GPU Technology Conference
Munich, Germany
09-11 Oct 2018

A Fully GPU-Based Out-Of-Core Approach to Handle Large Volume Data

Nicolas Courilleau[1,2], Jonathan Sarton[1], Florent Duguet[1,3],
Yannick Remion[1] and Laurent Lucas[1]

1 – Université de Reims Champagne-Ardenne, France

2 – Neoxia, France

3 – Altimesh, France

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

neoxia

Outline

❶ Background and motivation

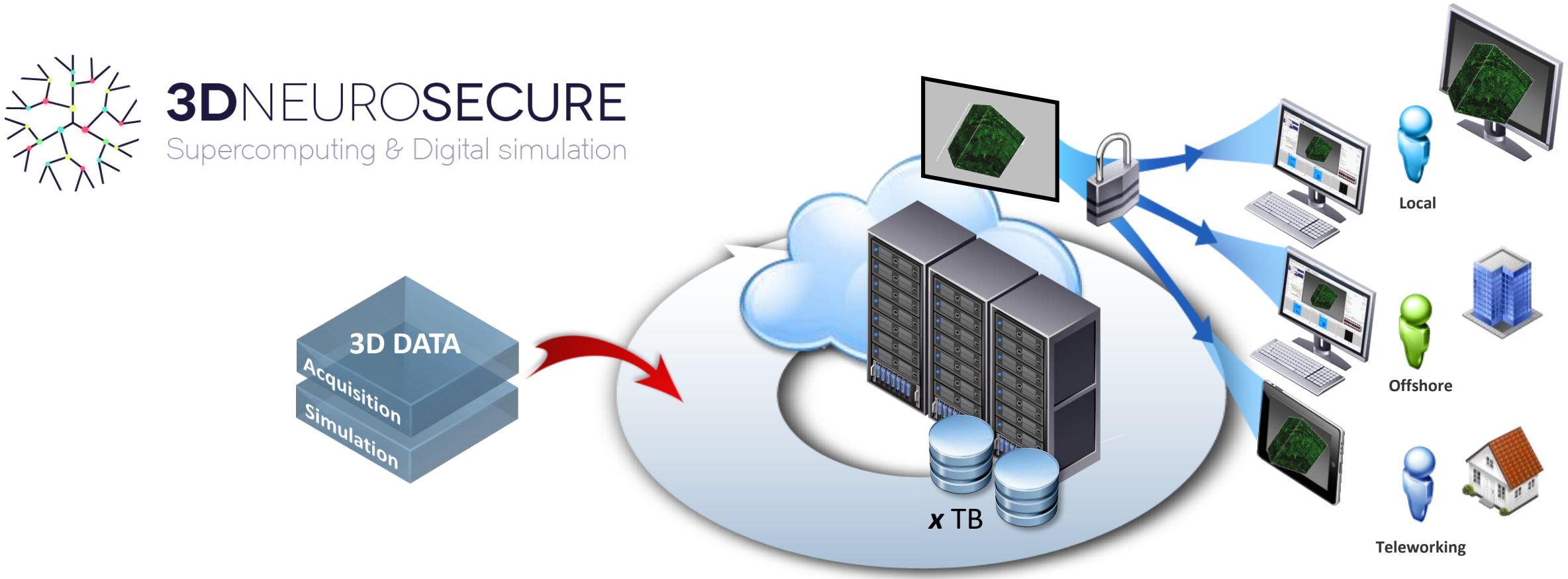
❷ Previous works

❸ Out-of-core model presentation

❹ Model in action: application to visualization

❺ Conclusion and outlook

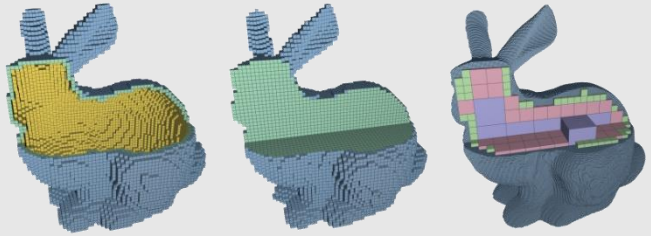
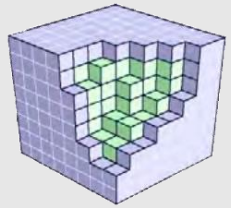
Context

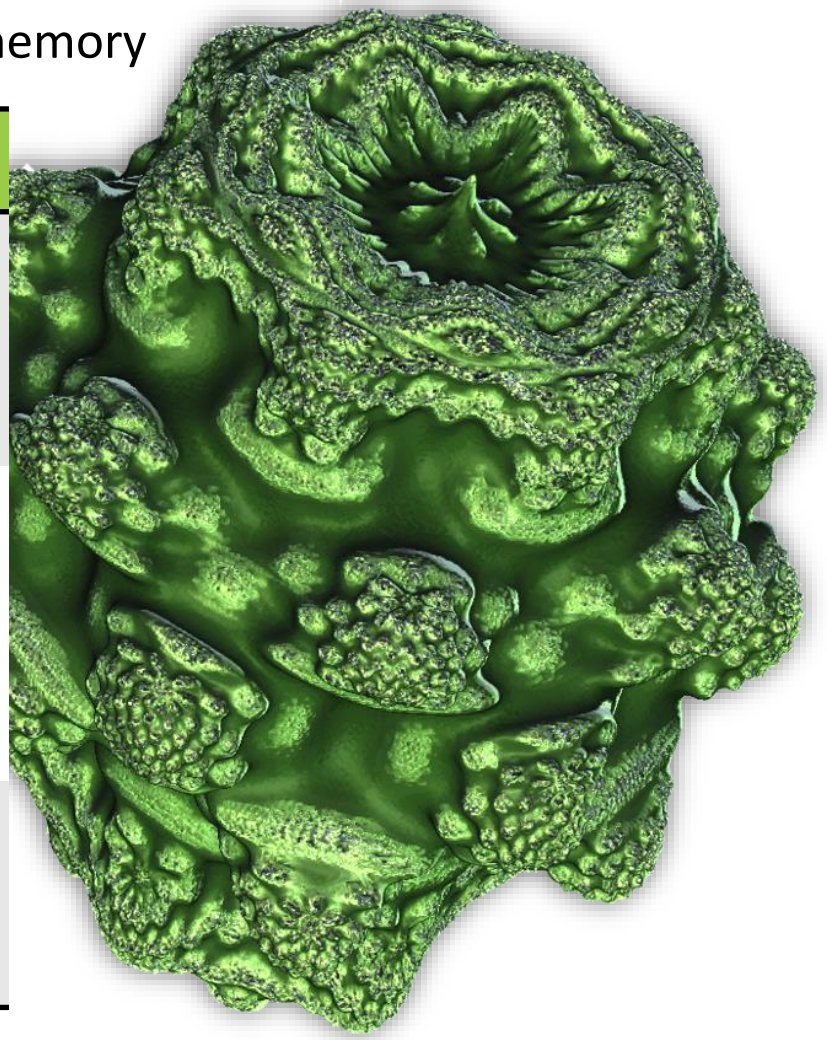


- Targets HPC of 3DNeuroSecure
  - Interactive **processing** and **visualization** (virtual microscopy, DVR) of very large biomedical datasets
- Accelerating drug discovery for Alzheimer disease

Problematic

- Designing out-of-core algorithms
- Voxel representation → High volume of data >> CPU and GPU memory

Domain/Application	Data size
Mesh voxelization 	~ 100 GB
Histology Electron microscopy 	~ 100 GB to several TB
Regular 3D grid 	And beyond



4352^3 (RGBA – 32bits) ≈ 330 GB

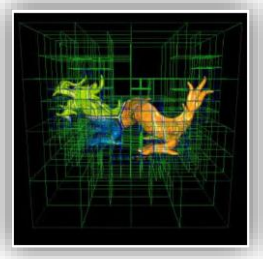
Previous works

Previous works

2009

[Crassin et al.]

ACM SIGGRAPH i3D



2011

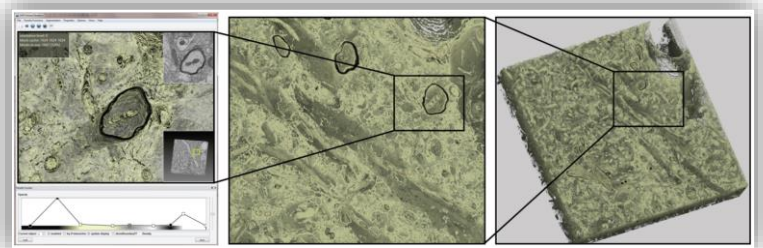
[Klaus Engel]

IEEE Symposium on Large Data Analysis & Visualization

2012

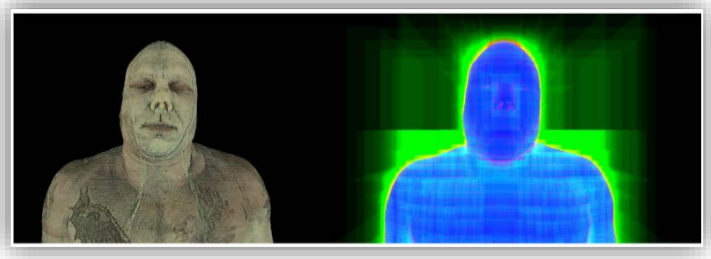
[Hadwiger et al.]

IEEE Transaction on Visualisation & Computer Graphics



2013

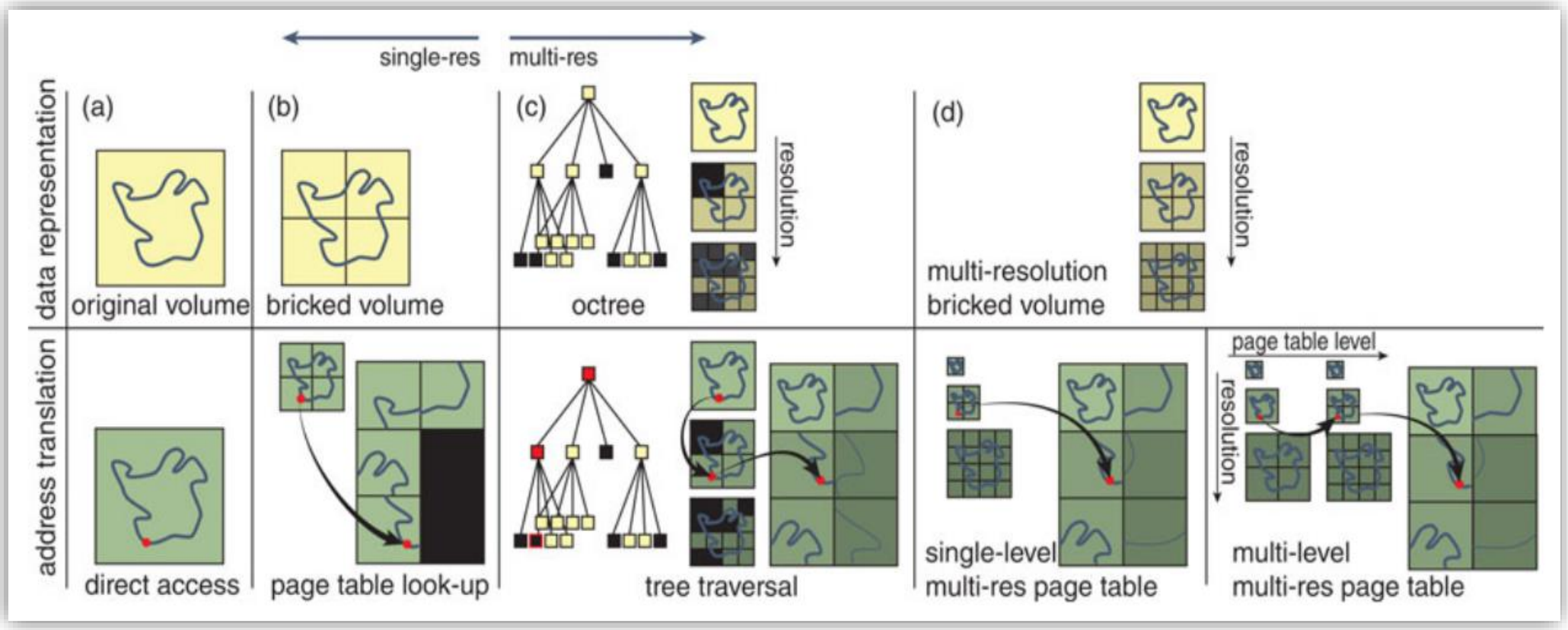
[Fogal et al.]



IEEE Symposium on Large Data Analysis & Visualization

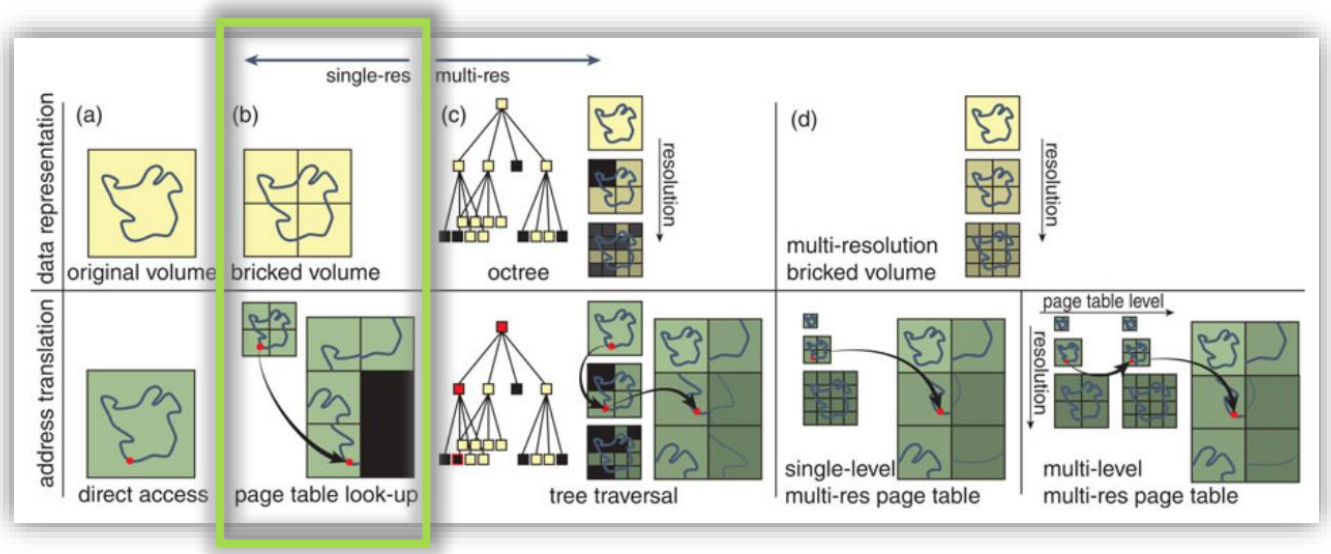
Previous works… at a glance

- Address translation taxonomy [Beyer et al. 2015]

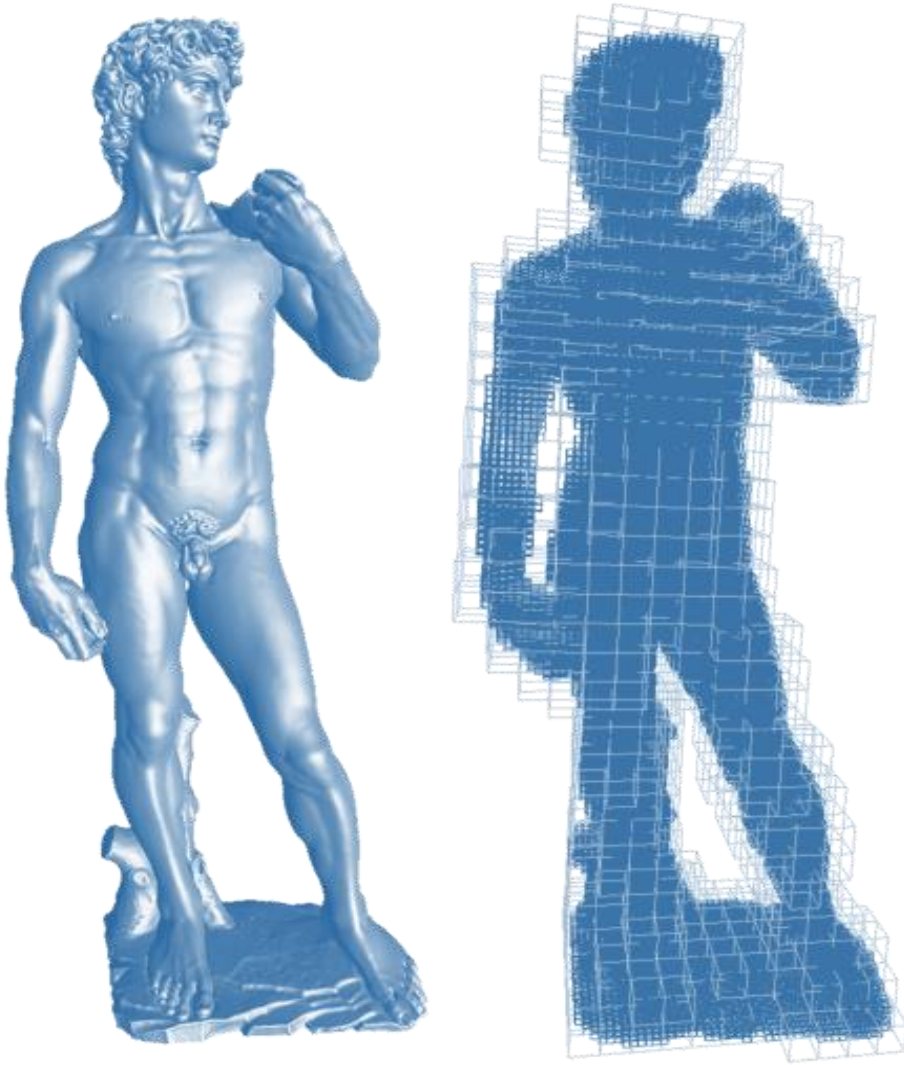


Previous works… at a glance

- Bricking: Page table look-up
- Octree multi-resolution: tree traversal
- Multi-resolution page table

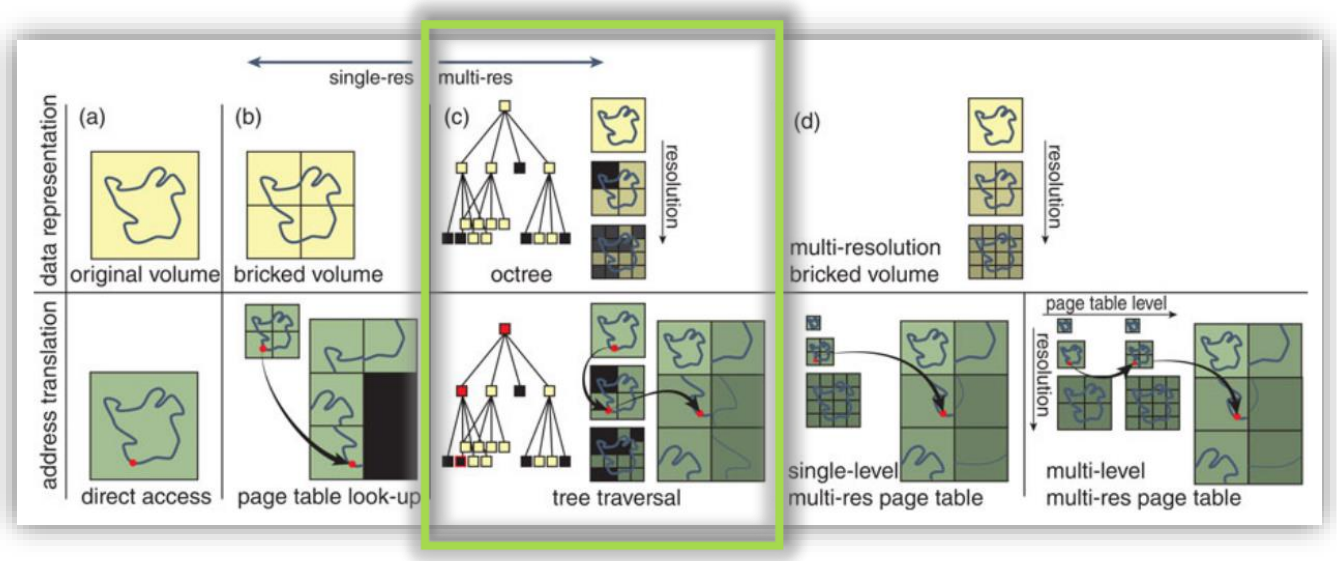


[Beyer et al. 2015]

