J.Fig Visualisation 2017

Une approche out-of-core entièrement basée GPU pour la visualisation et le traitement de gros volumes de données

J. Sarton[1], N. Courilleau[1,2], Florent Duguet[1,3], Y. Remion[1], L. Lucas[1]

1. Université de Reims Champagne-Ardenne, CReSTIC
2. Neoxia, France
3. Altimesh, France

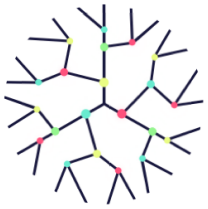
24 Octobre 2017

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

3DNEUROSECURE
calcul intensif et simulation numérique

Outline

Motivations



3DNEUROSECURE
calcul intensif et simulation numérique

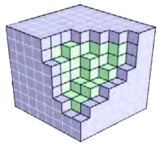
Objectifs

- **Visualisation** : microscopie virtuelle, ray-casting volumique
- **Traitement** : segmentation, classification (deep learning) ...

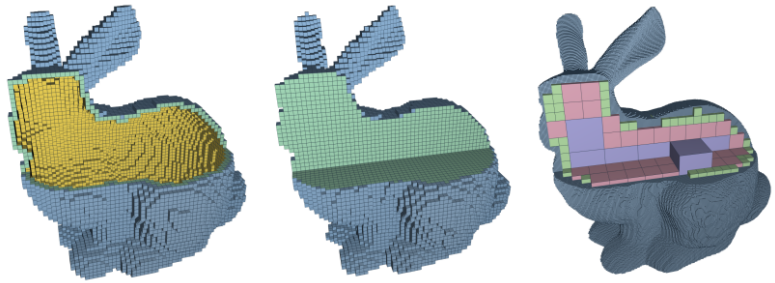
En temps interactif sur des **données volumique** → **GPU** :)

- Données médicales (scans, IRM ...)

- Grille régulière 3D → voxels



- Voxelisation de maillage



[Schwarz et al., SIGGRAPH Asia, 2010]

Problématique

Mémoire

Représentation voxels → de **gros volumes de données**.
Dépassant souvent la quantité mémoire GPU et même CPU.
Traitements en temps interactif = compliqué.

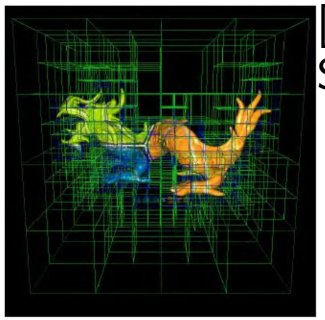
- Voxelisation : 4096^3 en RGBA → ~275 Go
- Histologie → quelques 100aines Go - quelques To
- Microscopie électronique → quelques 100aines de To - quelques Po

→ **Elaboration d'algorithmes out-of-core**

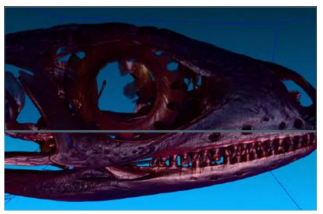
Etat de l'art

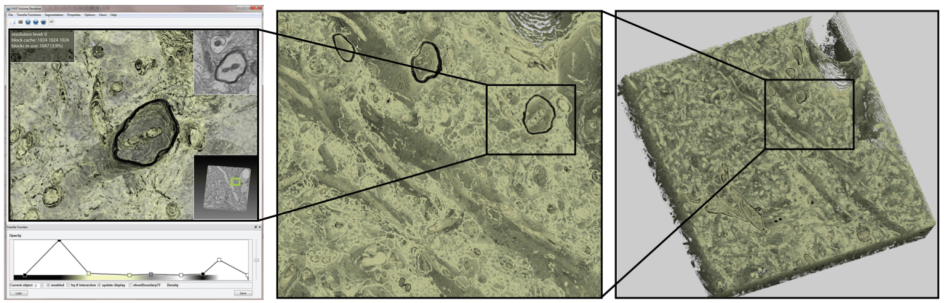
Contexte :

- Ray-casting volumique Out-of-core sur GPU.

