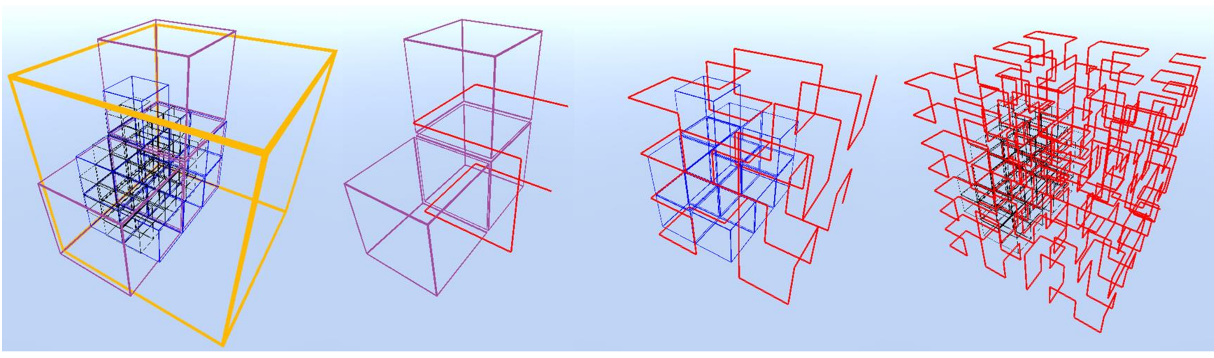


Figure 3: Bricking stage

Our conversion scheme (see Fig. 3) is largely derived from the work of [3], which produces a set of **AMR** bricks containing the same number of voxels with different spatial resolutions but the same memory footprint. (See Fig. 4 for **BS-AMR** topology with $\Gamma_0^{0..3}$ the union of 4 subgrids and Λ_0^1 grid of refinement level 1)

Once all the **AMR** brick data is obtained, we use a *Space-Filling Curve* (**SFC**) path to adress them uniquely and individually. Defined in a unit cube, this 3D **SFC** allows us to convert 3D position into a single Id. We chose Hilbert's curve for locality preservation [4]. (See Fig. 4 (b) (c) (d) for **SFC** indexing example)



(a) Card($\Gamma_0^{0..3}$) = {1,3,11,57} (b) Λ_0^1 3 bricks (c) Λ_0^2 11 bricks (d) Λ_0^3 57 bricks

Figure 4: Four refinement level **BS-AMR** topology of a unique binary volume of 188 × 136 × 85 voxels. Illustrates for each level the corresponding 3D Hilbert's curve traversal in red.

Finally, we store each **AMR** brick data scalars into individual binary files in addition to a supplementary *JavaScript Object Notation* (**JSON**) files at the top level of the hierarchy. We can see an overview of the organization in figure 5.