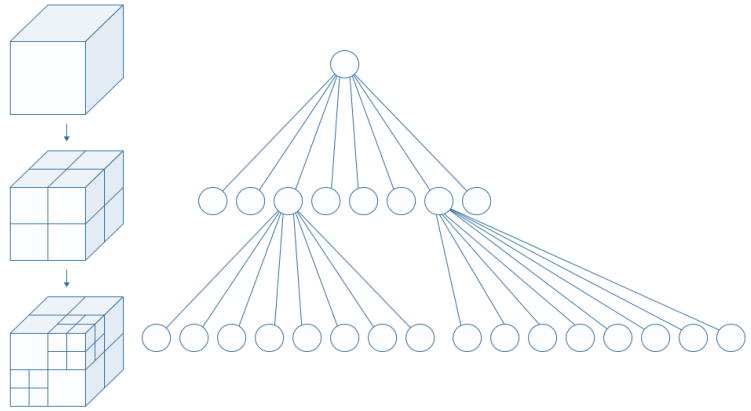


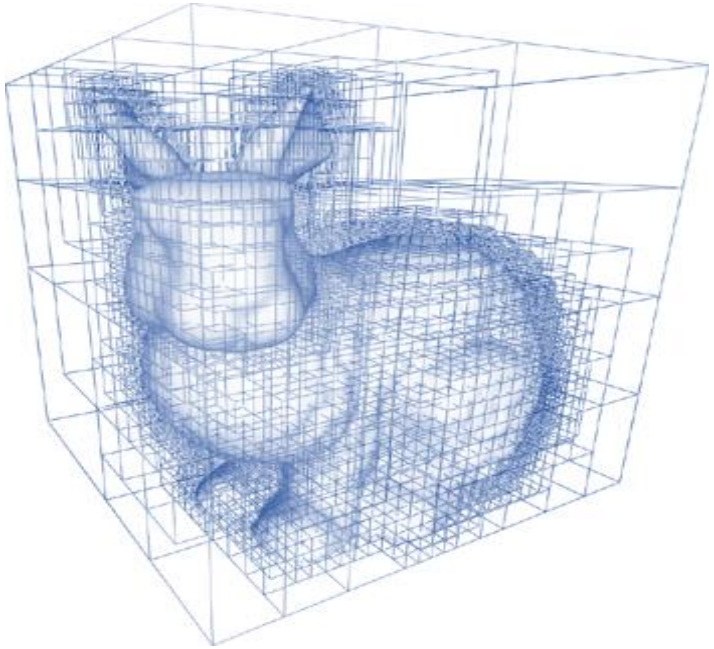
Previous works… at a glance

- Bricking: Page table look-up
- Octree multi-resolution: tree traversal
- Multi-resolution page table



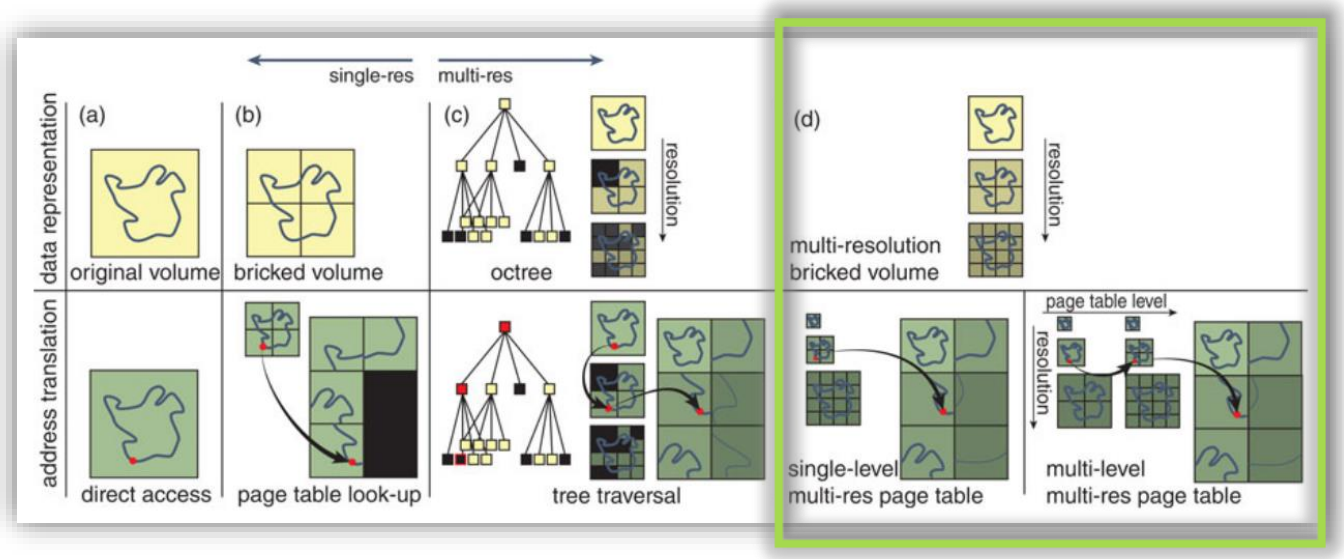
[Beyer et al. 2015]



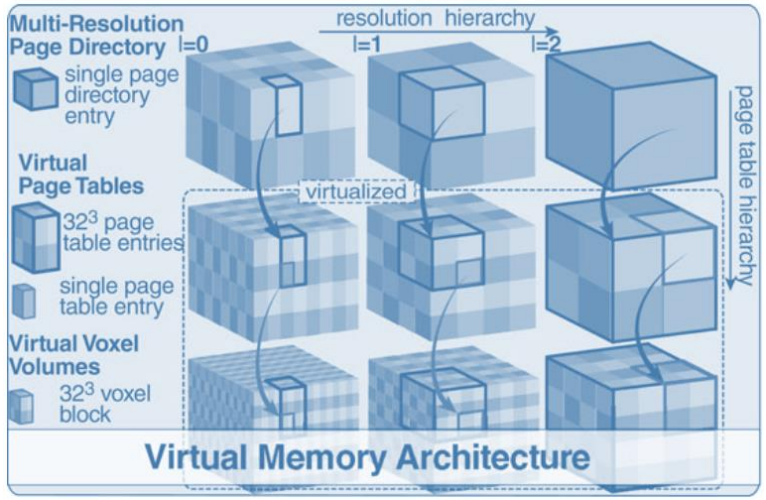


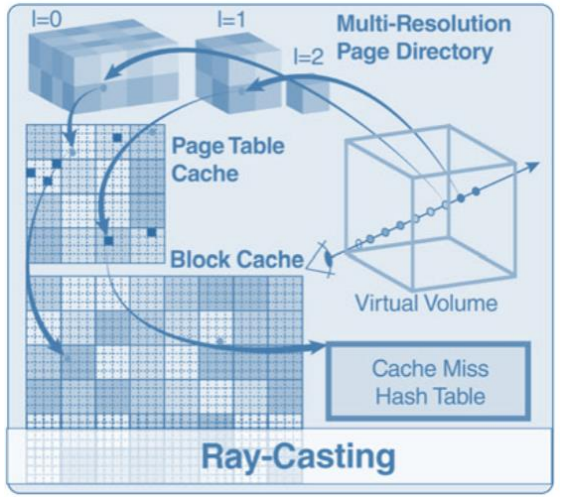
Previous works… at a glance

- Bricking: Page table look-up
- Octree multi-resolution: tree traversal
- Multi-resolution page table



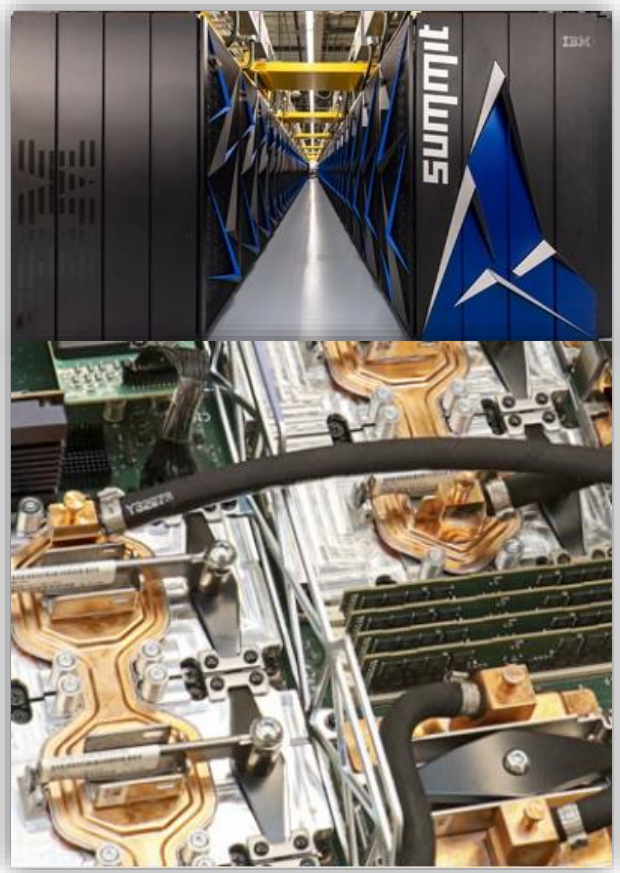
[Beyer et al. 2015]





And Nvidia – Pascal / Volta unified memory

- GPU memory oversubscription (unified memory)
  - Limited to host memory / OS specs limitation
  - Volume decomposition still needed
- Volta using
  - Nvidia Tesla V100
  - IBM Power 9
  - NVLink 2 (+ OS ATS)
  - Unix « mmap »
  - Unix kernel 4.16 (at least)
- Limitations
  - ATS over NVLink 2 = Power 9
  - NVLink 2 = Tesla V100
  - No page fault control
  - No texture memory



Summit - DOE/SC/Oak Ridge National Laboratory

[Everything you need to know about unified memory, Nikolay Sakharnykh, GTC 2018]

Our contributions

- GPU based out-of-core data management
  - Multiresolution multilevel page table hierarchy
  - Managed entirely on GPU
- Any kind of applications (regular 3D grids of voxels)
  - Interactive visualization
  - On-demand data processing
  - Both at the same time
- CPU – GPU communications reduced
- Complete pipeline – From storage to GPU

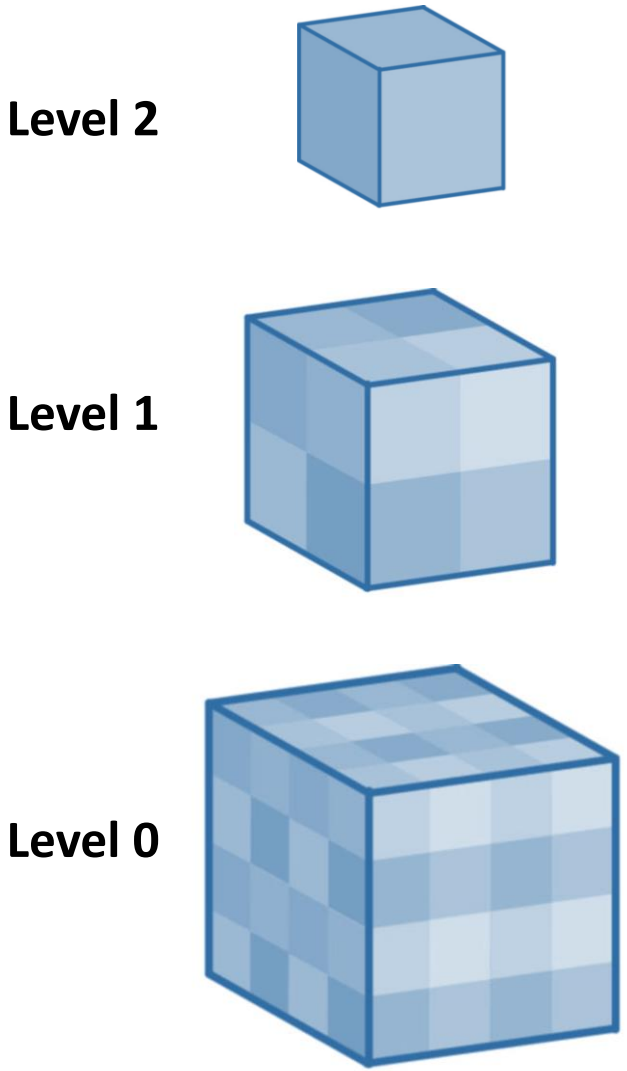
In addition,

- Multi OS support, since Kepler architecture

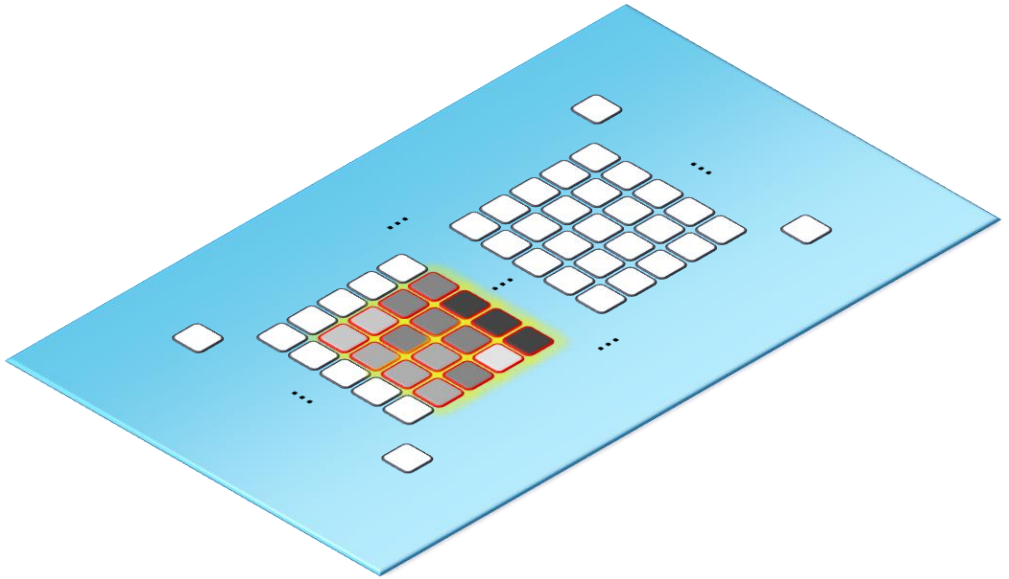
Out-of-core model presentation

Data representation and storage

- (1) Multiresolution – Level of details
- (2) Bricking – Level subdivision
  - Allows the out-of-core approach
- (1) + (2) = Bricked multiresolution 3D pyramid
- Bonus: Data compression (LZ4 – Loss less and real-time decompression)

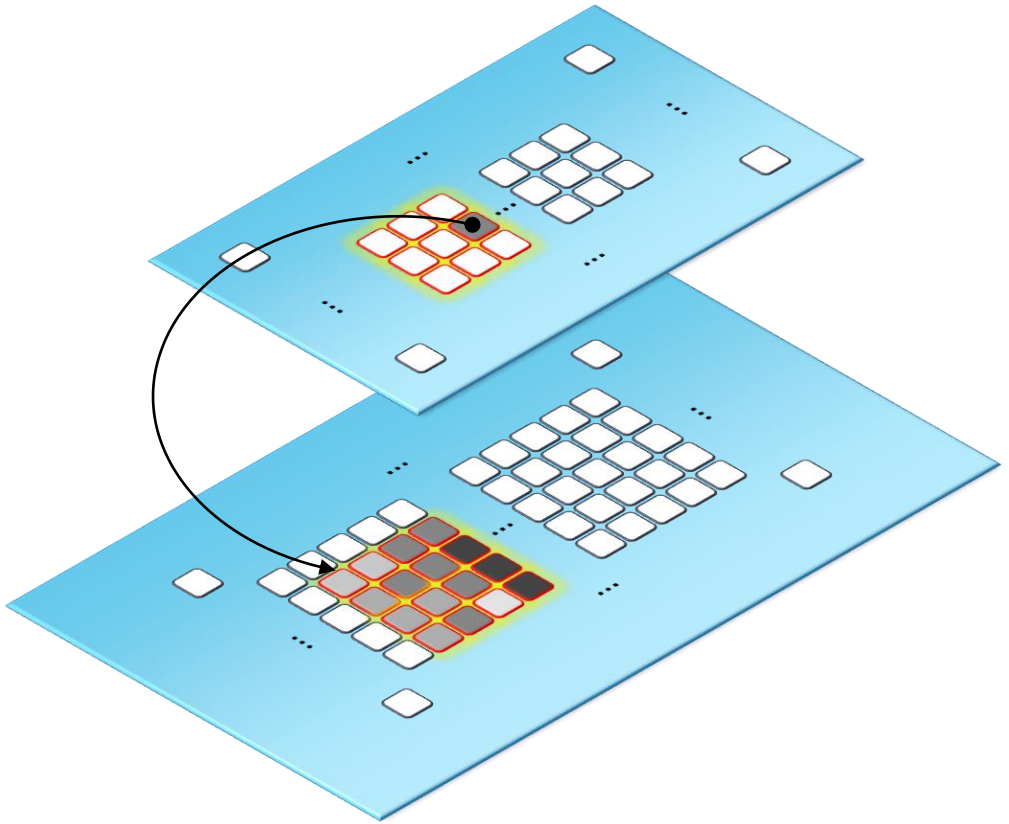


Multiresolution multilevel page table hierarchy



Brick cache

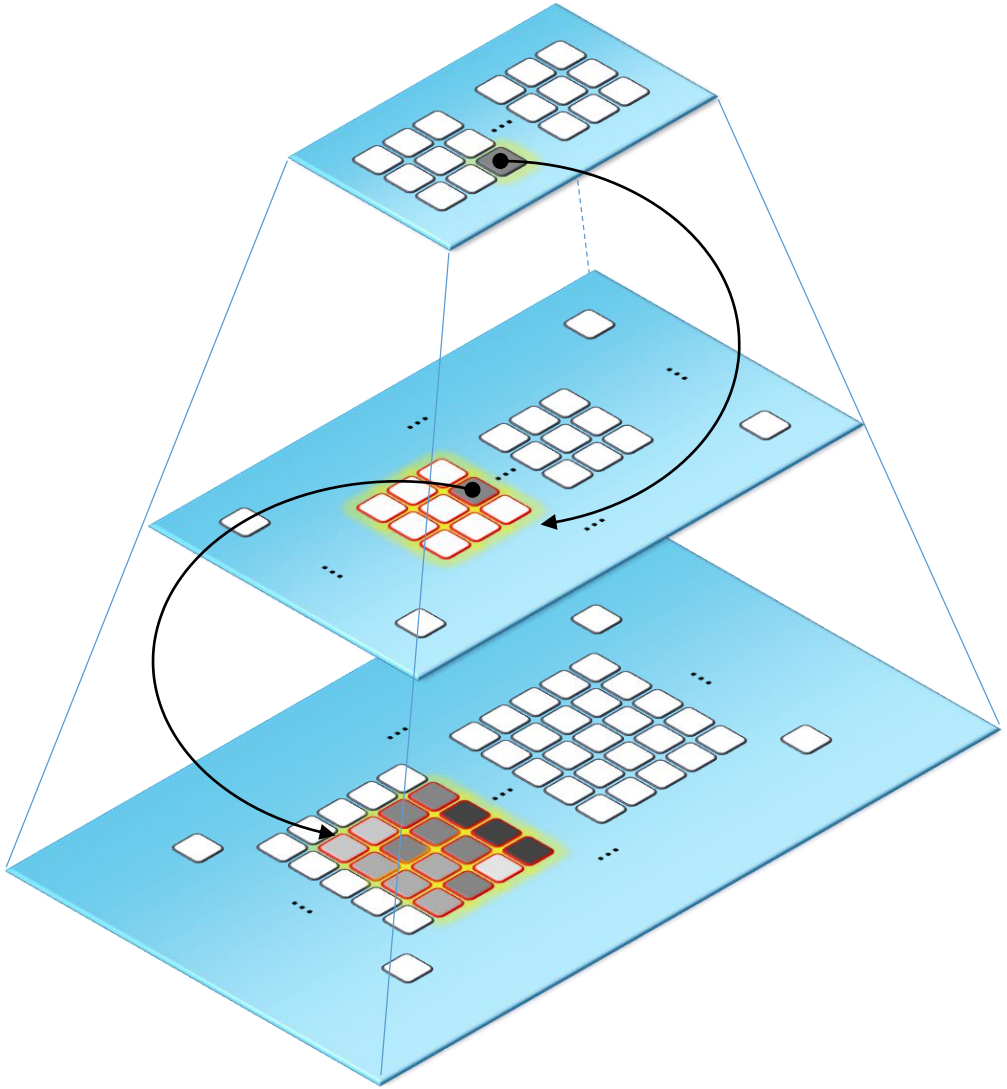
Multiresolution multilevel page table hierarchy