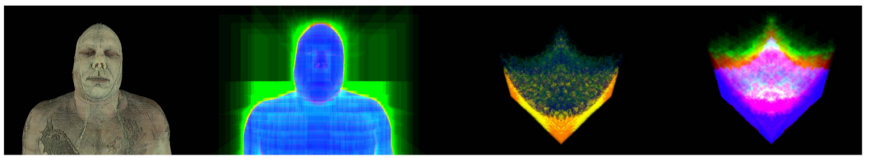


[Crassin et al., ACM SIGGRAPH i3D, 2009]





[Hadwiger et al., IEEE SciVis 2012]



[Fogal et al., IEEE Symposium on Large Data Analysis and Visualization 2013]

Approches :

- Représentation : bricking, multi-résolution ...
- Stockage : Cache de briques en mémoire texture
- On-demand paging / data streaming

Complete State Of The Art : [Beyer et al., Computer Graphics Forum 2015]

Contributions

Visualisation augmentée

Une solution permettant de :

- **Visualiser**
- **Traiter**

de **très gros volumes** dépassant la quantité mémoire GPU ou CPU **en temps interactif !**

Un **pipeline out-of-core** complet, du disque jusqu'au GPU, pour **accéder à l'ensemble des données à la demande** de l'application. Intégrant :

1. Une **structure d'adressage virtuel** : hiérarchie de table de pagination [Hadwiger et al., IEEE SciVis 2012]
2. Une gestion de la structure **entièrement sur GPU** [Crassin, PhD Thesis, 2011]

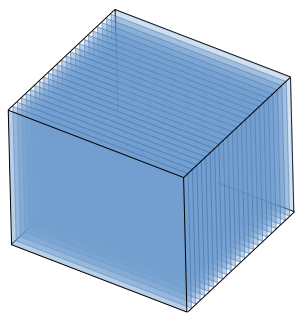
Representation et stockage des données

Le volume est stocké sur un device avec beaucoup d'espace mémoire.

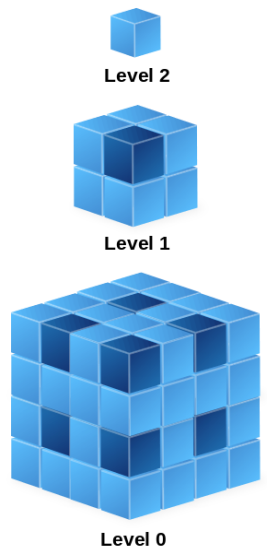
- **Multi-résolution**: permet de choisir le niveau de détail désiré.
  $\Rightarrow$ Réduit la quantité de données.
- **Bricking**: Subdiviser le volume en petites briques (e.g 16^3, 32^3).
  $\Rightarrow$ Permet une approche out-of-core.

Mipmap 3D

- Pyramide 3D multi-résolution briquée.