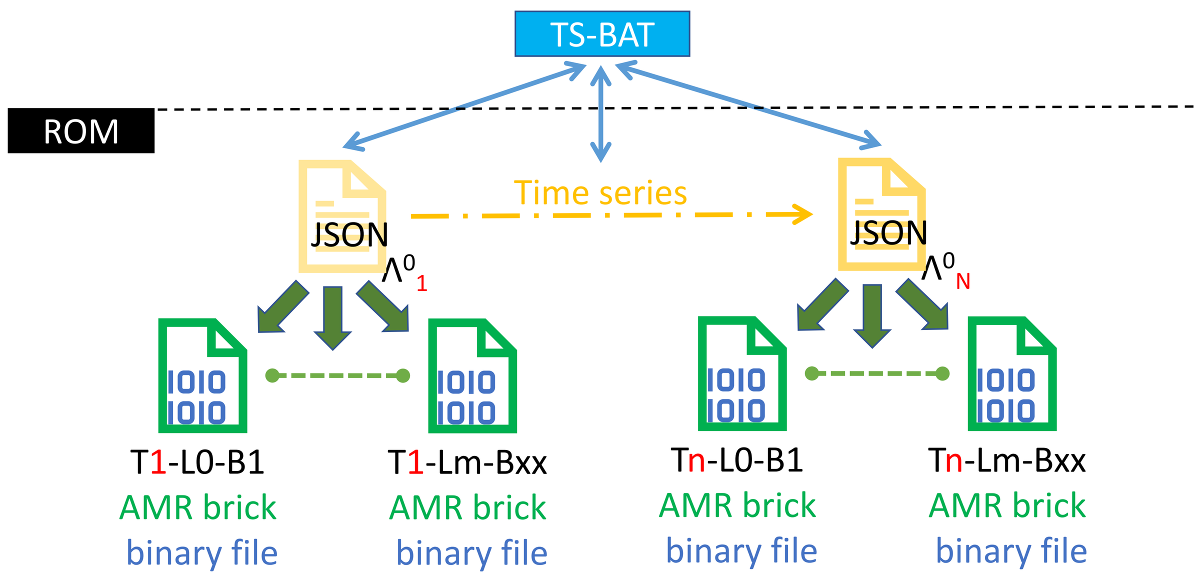


Figure 5: Produced data organization

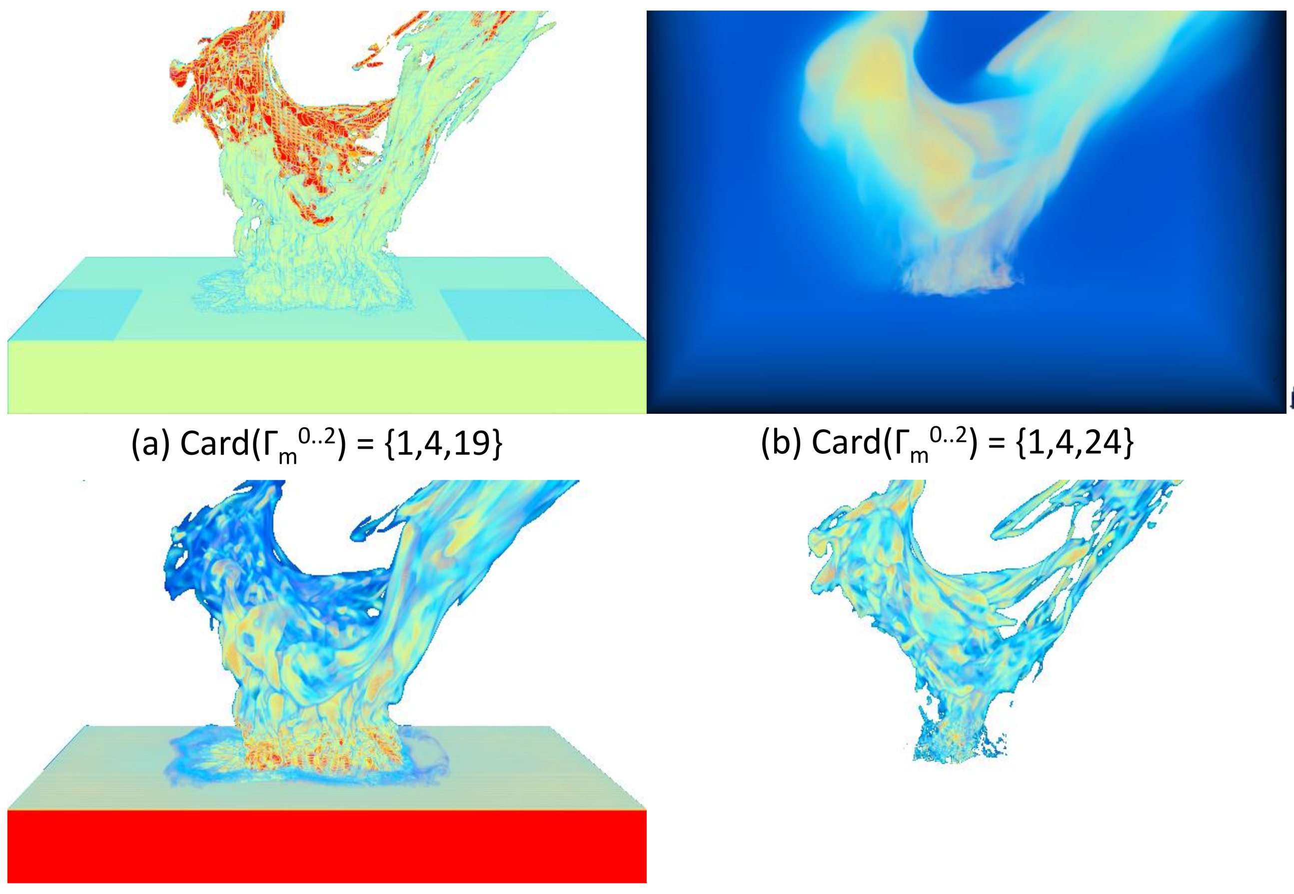
Then our integrating method called *Brick Addressing Table* (**BAT**) which is an **AMR** bricks array from a single step couple with our out-of-core approach called *Time Step Brick Addressing Table* (**TS-BAT**) which handles the collection of **BAT** alongside data cache management for the whole visualization stage.

RESULTS

<u>Dataset 1:</u> 269 volumes of 460 × 280 × 240 voxels on 128bits
<u>Preprocessing time:</u> 9min for 133Go memory size
<u>Average single step processing:</u> 2s for 184Mo