Multiresolution page table

Brick cache

Multiresolution multilevel page table hierarchy



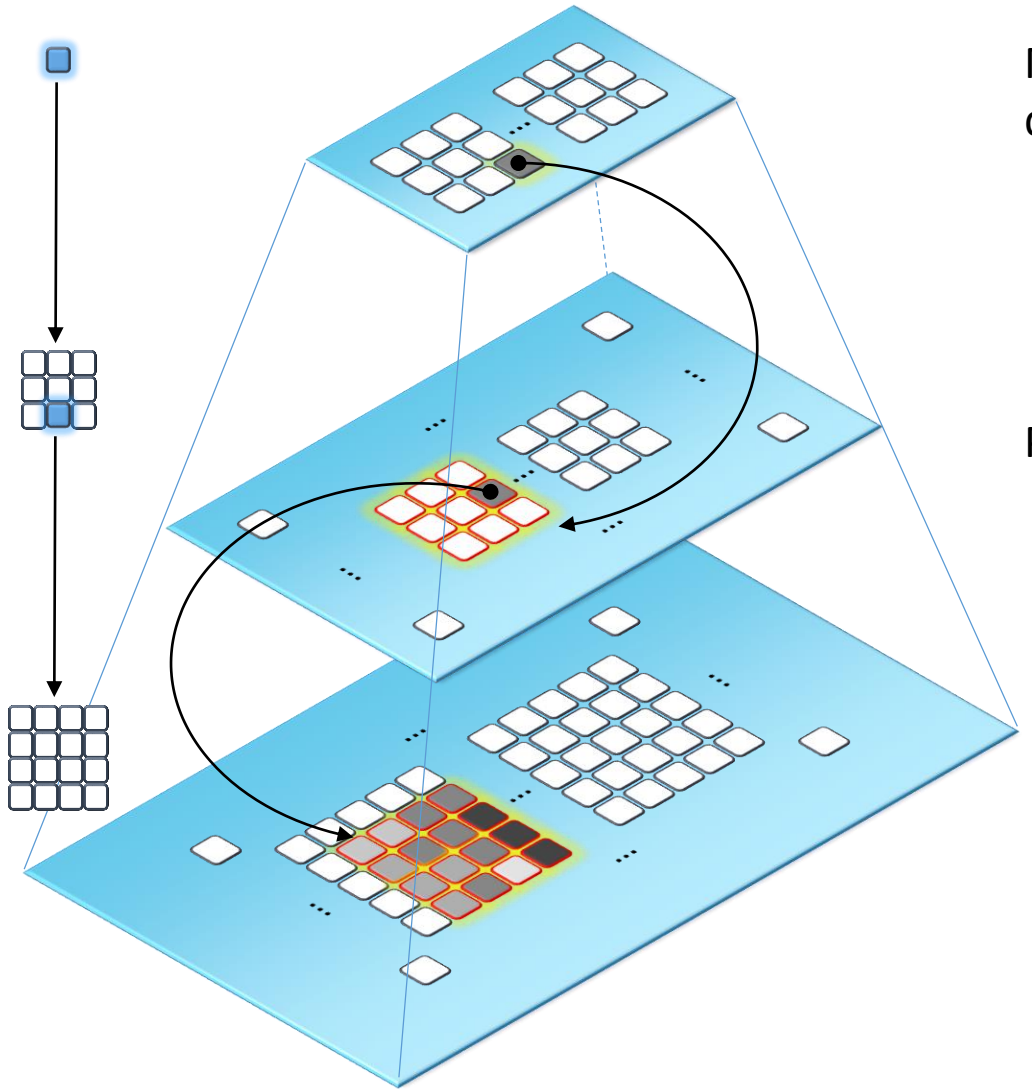
Multiresolution page directory

Page table cache

Brick cache

Multiresolution multilevel page table hierarchy

- Entry =
  - 3D coordinates of the block in the next cache
  - + Flag:
    - Mapped
    - Unmapped
    - Empty



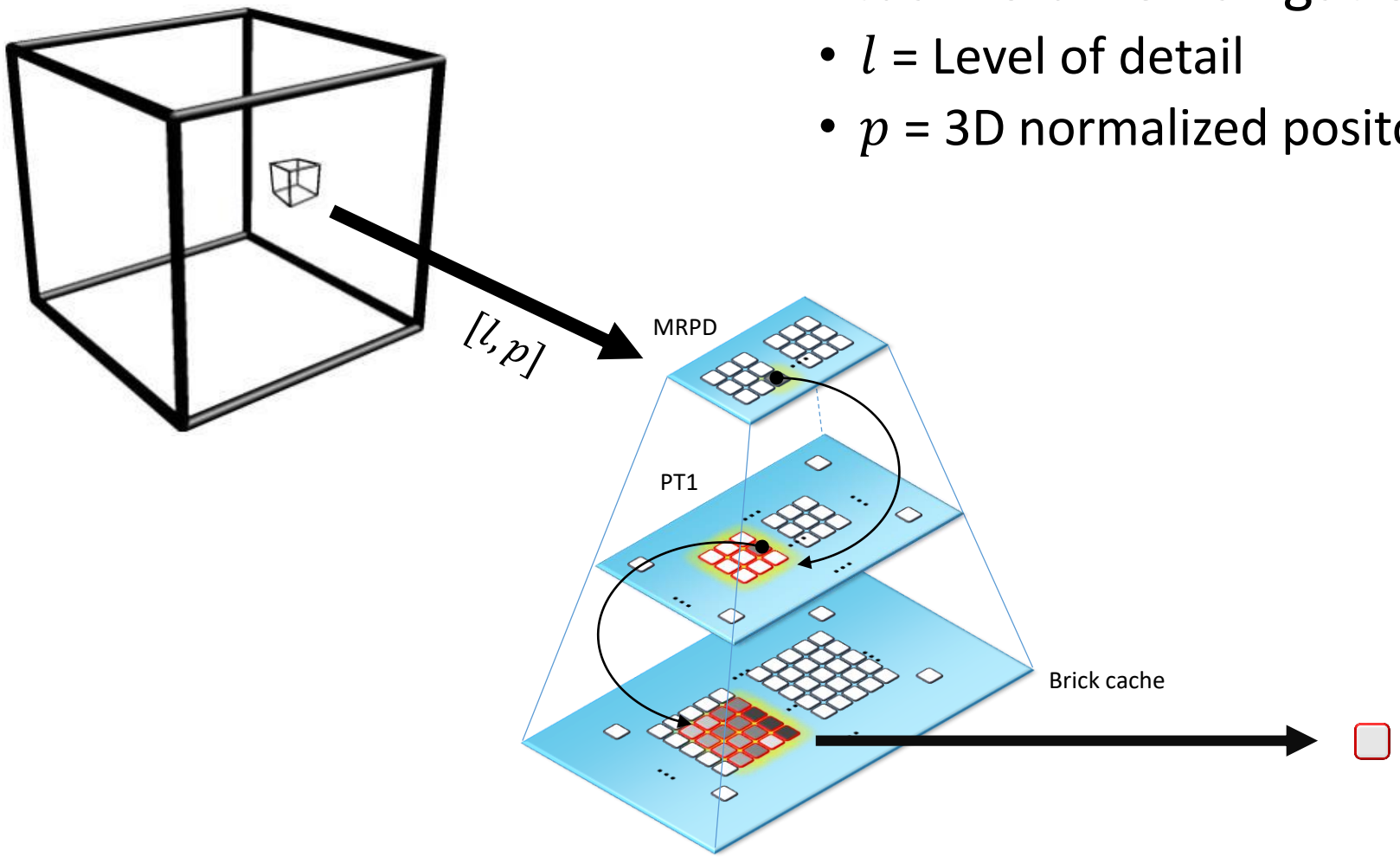
Multiresolution page directory

Page table cache

Brick cache

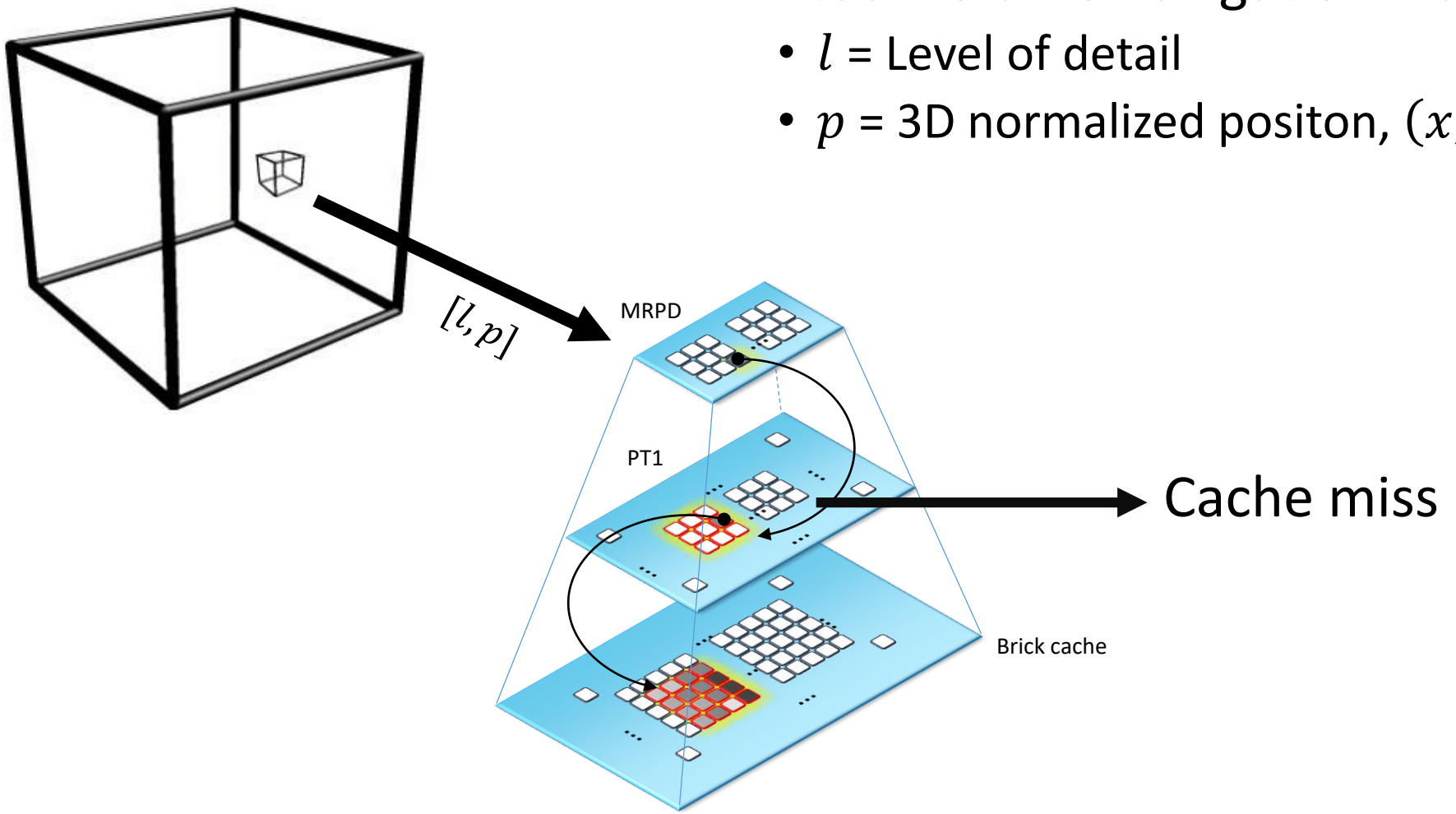
Virtual addressing

- Virtual volume navigation – address = $[l, p]$
  - l = Level of detail
  - p = 3D normalized positon, $(x, y, z) \in [0, 1)^3$



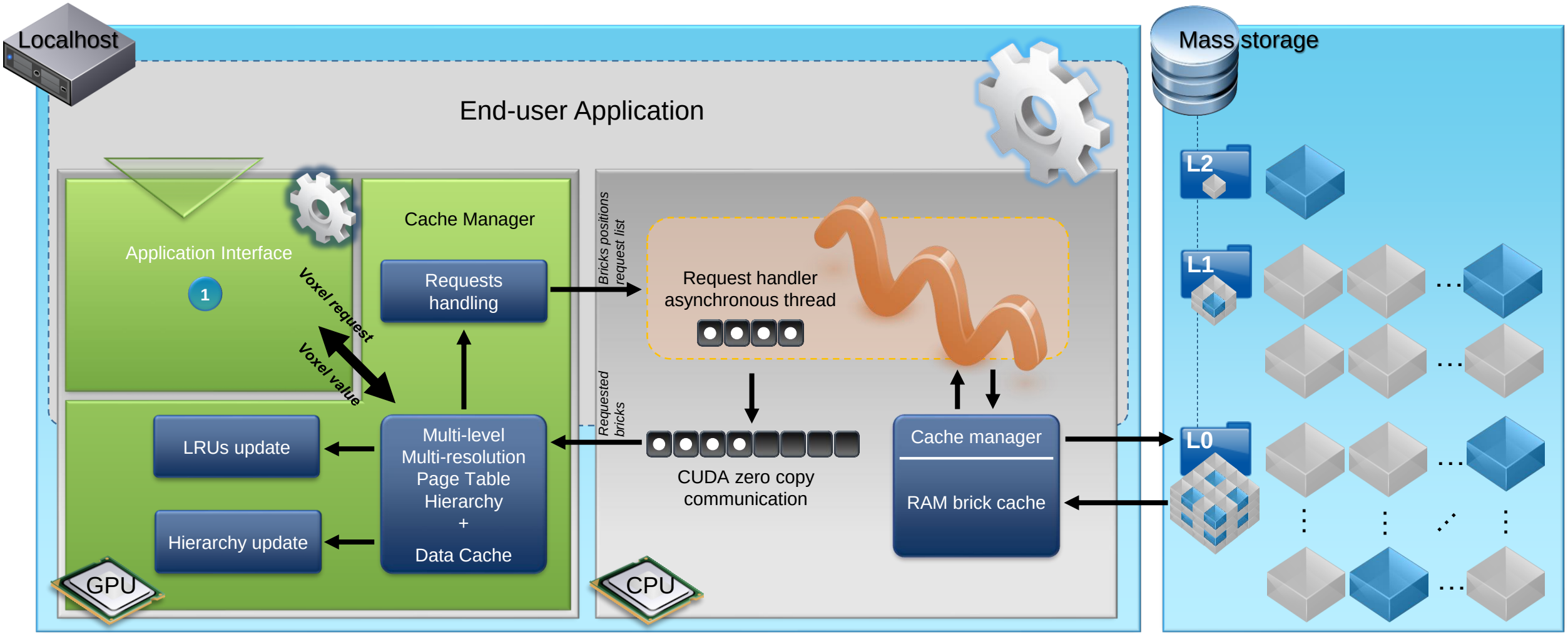
Cache miss

- Virtual volume navigation – address = $[l, p]$
  - l = Level of detail
  - p = 3D normalized positon, $(x, y, z) \in [0, 1)^3$