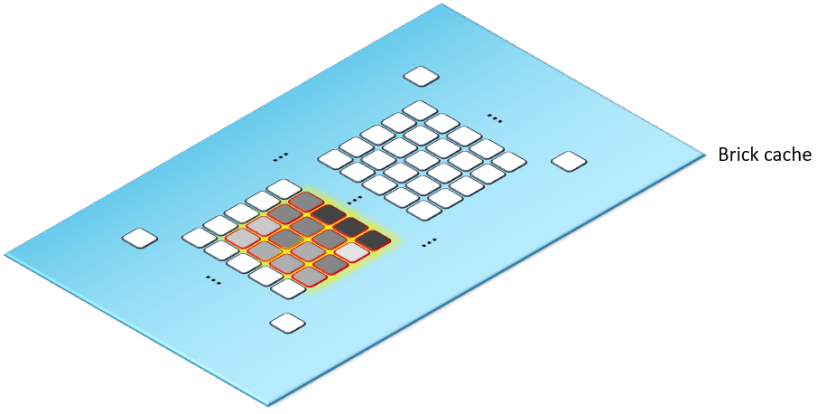


$\Rightarrow$

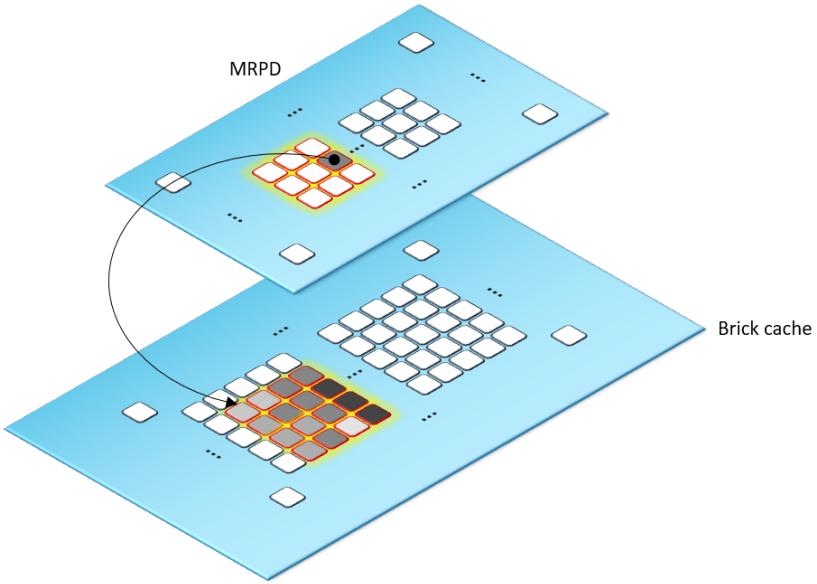


Stockage avec **compression** (Algorithme LZ4, décompression temps-réel).

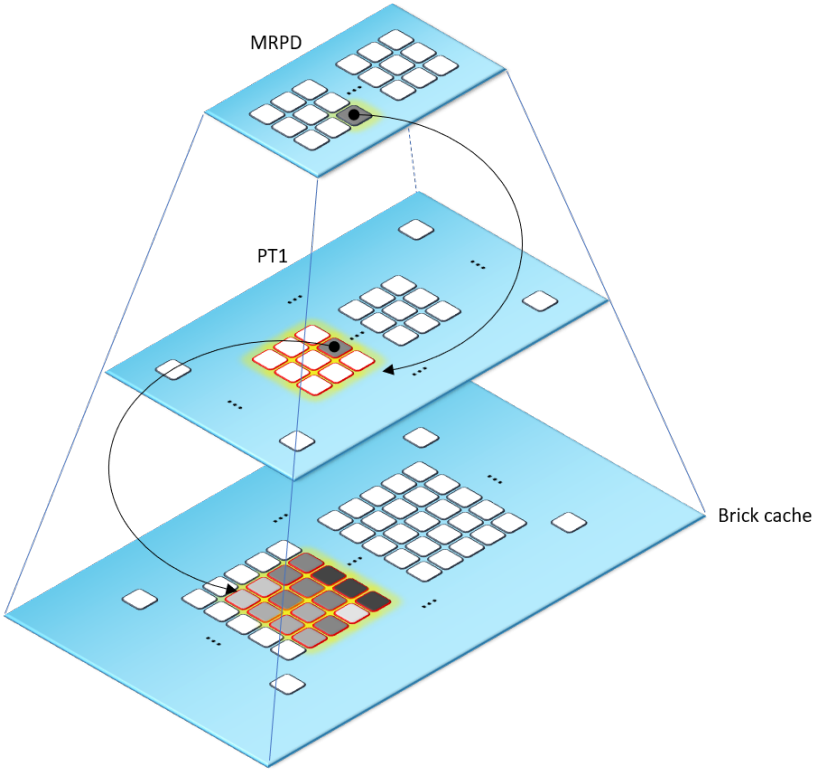
Hiérarchie de table de pagination multirésolution multi-niveaux



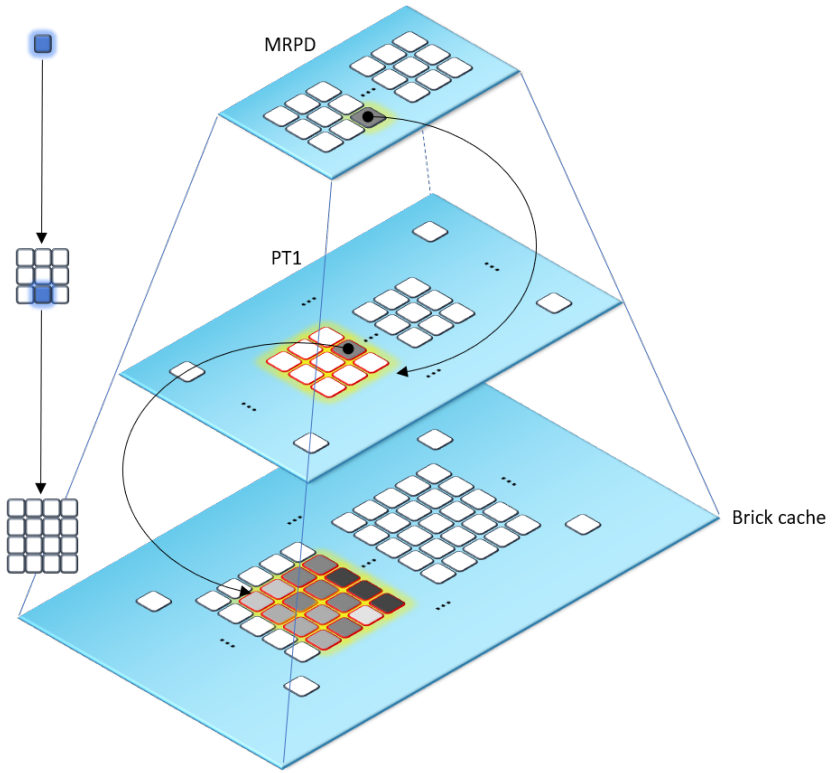
Hiérarchie de table de pagination multirésolution multi-niveaux