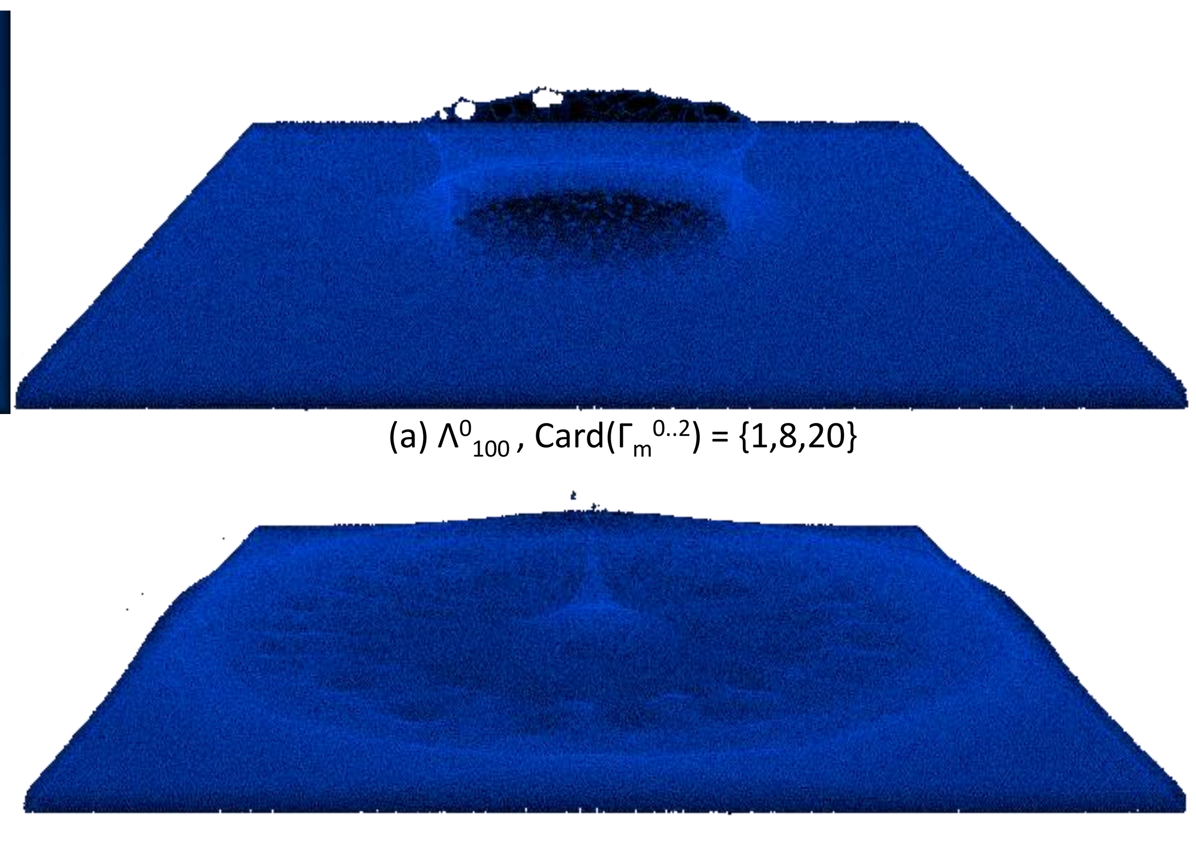
<u>Dataset 2:</u> 311 volumes of 1000 × 1000 × 1000 binary voxels
<u>Preprocessing time:</u> 2,5hours for 1,2To memory size
<u>Average single step processing:</u> 30s for 1,8Go

To validate our approach initially on CPU, we have integrated our code in the OpenVKL [5] AMR volume rendering engine.



(a) Card($\Gamma_m^{0..2}$) = {1,4,19} (b) Card($\Gamma_m^{0..2}$) = {1,4,24} (c) Card($\Gamma_m^{0..2}$) = {1,4,22} (d) Card($\Gamma_m^{0..2}$) = {1,4,16}

Figure 6: Visualization of the four scalar fields at time step Λ^0_{218} from dataset 1 at around 4,2fps



(a) Λ^0_{100}, Card($\Gamma_m^{0..2}$) = {1,8,20} (b) Λ^0_{200}, Card($\Gamma_m^{0..2}$) = {1,8,30}

Figure 7: Scalar visions of two separate time steps with their AMR hierarchy from dataset 2 at around 2fps

REFERENCES

[1] Marsha J Berger and Joseph Oliger, « Adaptive mesh refinement for hyperbolic partial differential equations », Journal of Computational Physics, vol.53, no. 3, pp. 484-512, 1984.
[2] Jonathan Sarton, Nicolas Courilleau, Yannick Remion, and Laurent Lucas, « Intractive Visualization and On-Demand Processing of Large Volume Data: A Fully GPU-based Out-of-Core Approach », IEEE Transactions on Visualization and Computer Graphics, vol. 26, no. 10, pp. 3008-3021, Oct. 2020.
[3] D.T. Graves J.N. Johnson N.D. Keen T.J. Ligocki D.F. Martin P.W. McCorquodale D. Modiano P.O. Schwartz T.D. Sternberg . Adams, P. Colella and B. Van Straalen, « Chombo software package for amr applications – design document », Report LBNL-6616E, Lawrence Berkeley National Laboratory Technical Report, 2015.
[4] Revital Dafner, Daniel Cohen-Or and Yossi Matias, « Context-based space filing-curves », Computer Graphics Forum, vol.19, 05 2000.
[5] Intel, « Openvkl high performance volume kernels », https://www.openvkl.org/, 2020, Accessed: 2022-02-15.

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

LICIIS DIGIT

cea

EG'22 TheEvent