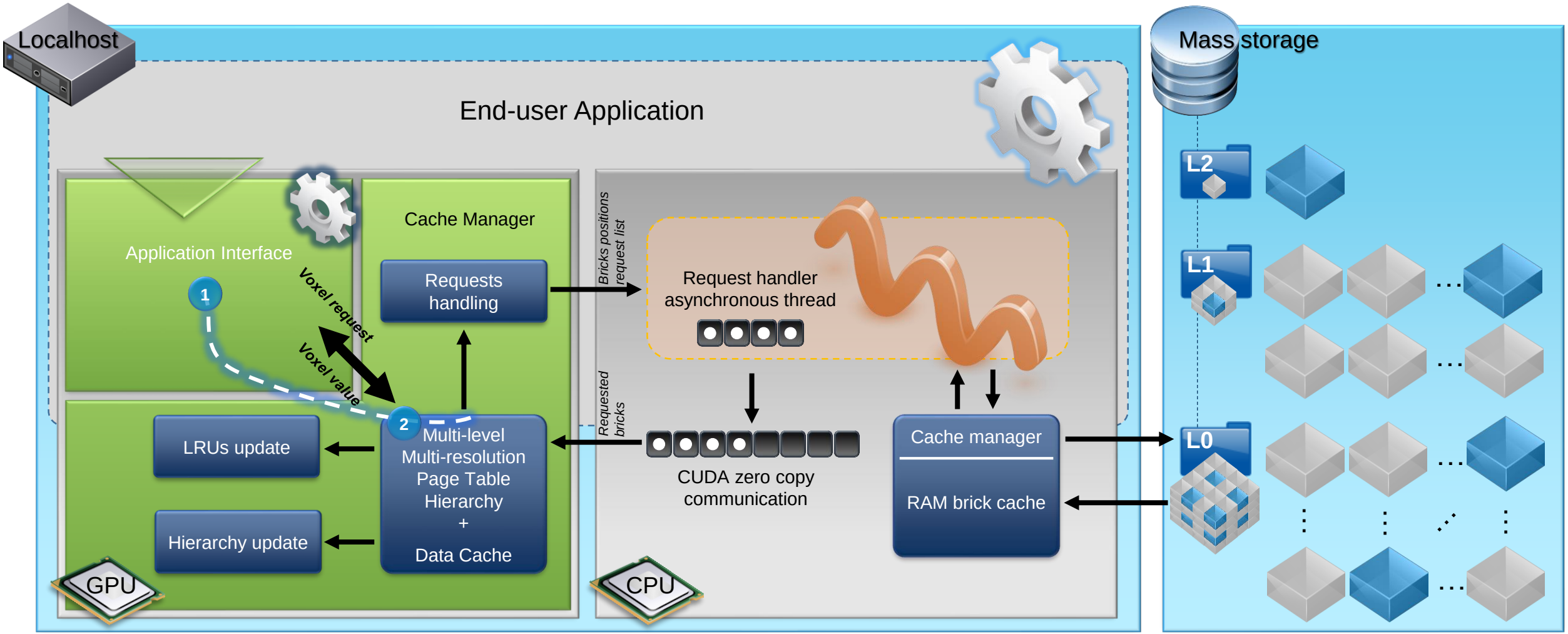
Pipeline

1 – Voxel cache request



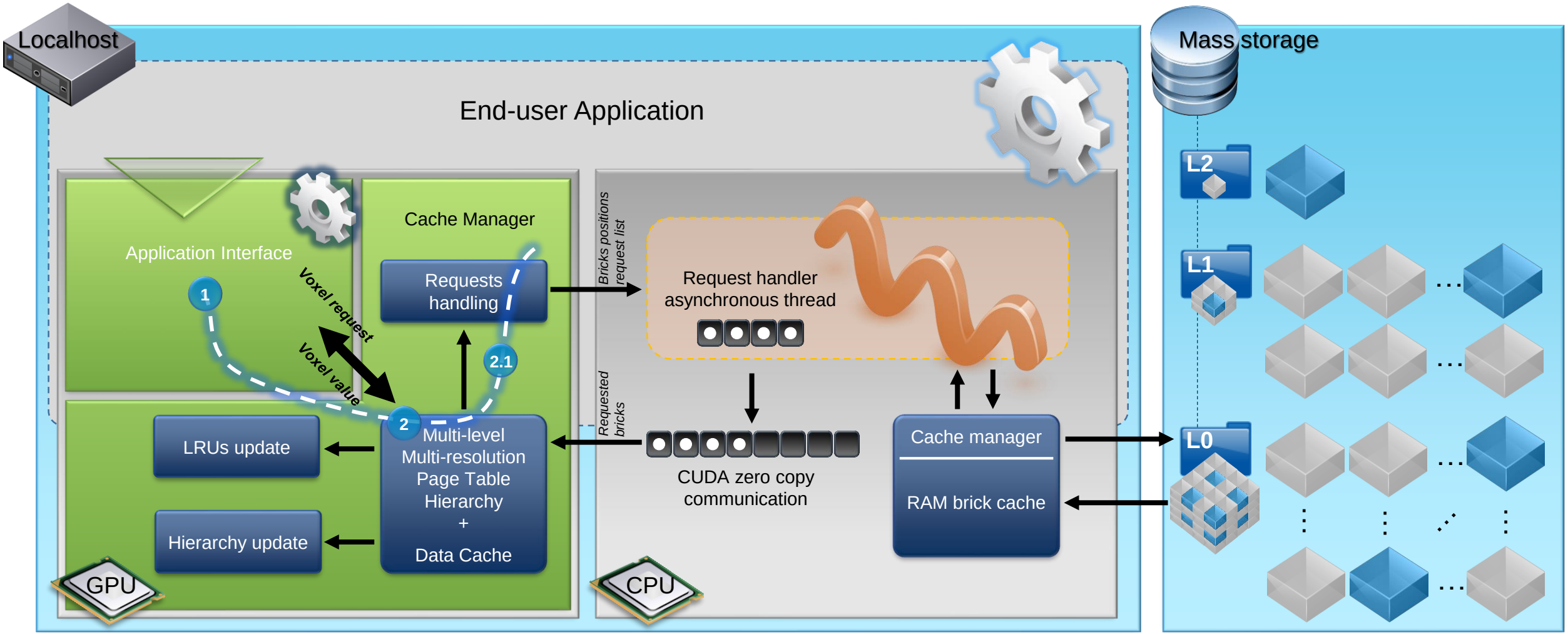
Pipeline

2 – Hierarchy look-up



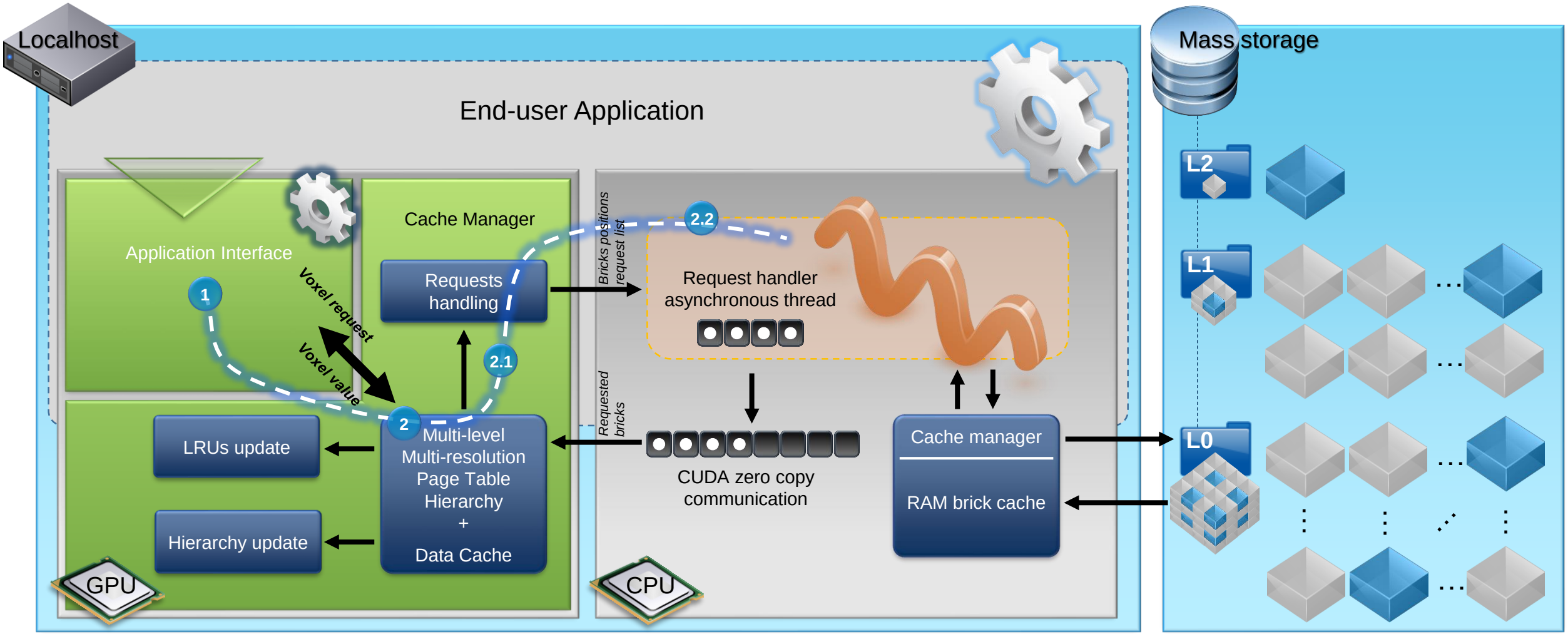
Pipeline

2.1 – Request list creation



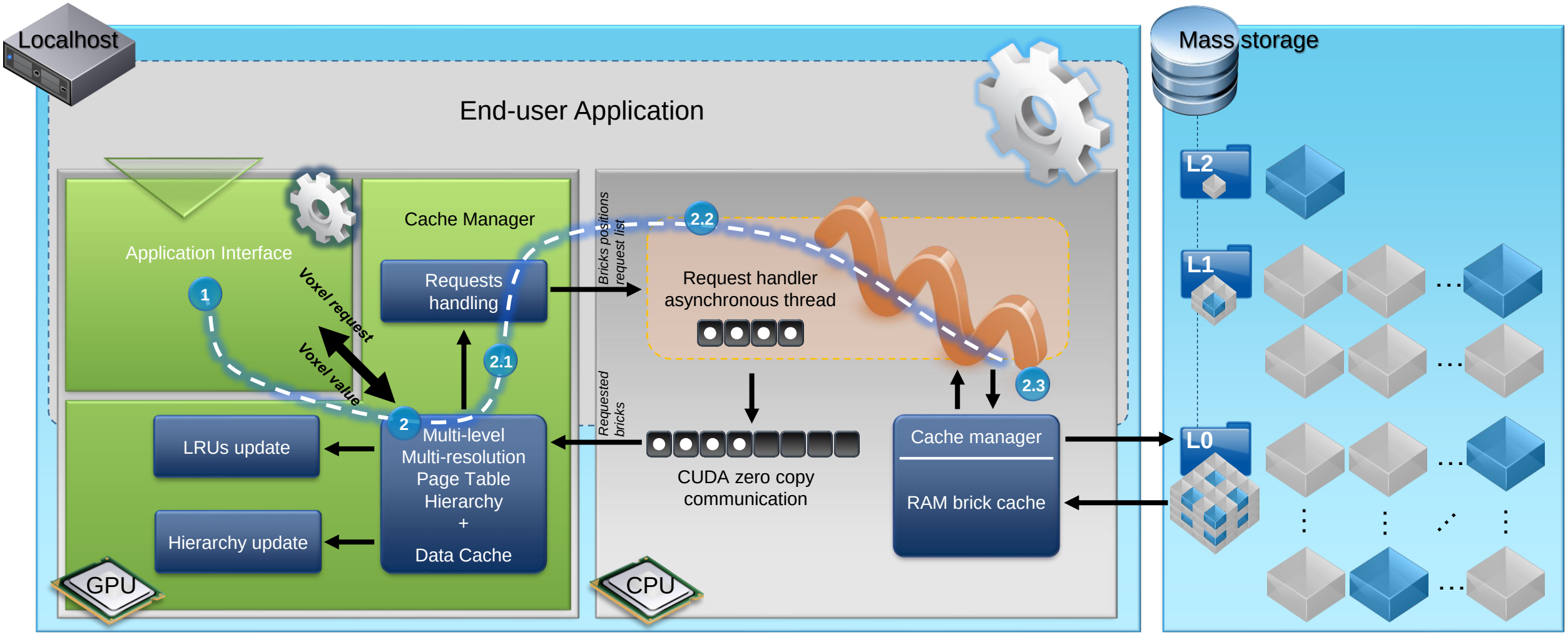
Pipeline

2.2 – Request list asynchronous handling



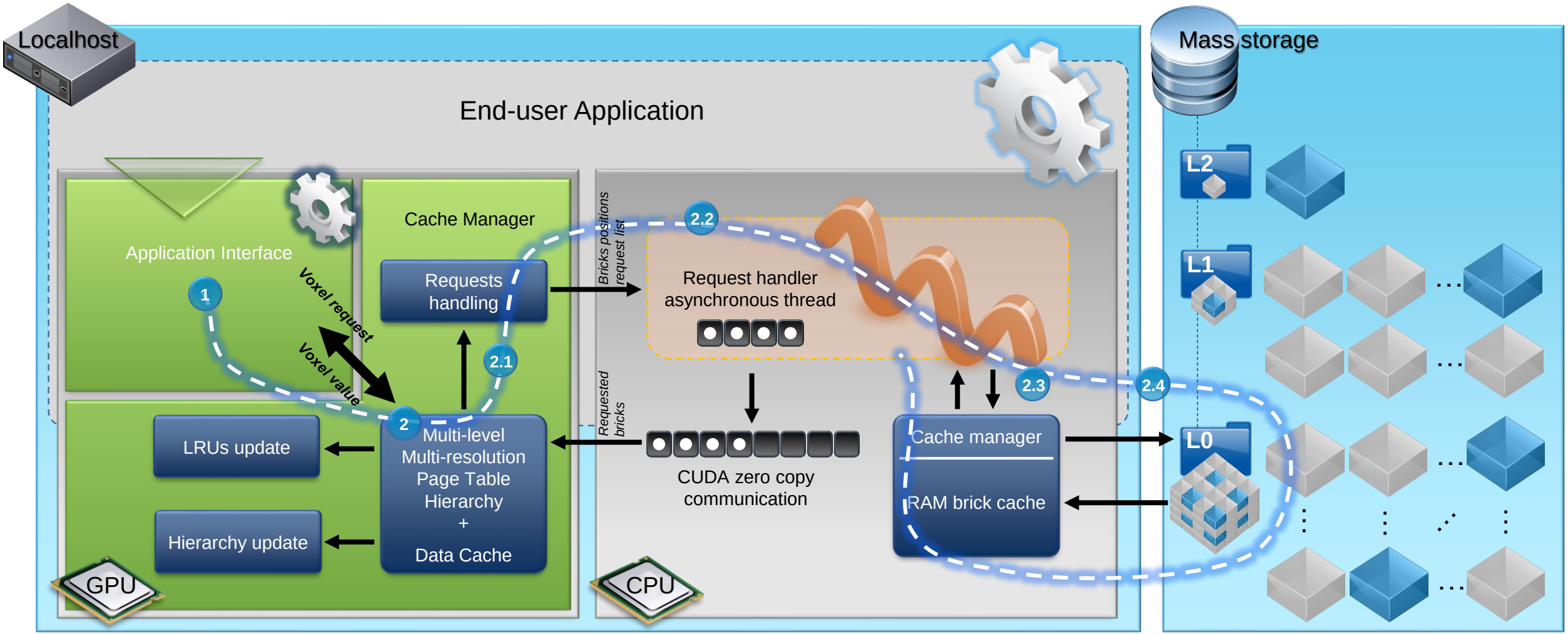
Pipeline

2.3 – CPU cache look-up (simple cache)