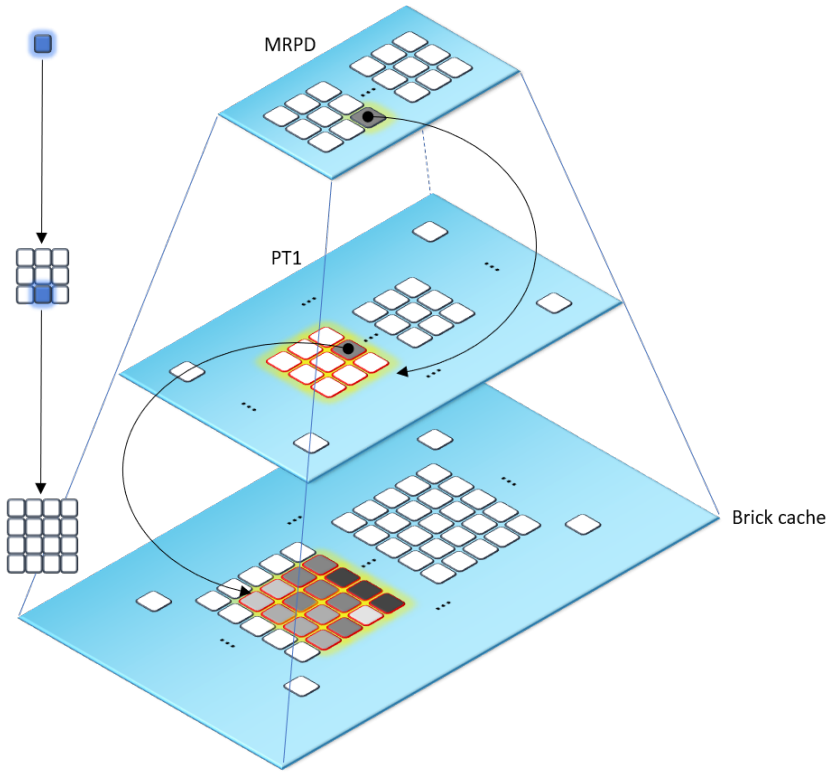
Hiérarchie de table de pagination multirésolution multi-niveaux