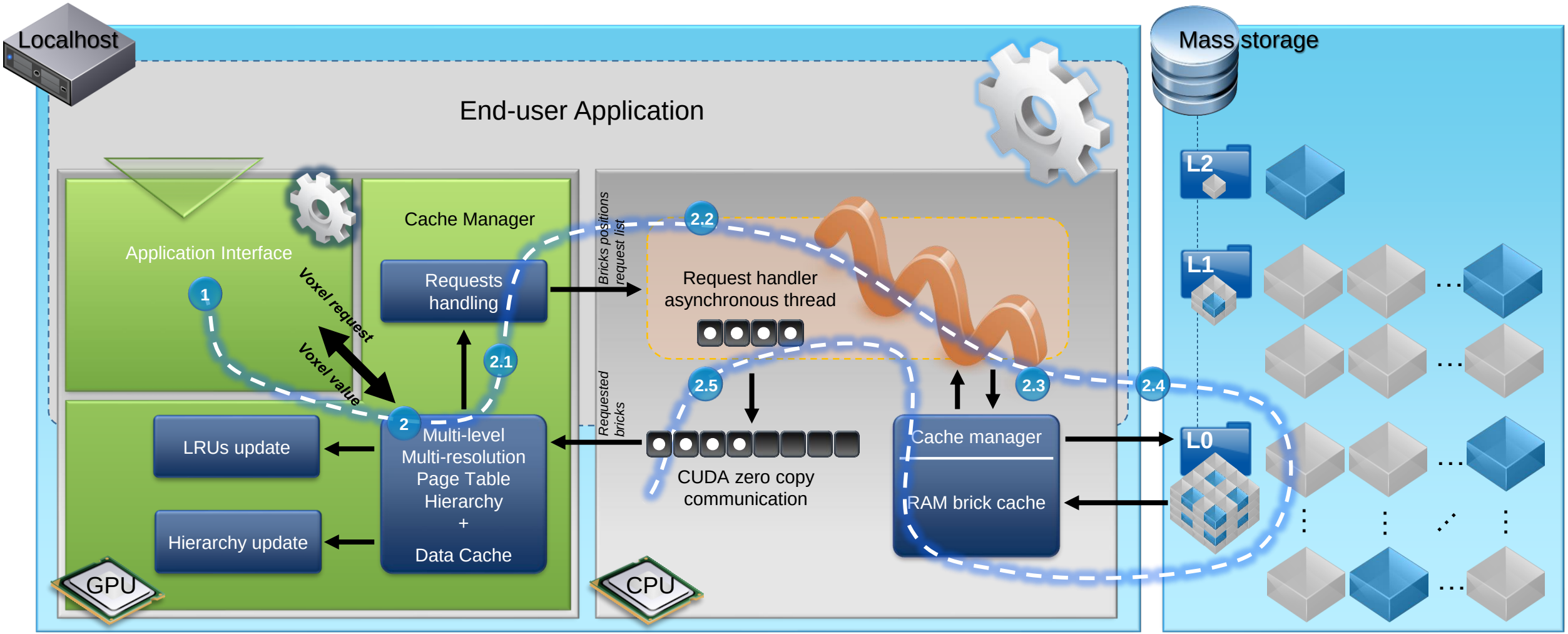
Pipeline

2.4 – If not in CPU cache = Loading from mass storage



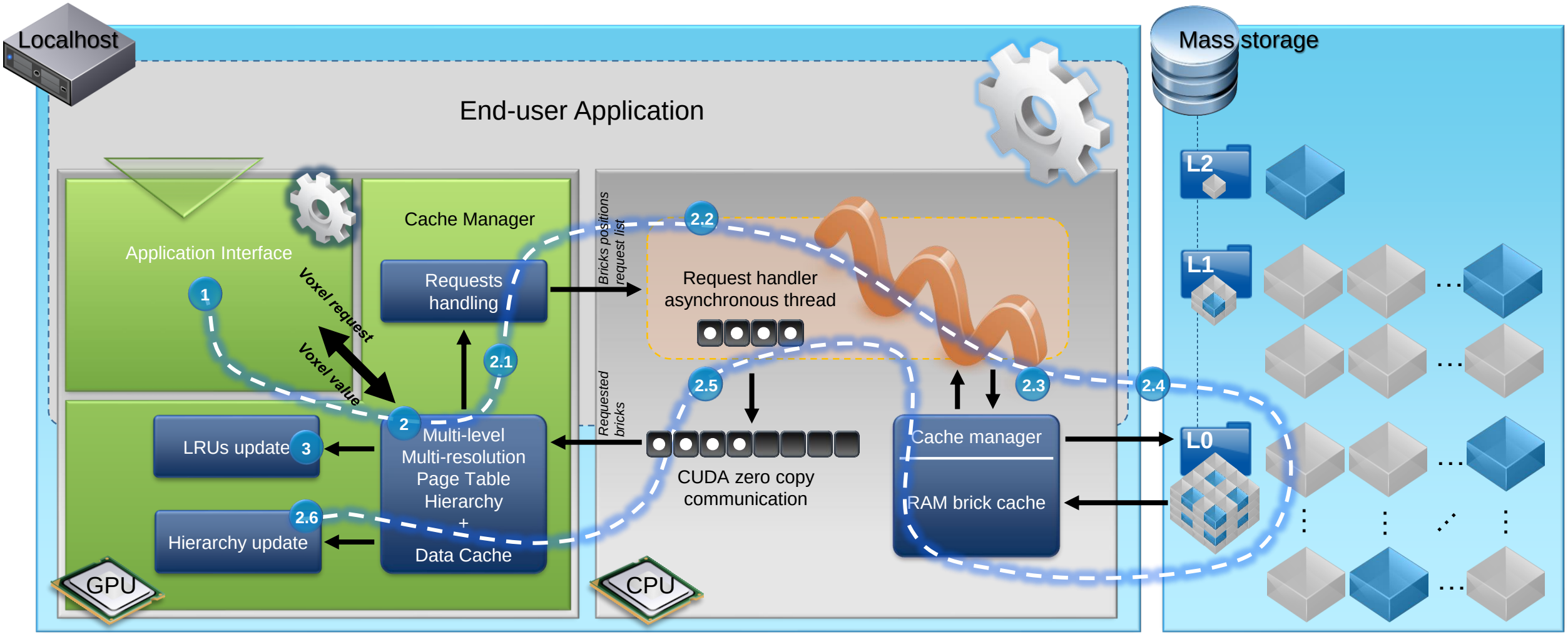
Pipeline

2.5 – Load bricks in a Cuda zero copy buffer

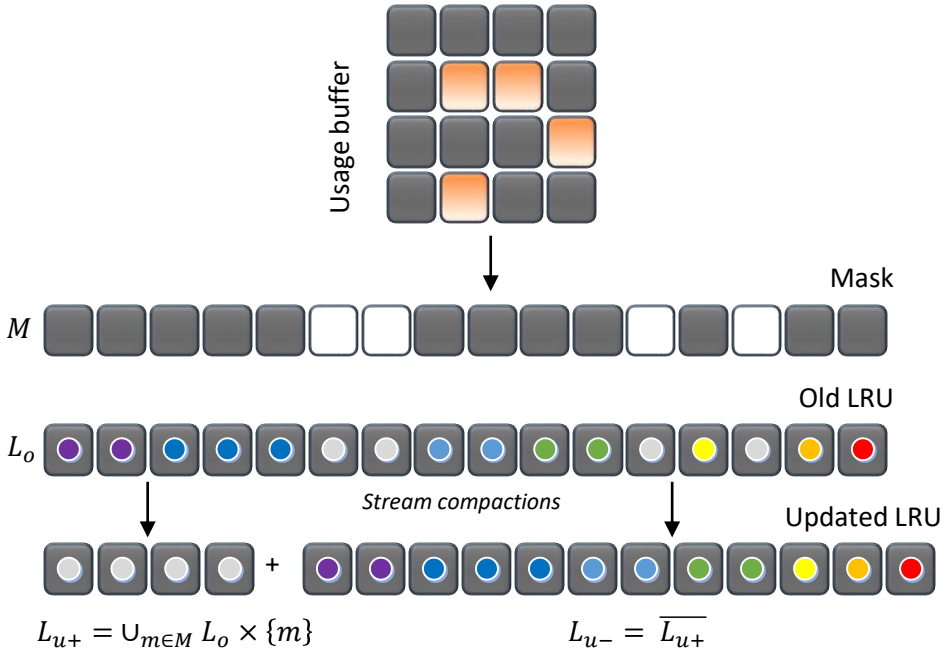


Pipeline

3 & 2.6 – LRUs update then address bricks in GPU cache hierarchy

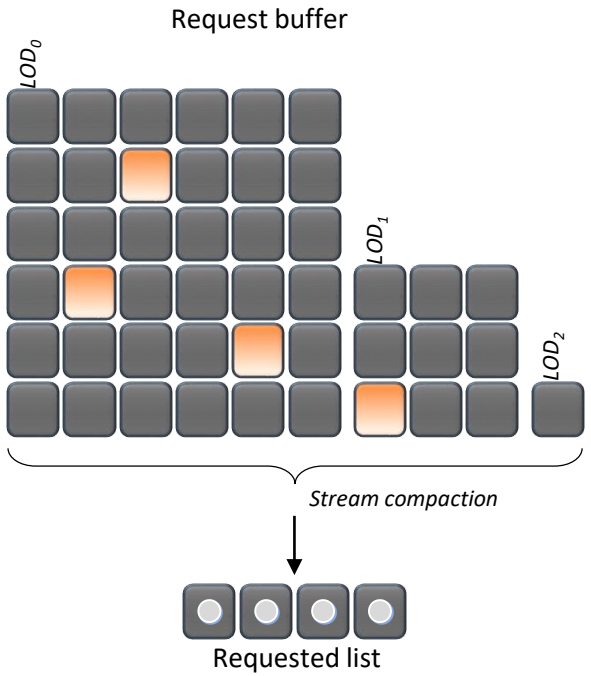


LRU updates on GPU



- Size = number of bricks in the cache
- Marked with a timestamp

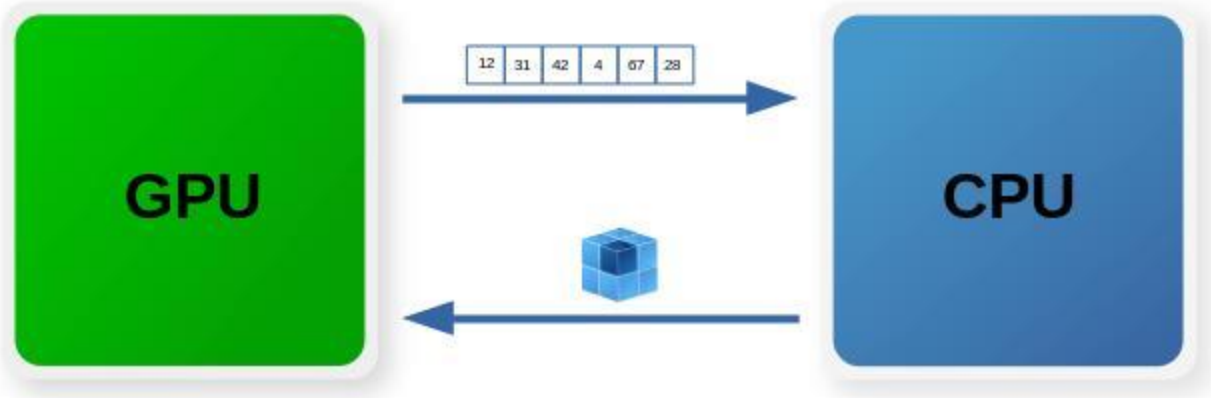
Brick request management on GPU



- Size = number of bricks in the volumes (All LODs)
- Marked with a timestamp

CPU / GPU transfer

GPU to CPU – Requested bricks IDs

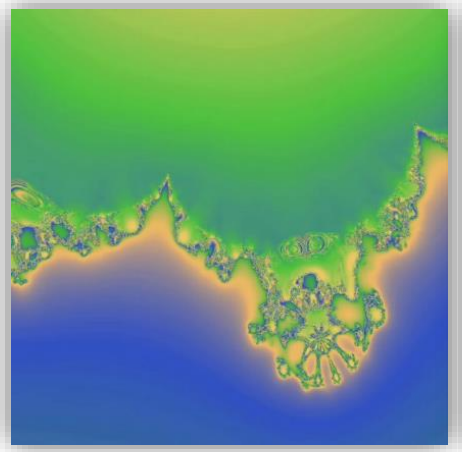


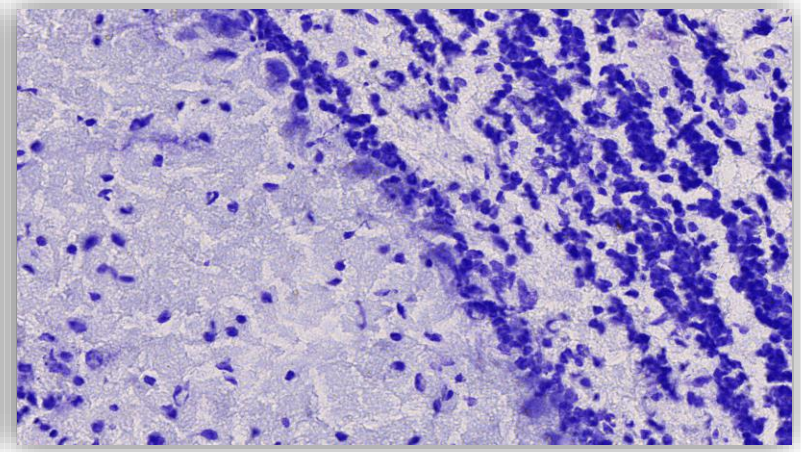
CPU to GPU – The data (bricks) [Cuda Zero copy]

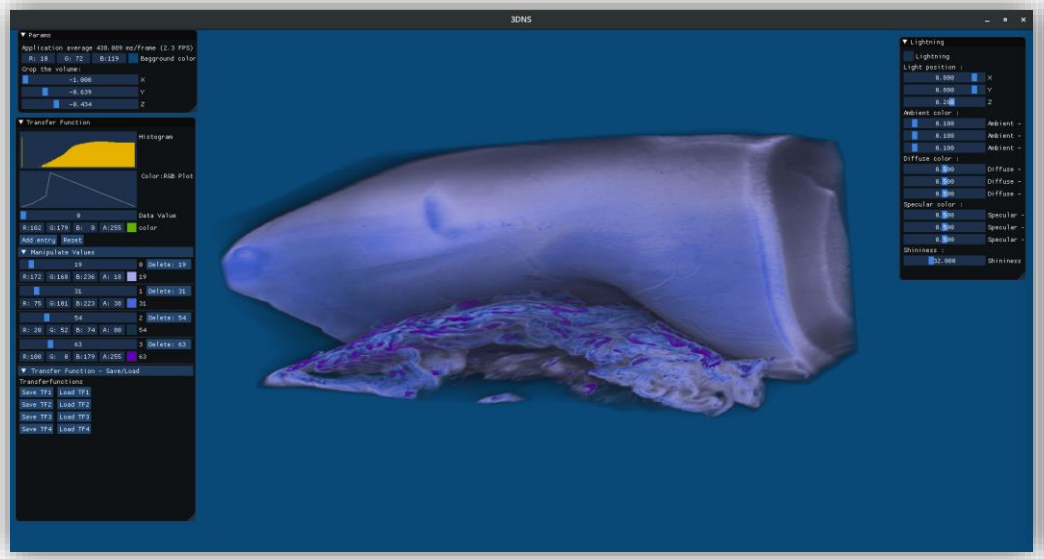
Model in action: application to visualization

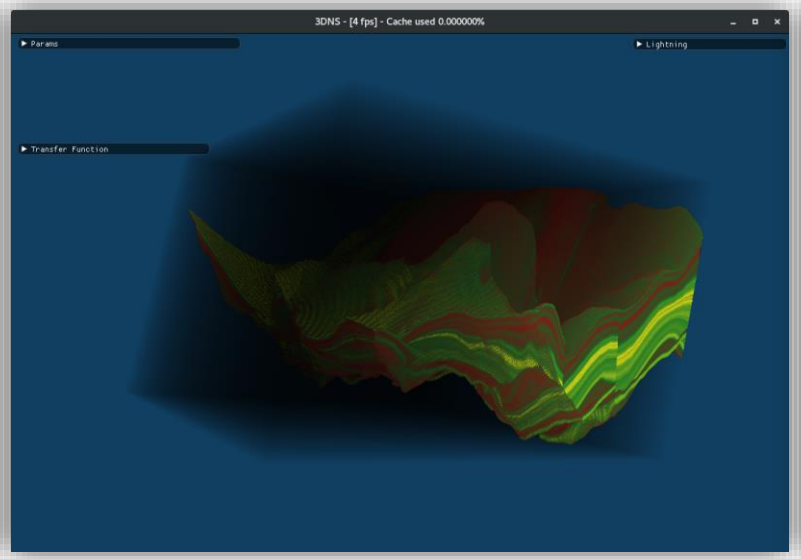
General purpose applications

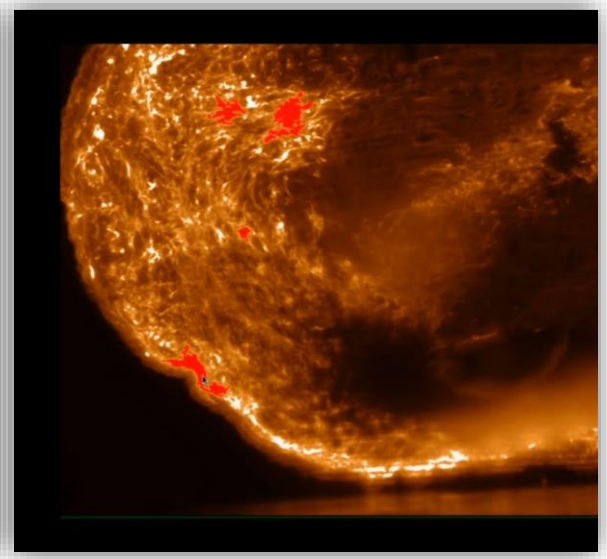
- Processing
  - Convolution, classification, etc.
- Visualization
  - Virtual microscopy
  - Direct volume rendering





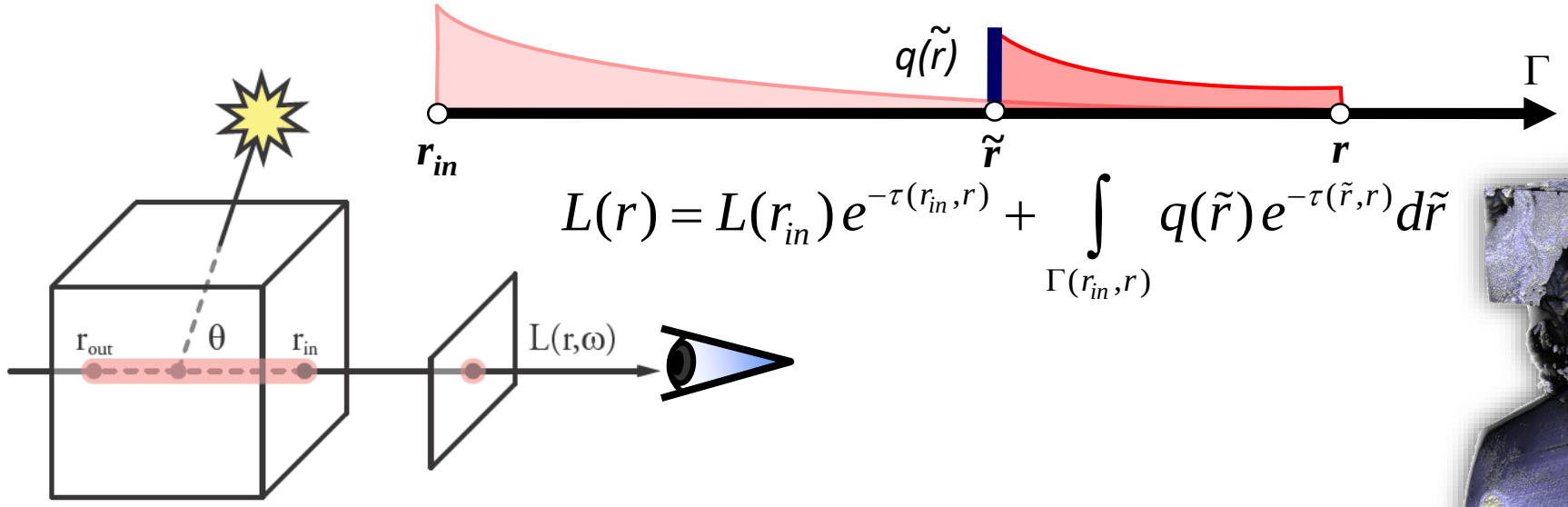


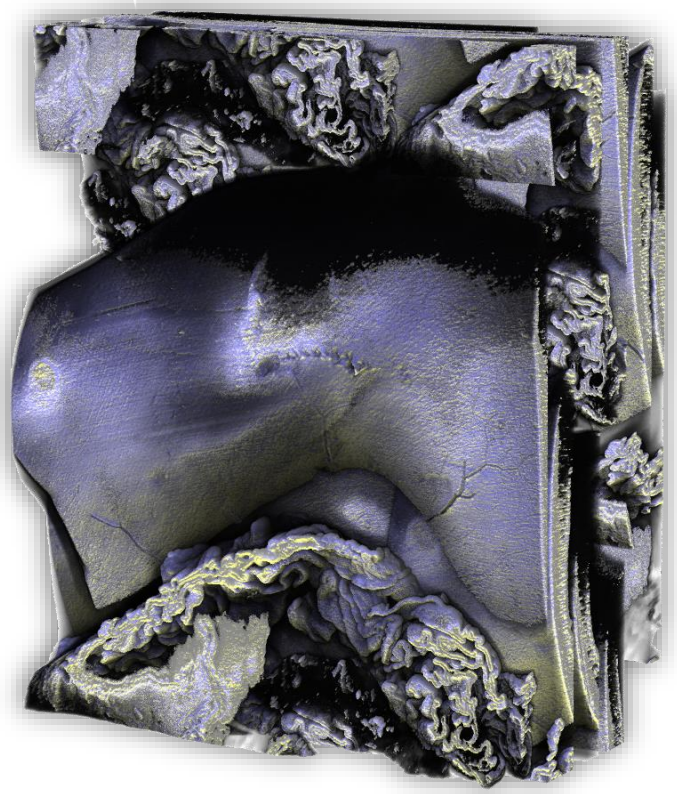




Multi-resolution volume Ray-Casting

- Cast primary rays to integrate the volume rendering equation. Color and opacity samples accumulation according to transfer function

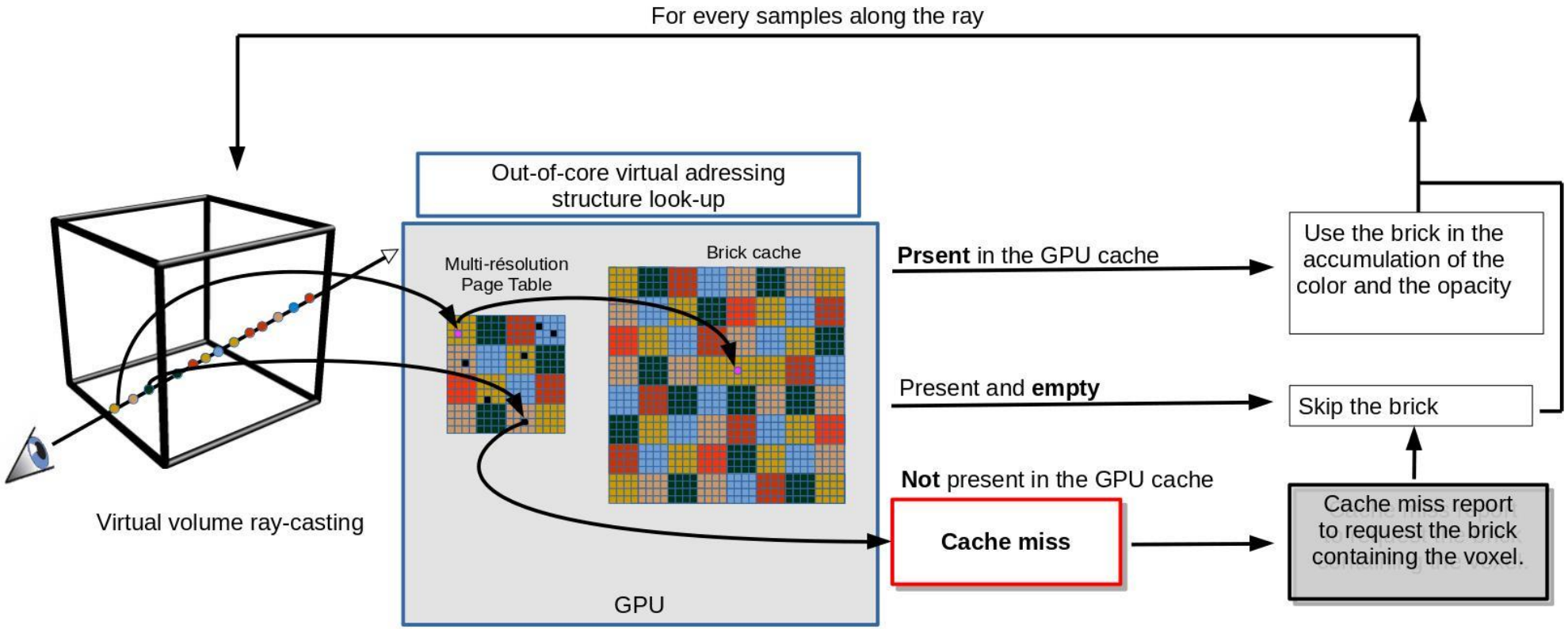




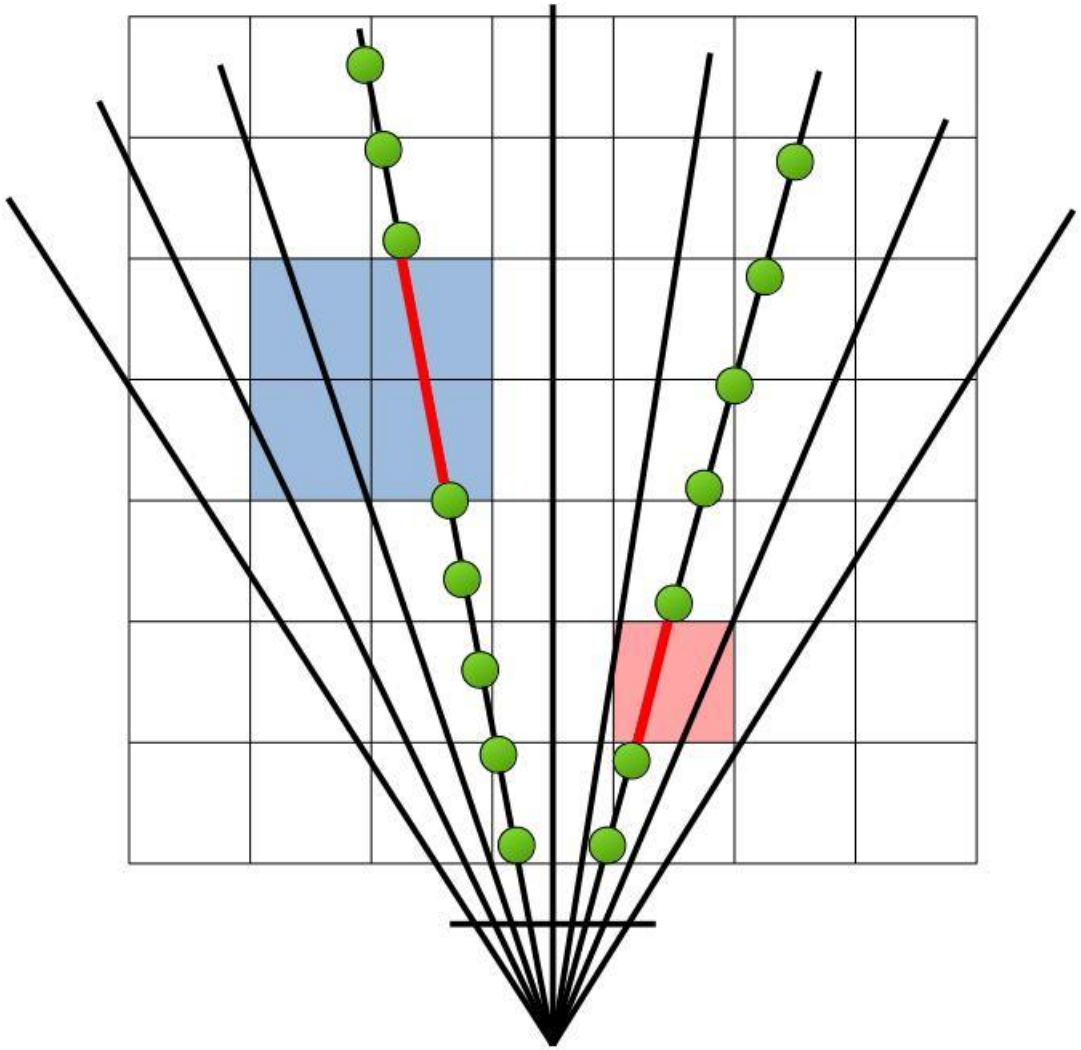
- Multiresolution
  - LOD selection for each samples during ray-marching
  - Adaptive sampling according to the multiresolution representation

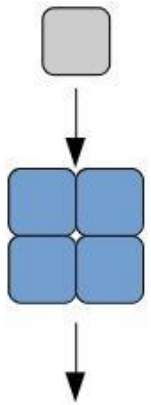
Ray-guided approach

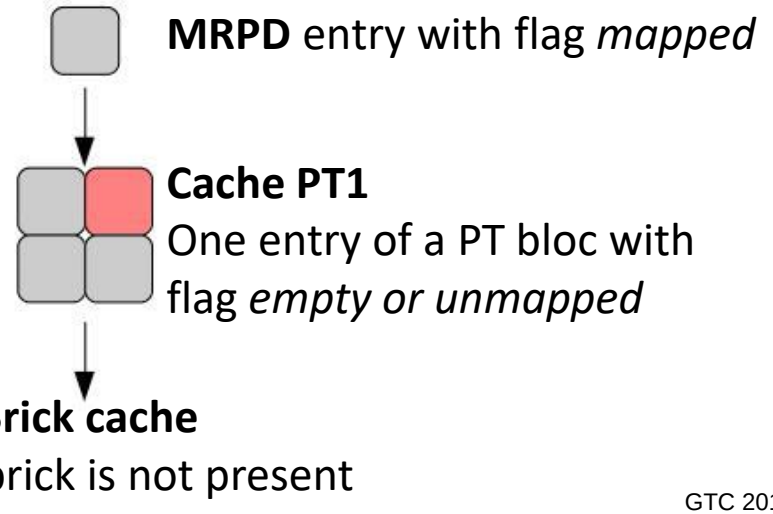
- Modern approach for out-of-core GPU based volume ray-casting on large data
  - Intuitive visibility selection : no additional culling calculation
  - Intuitive out-of-core integration : only load visible bricks on GPU cache.



Space skipping

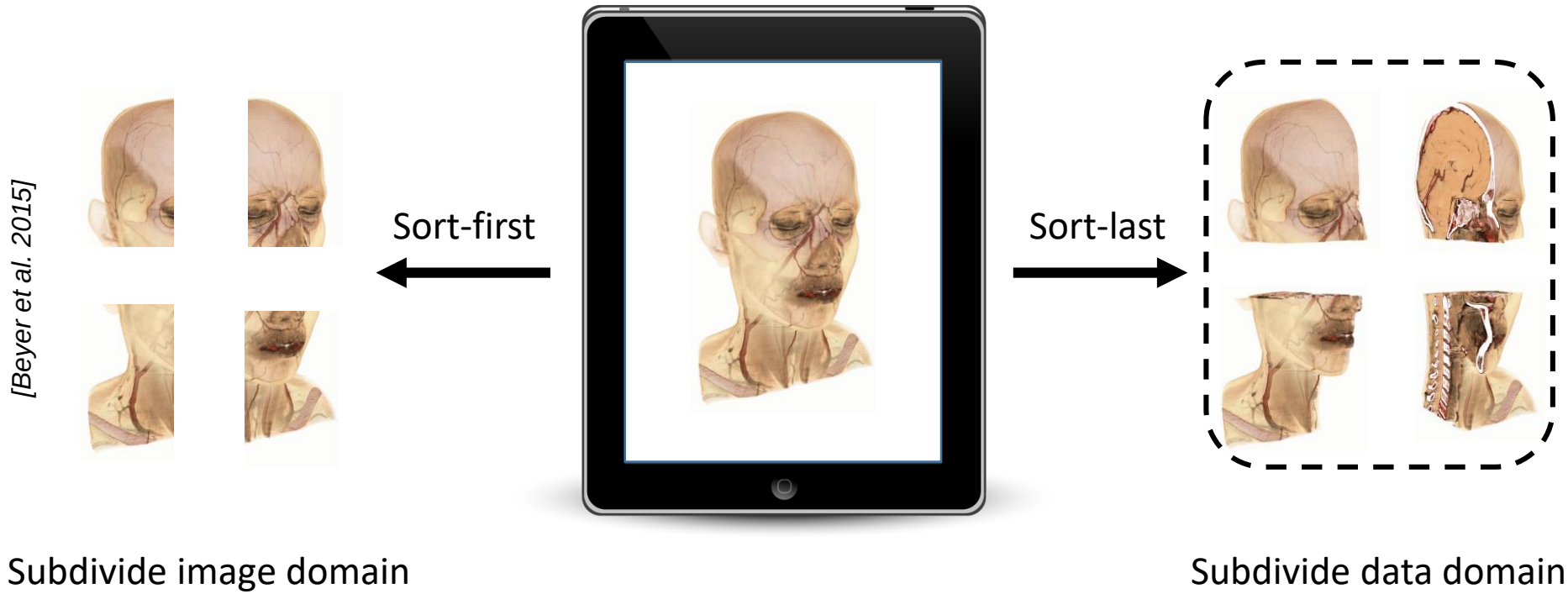






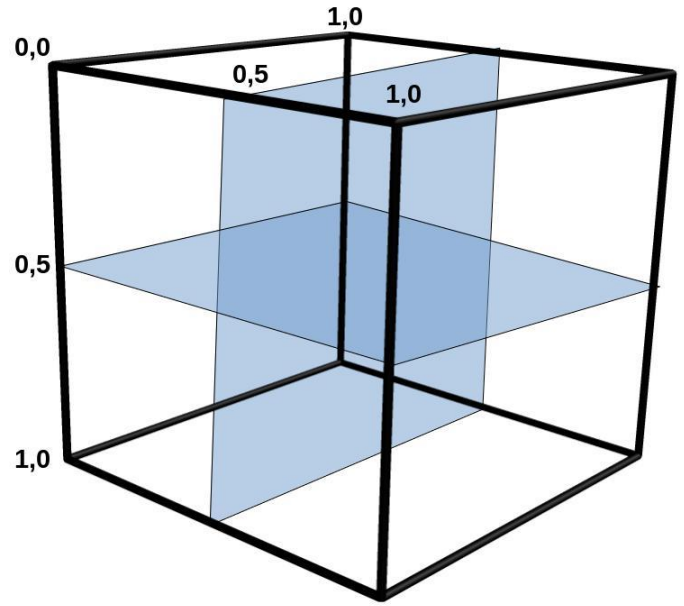
Multi-GPU rendering strategy

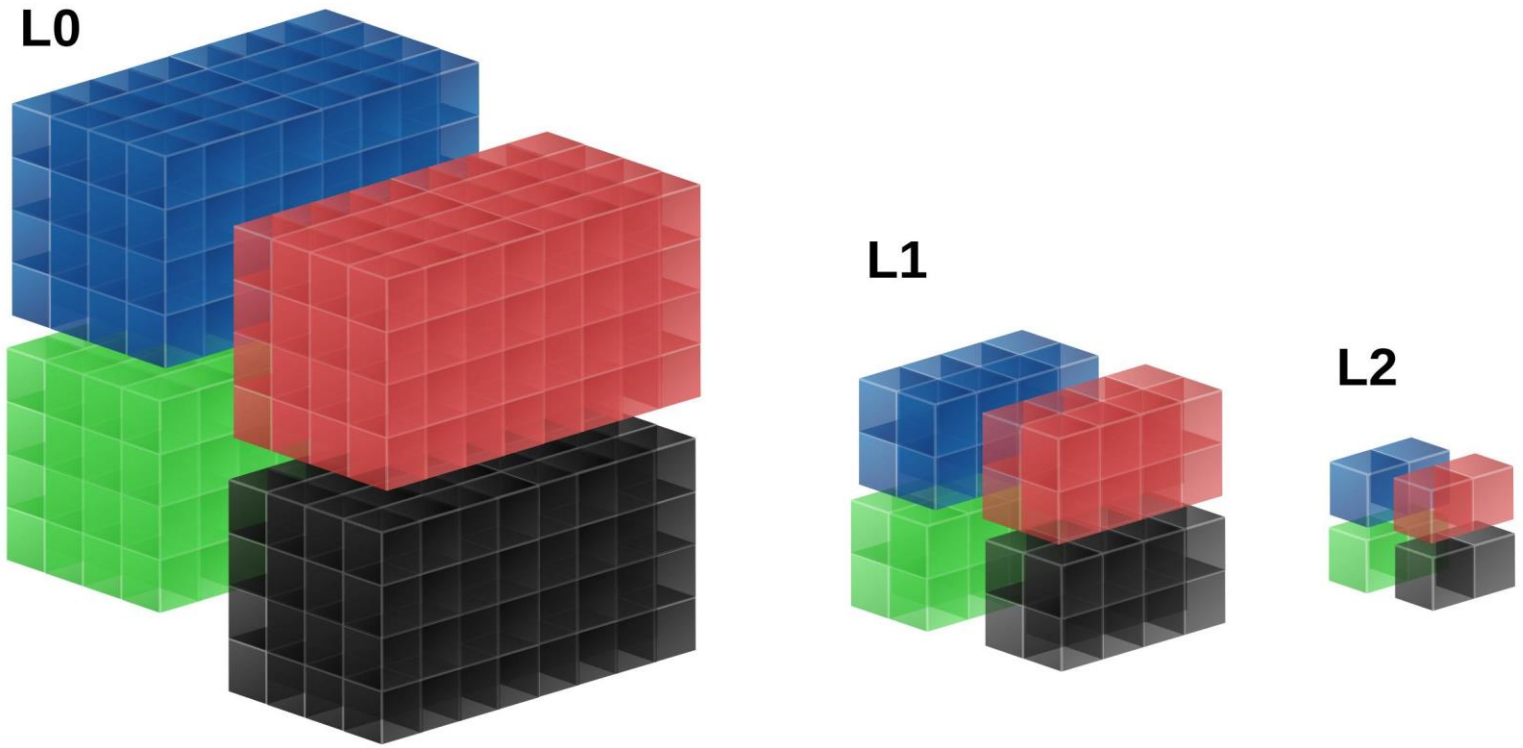
- **Sort-last** distributed approach
- Compute scalability + **memory scalability**
- Compositing : reconstruct final rendering from local GPU contribution



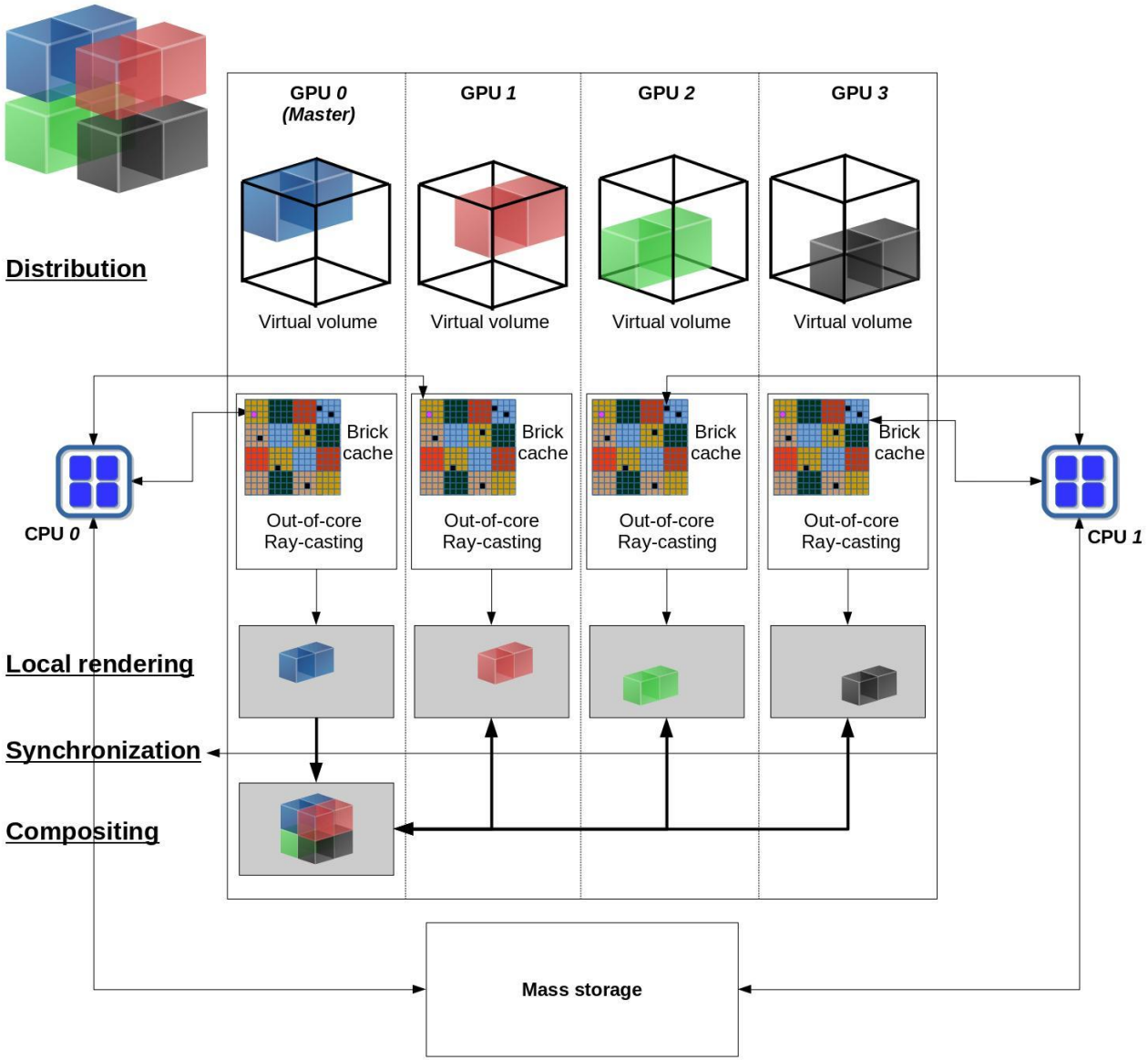
Volume distribution

- Virtual volume distribution: restrict virtual adressability range of each GPU
- Implicit multiresolution volume distribution





Distributed out-of-core volume rendering



GPU Composition

- Peer to peer GPUs communication with UVA if possible
- Intermediate CPU transfers otherwise

- Front-to-back over operator

$$C'_i = C'_{i-1} + \left(1 - \propto'_{i-1}\right) C_i$$
$$\propto'_i = \propto'_{i-1} + \left(1 - \propto'_{i-1}\right) \propto_i$$

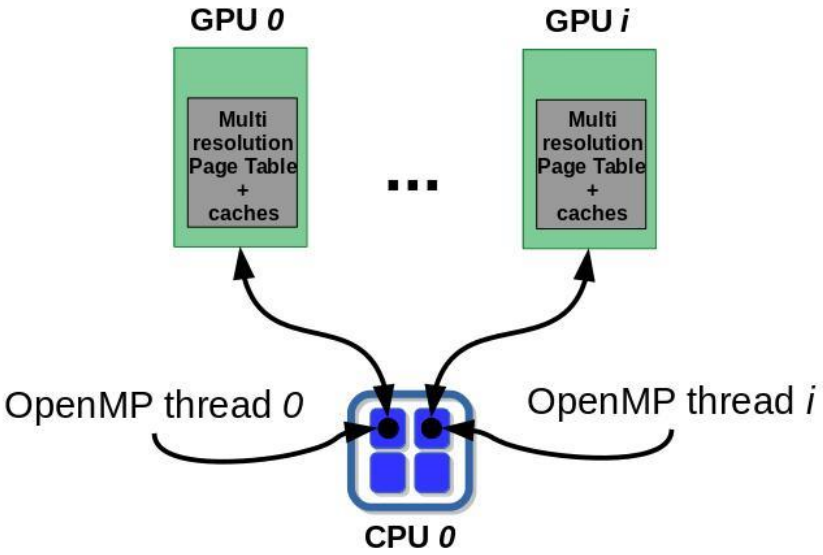
With C: RGB color and $\propto$: opacity

Out-of-core distribution & communications

- One instance of the virtual addressing structure and its cache manager by GPU
- A single virtualization configuration (# of level, brick size and PT block sizes)
- A single CPU cache

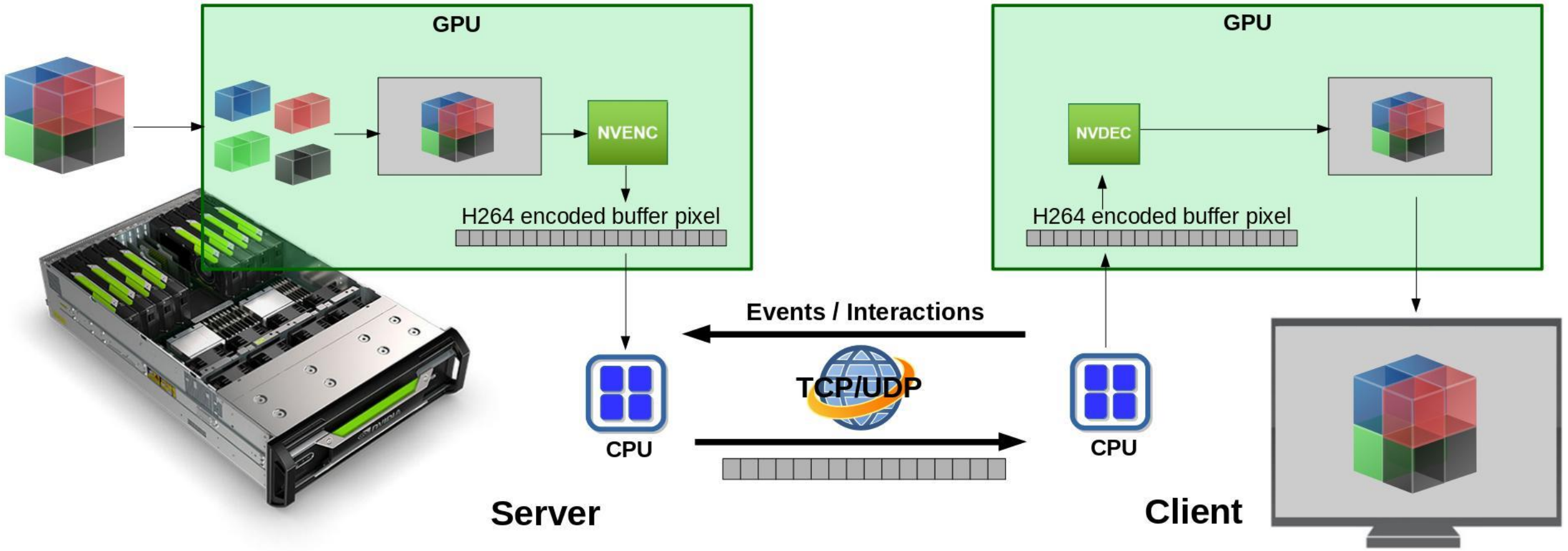
Strategy multi-thread and communications