Hiérarchie de table de pagination multirésolution multi-niveaux

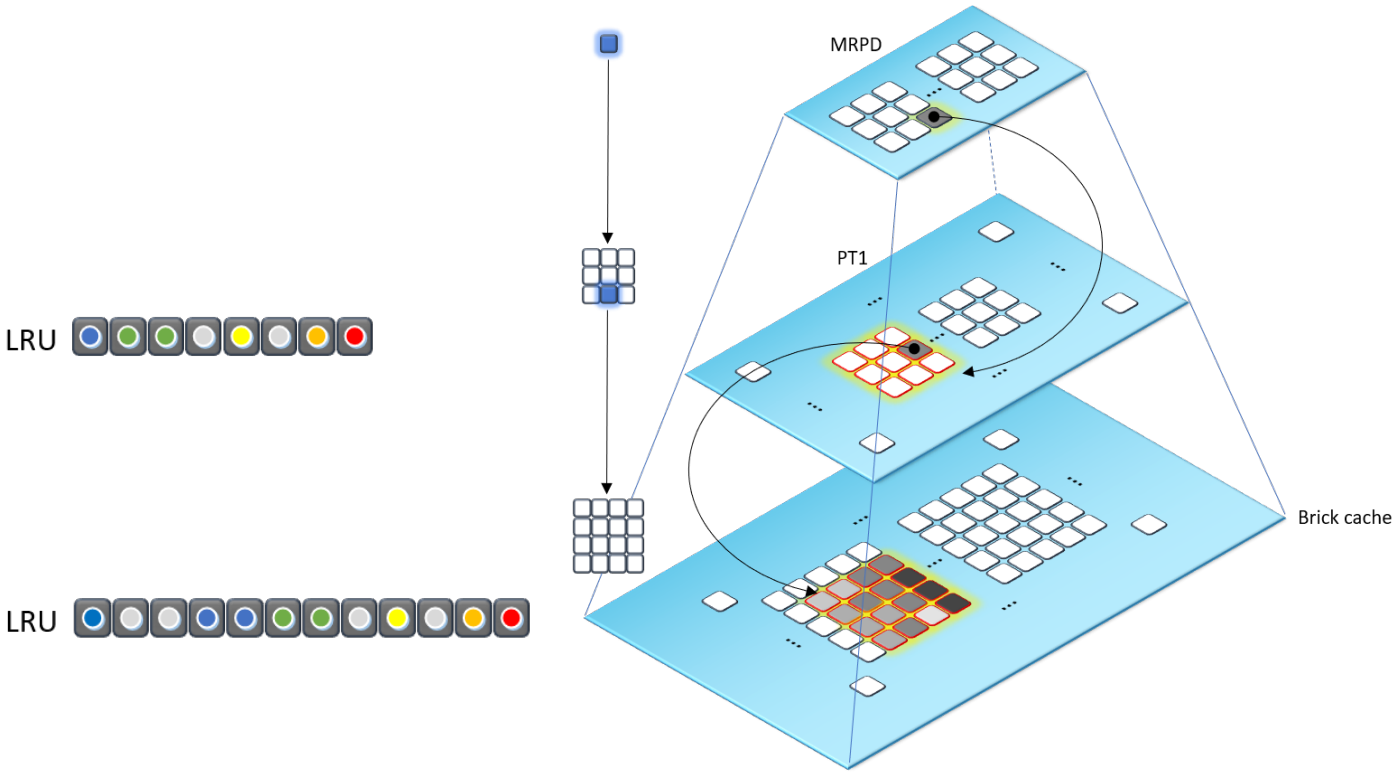


Hiérarchie de table de pagination multirésolution multi-niveaux



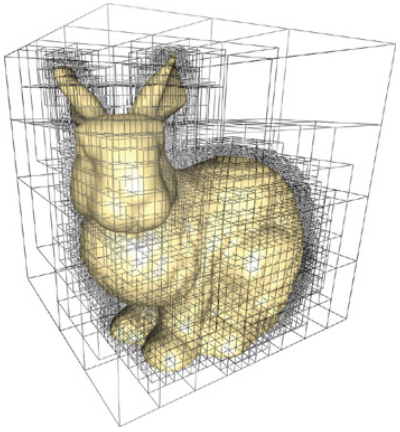
- Une page = adresse 3D vers un bloc dans le niveau inférieur + un flag :
  - 0 : non présent sur le GPU.
  - 1 : présent en cache GPU.
  - 2 : adresse une brique vide.

Hiérarchie de table de pagination multirésolution multi-niveaux

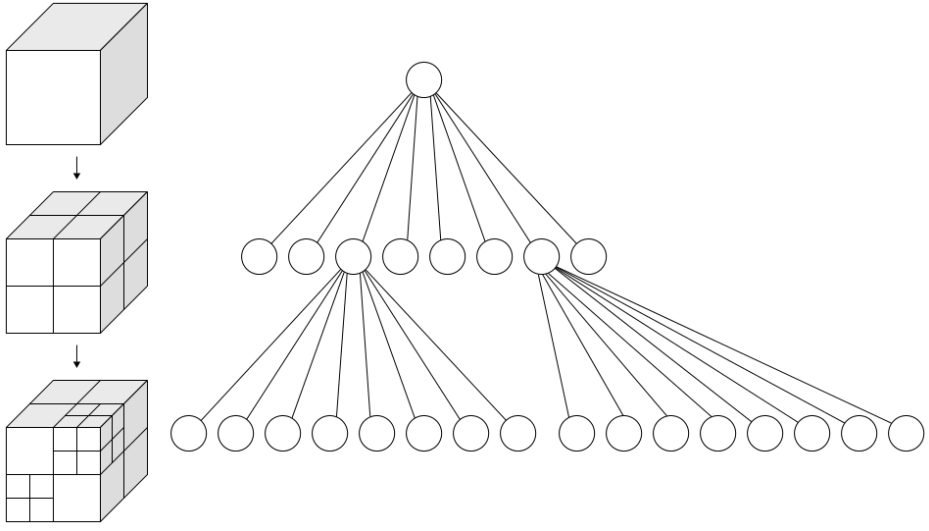


- Implémentation : Texture 3D lecture/écriture
- Algorithme de remplacement des caches : Least Recently Used (LRU)