- One openMP thread by CUDA context (by GPU)
- Attributed on one core of the associated CPU



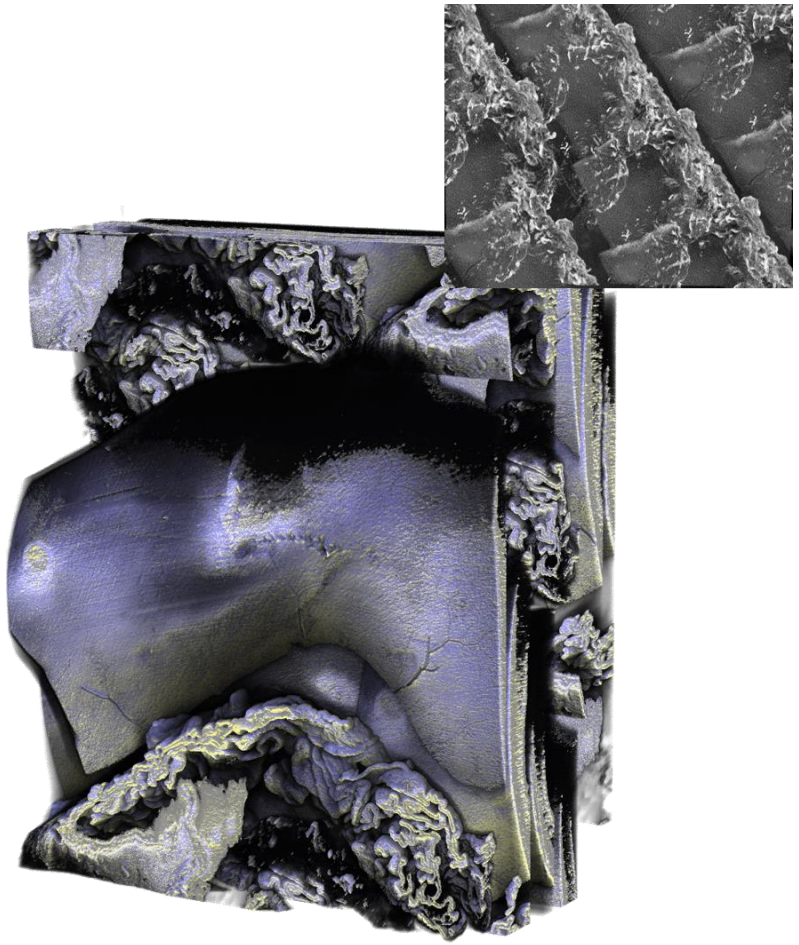
Remote rendering

- Interactive display streaming from remote server rendering

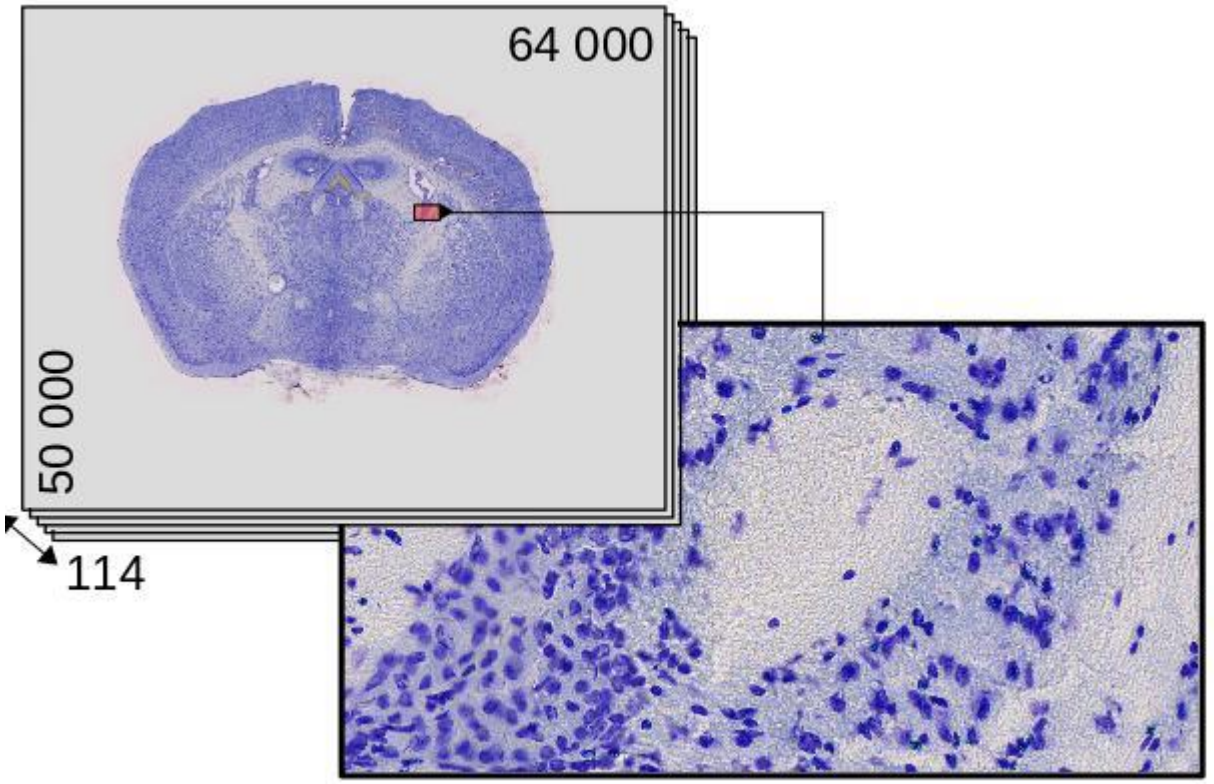


Evaluation and results

Datasets



Light sheet microscope primate hippocampus
2160 x 2560 x 1072 16 bits ≈ **12 GB**



Histological mouse brain
64000 x 50000 x 114 RGBA ≈ **1,5 TB**

Memory occupancy

- "light sheet“ microscopy 2 160 × 2 560 × 1 072 ≈ 12 GB
  - Brick size 64^3 → ≈ 27 000 bricks (7 LOD)
  - One virtualization level (data cache = 7000 bricks)
    → **1.2 MB** on GPU needed

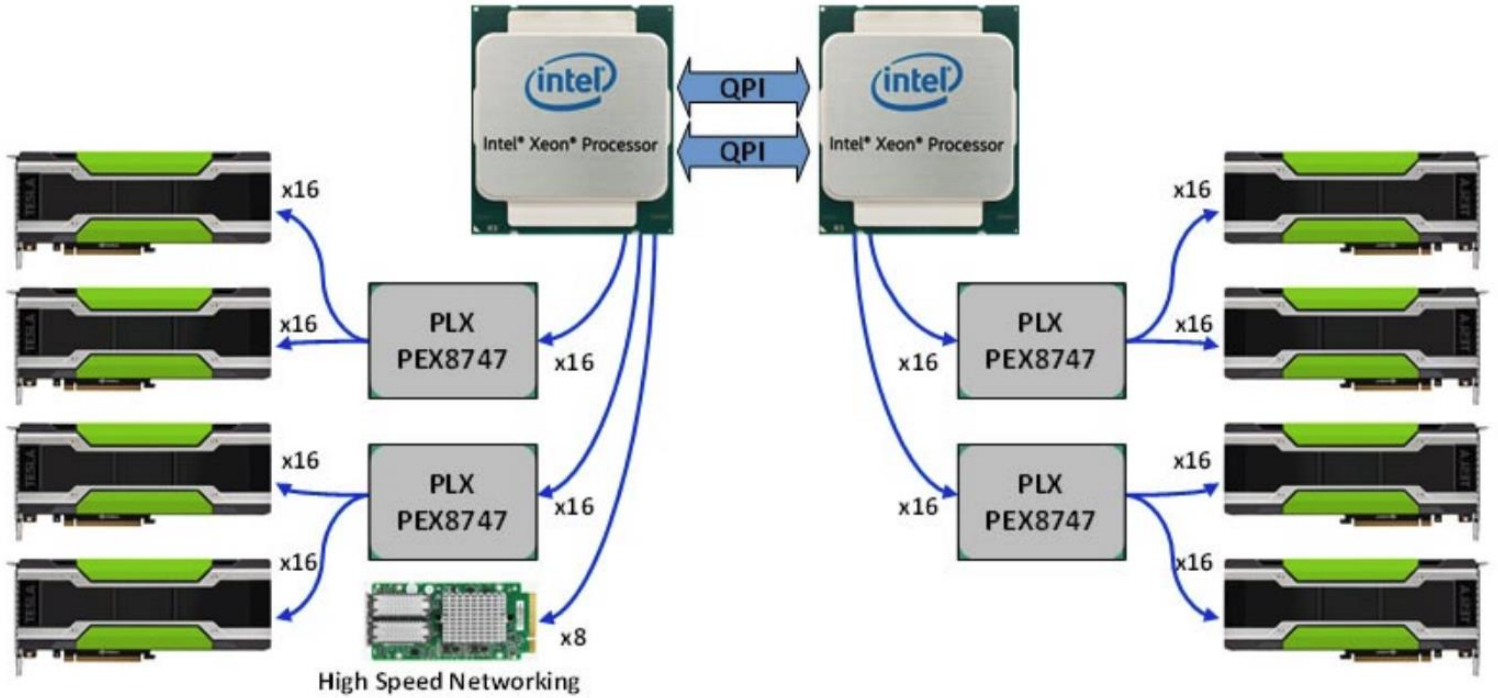
- Mouse brain 64 000 x 50 000 x 114 ≈ 1.5 TB
  - Brick size 64^3 → ≈ 3.13 million bricks (10 LOD)
  - One virtualization level → ≈ **63 MB** on GPU needed
  - Two virtualization levels → ≈ **13 MB** on GPU needed

- Note
  - Three virtualization levels and PT blocks of 64^3
    → One MRPD entry addressing ≈ 68 billions of bricks

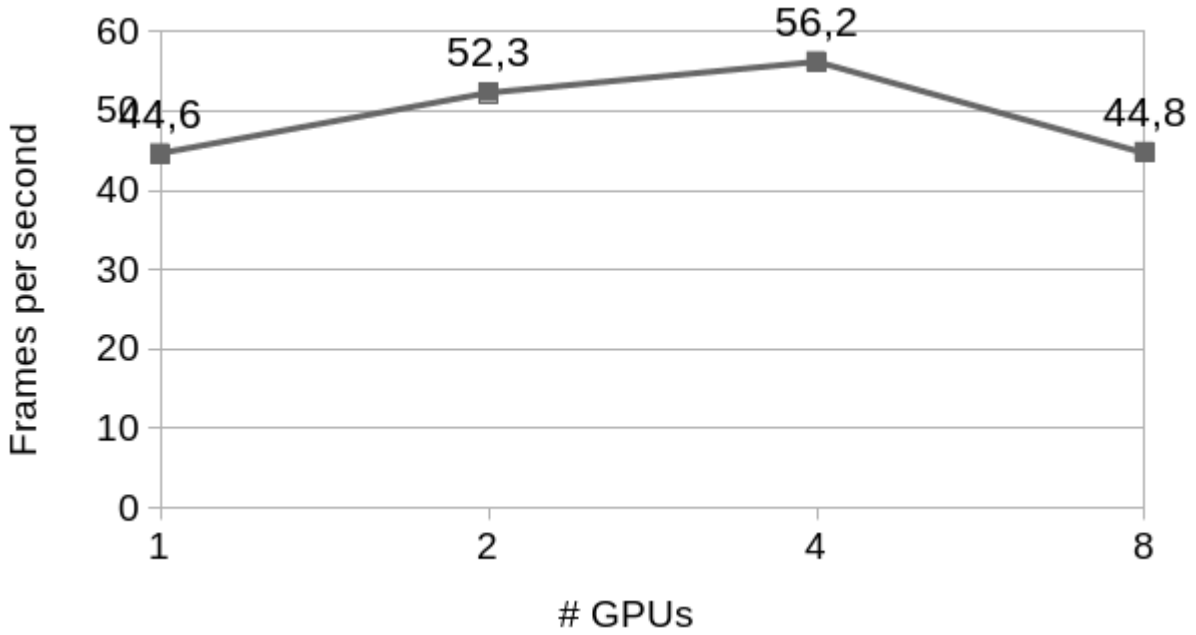
Rendering server

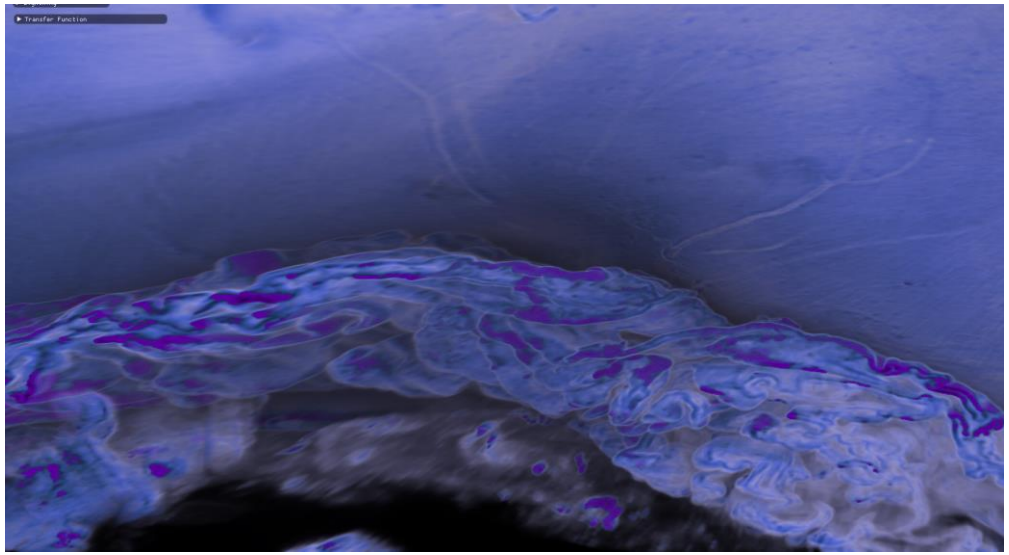
- Nvidia Quadro VCA
- 8 GPUs Quadro P6000
- 2 CPUs Intel Xeon

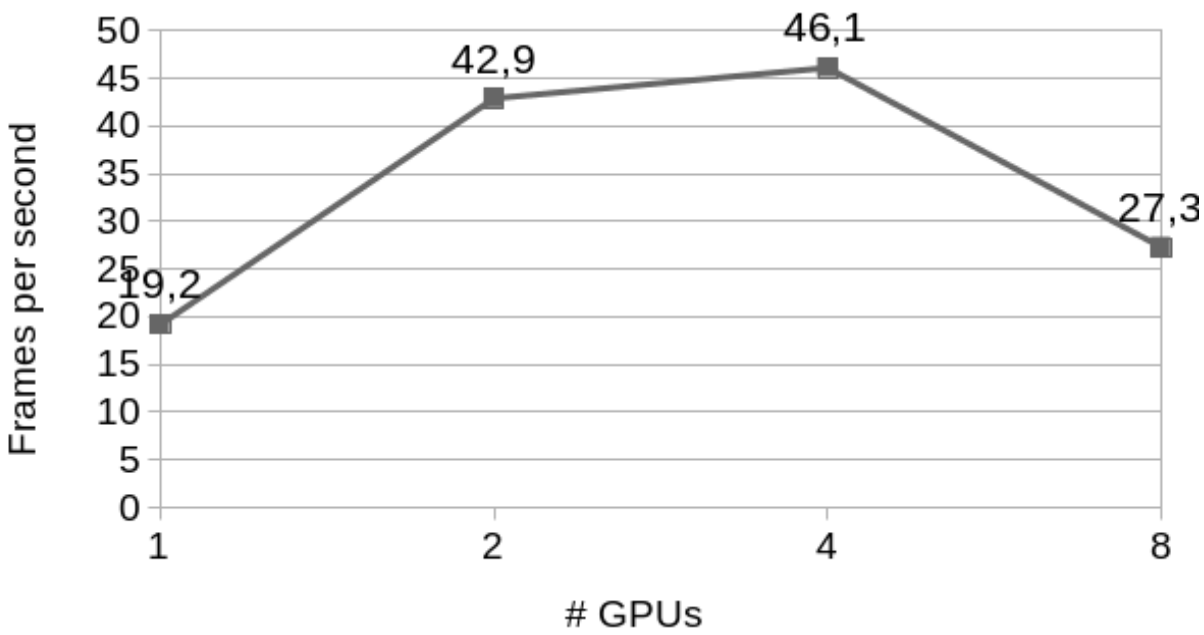
- One hybride node multi-GPUs multi-CPUs

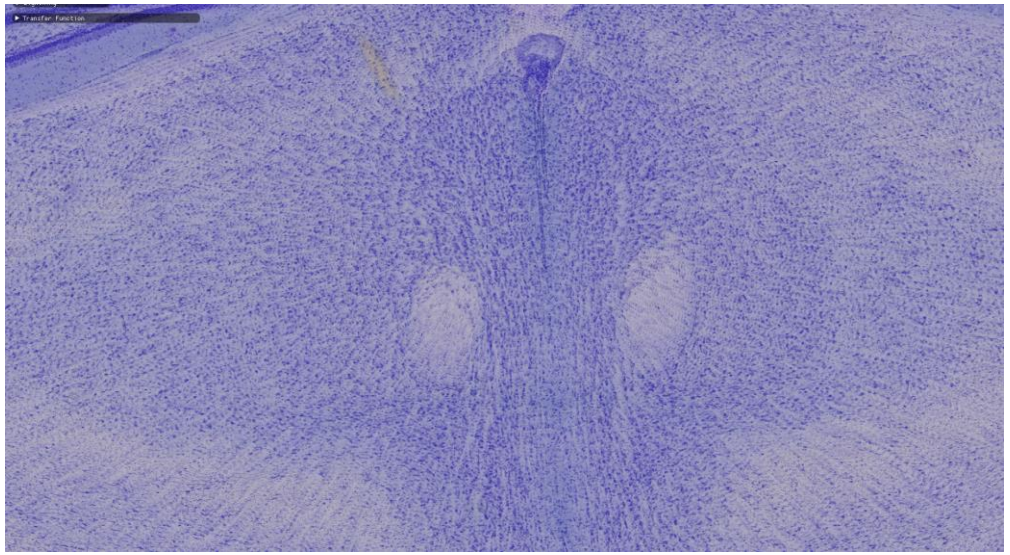


Frames frequency performances

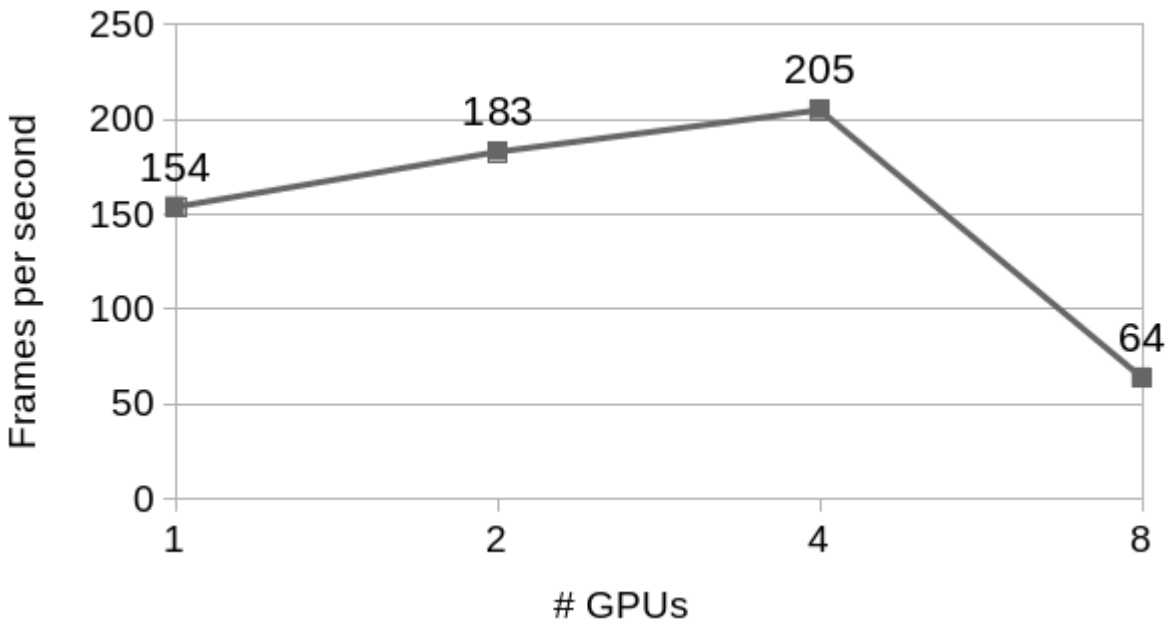


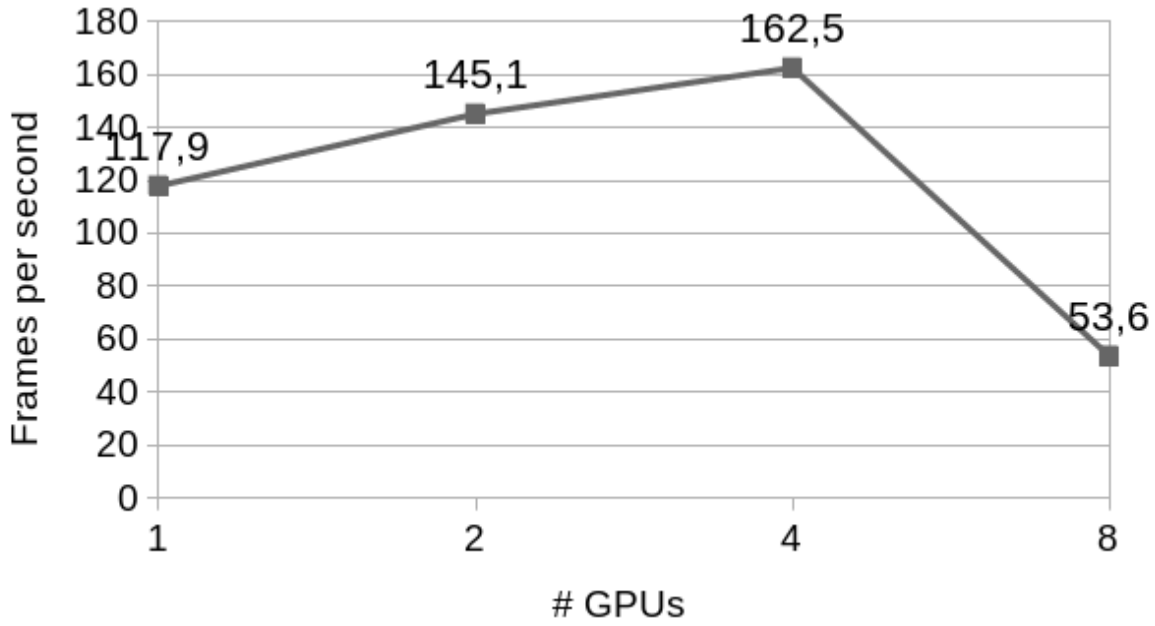


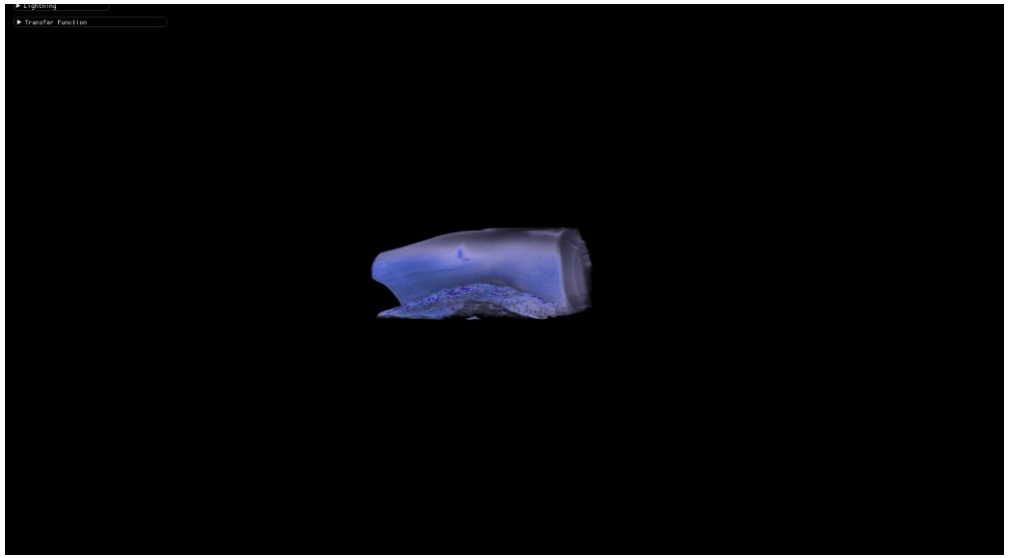


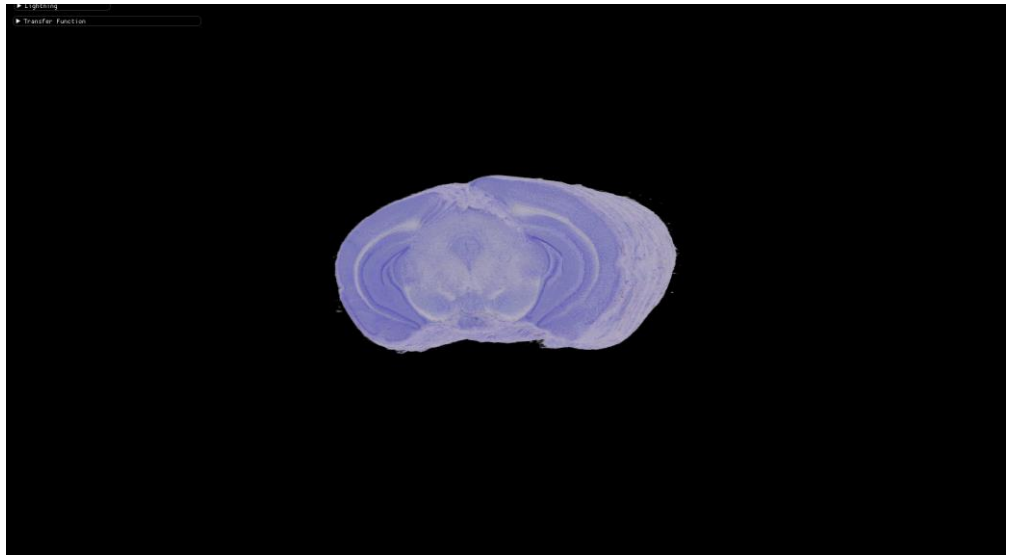


Frames frequency performances



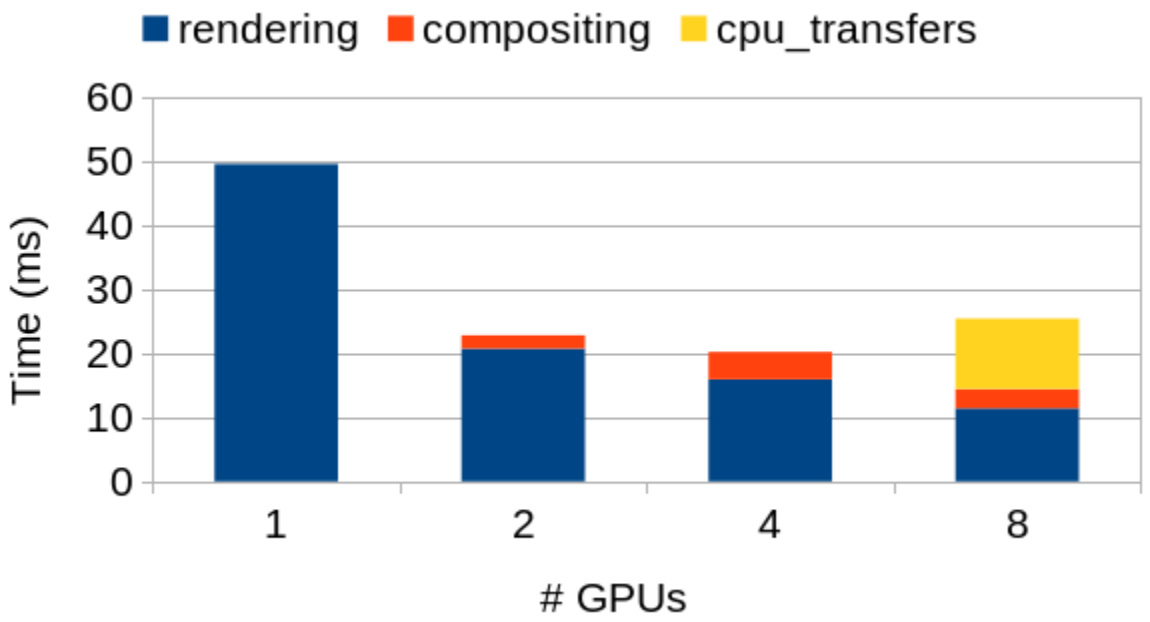


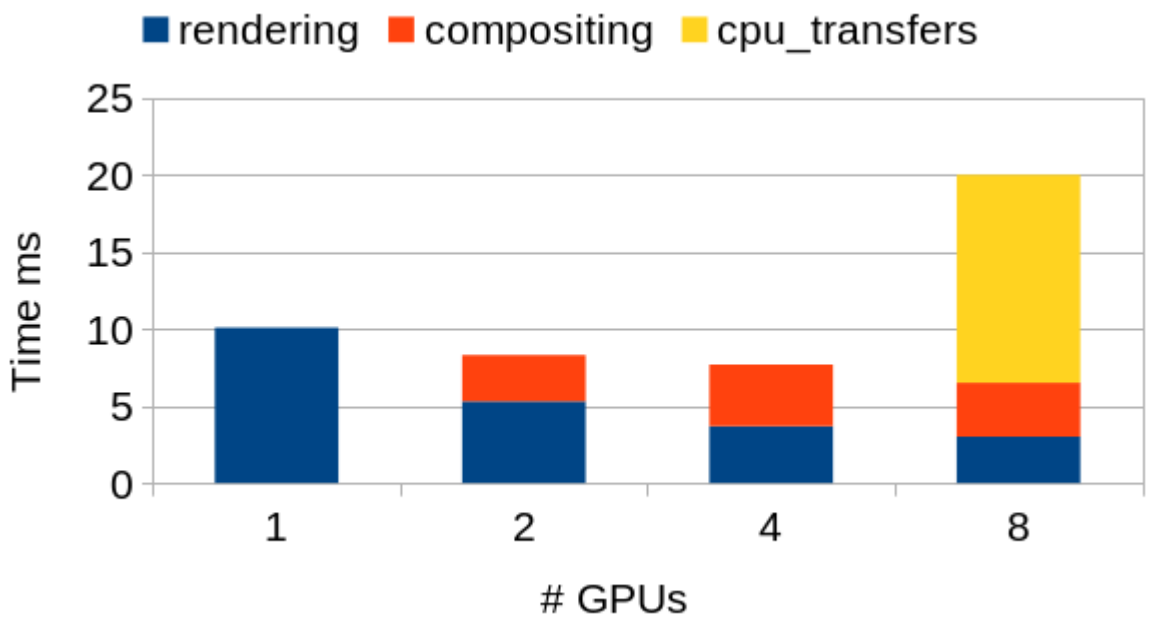




Frames frequency performances

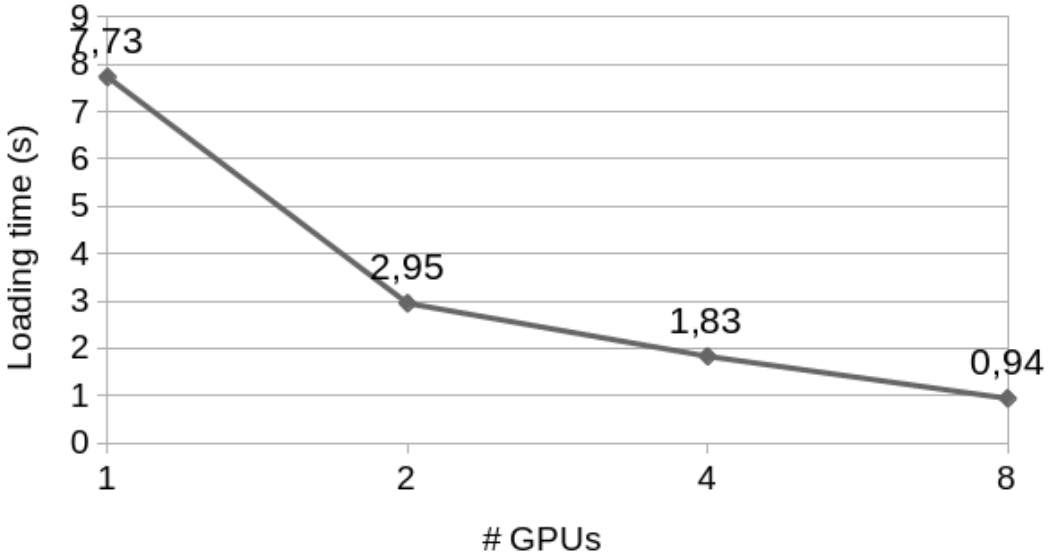
Details of the different steps of the rendering

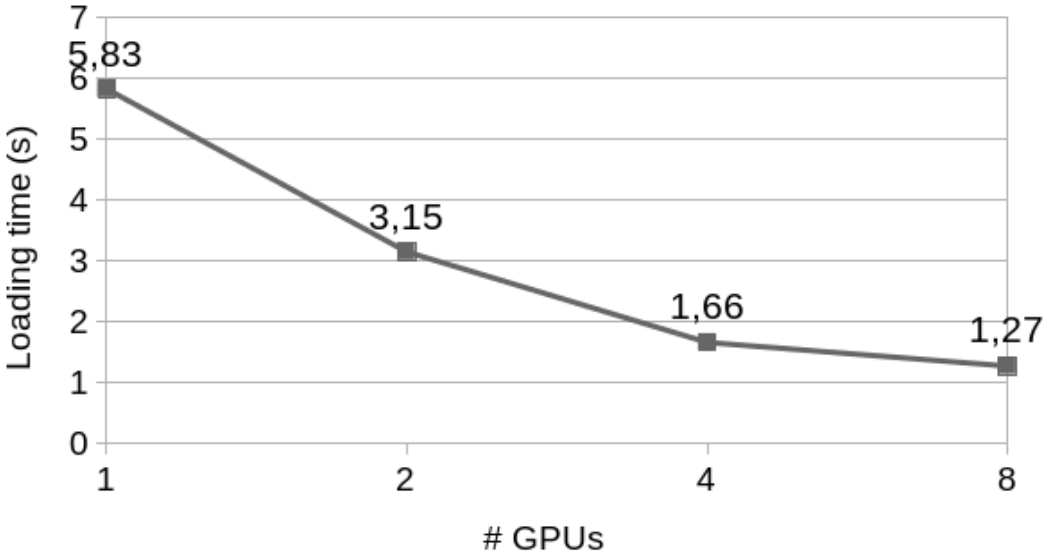


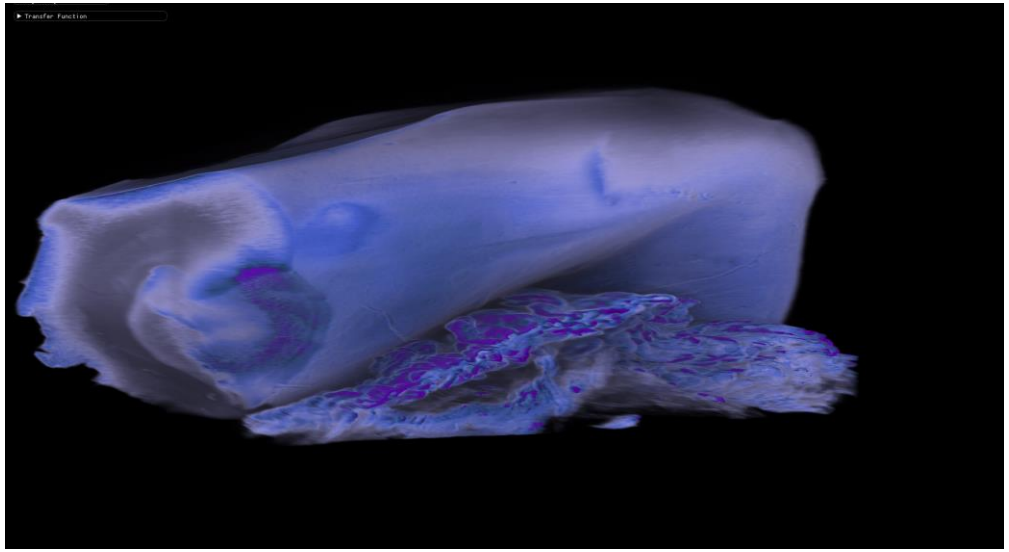


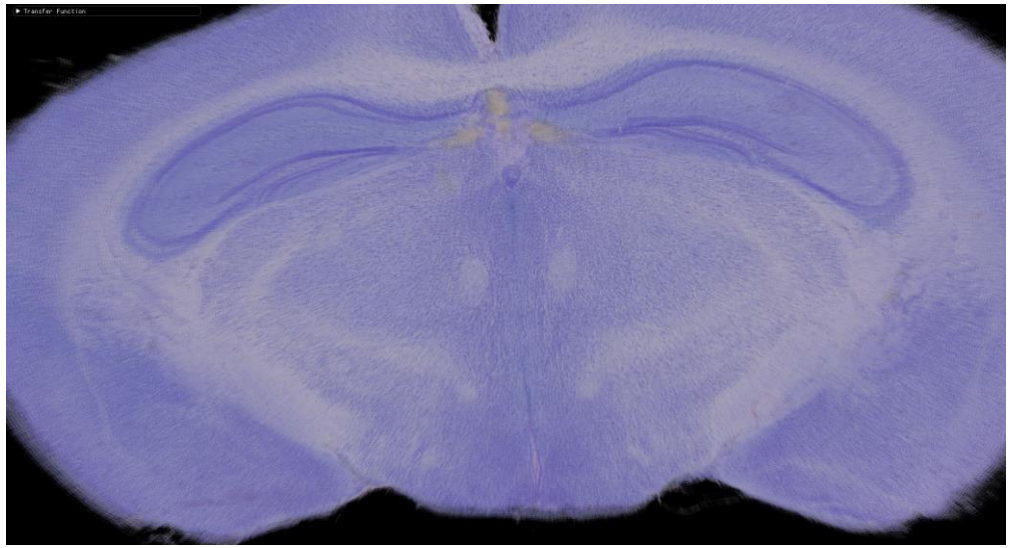
View loading performances

Worst case scenario: GPU and CPU empty caches









Conclusion and outlook

Conclusion

- Complete visualization-driven pipeline
- Out-of-core data management: multiresolution multilevel page table hierarchy
  - Entirely managed on GPU
  - GPU – CPU communication reduced
  - Any kind of applications (Regular 3D grid)
  - Adapted to HPC environment
  - Not OS depend
  - Weak GPU memory footprint
- Application: Multi-GPUs ray-guided multiresolution ray-casting
  - 1 – Good rendering frequency
  - 2 – Efficient on-demand data loading time
  - 1 + 2 – Even for very large volume of data (> TB)

Questions ?

A Fully GPU-Based Out-Of-Core Approach to Handle Large Volume Data

Nicolas Courilleau – nicolas.courilleau@neoxia.com

Jonathan Sarton – jonathan.sarton@univ-reims.fr

Florent Duguet – florent.duguet@altimesh.com

Yannick Remion – yannick.remion@univ-reims.fr

Laurent Lucas – laurent.lucas@univ-reims.fr