Comparaison avec un octree multi-resolution



[GPU Gems 2, Chap. 37
Octree Textures on the GPU]

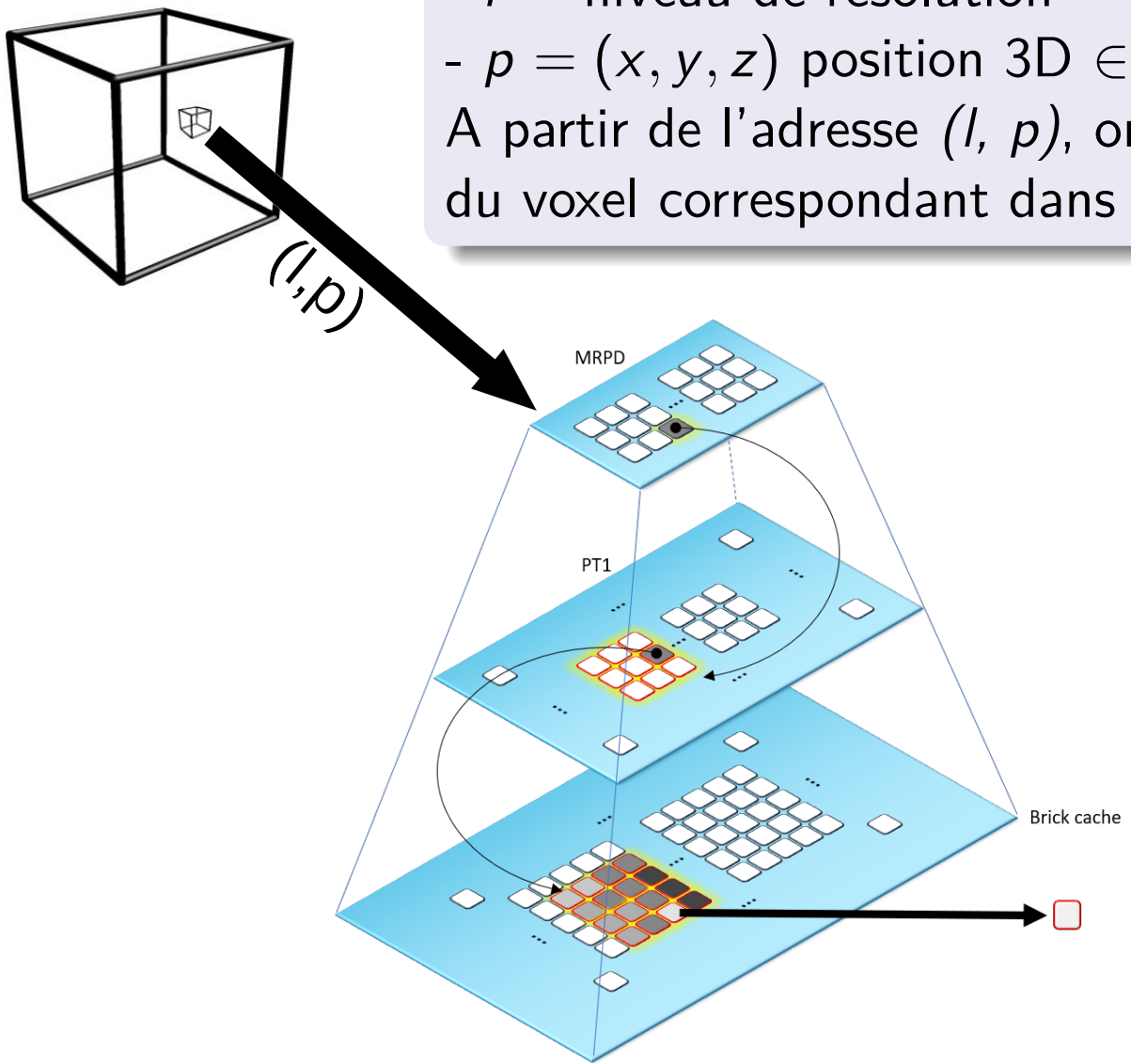


- Pas de maintien ou de parcours d'arbre sur GPU.
- Adressage en temps presque constant, quelque soit le niveau de résolution demandé.
- Permet de dissocier la qualité de rendu et le temps d'accès (arbre trop profond ou trop large).

Adressage virtuel

Navigation dans un **volume virtuel** $\rightarrow$ adresse *(l, p)*
- l = niveau de résolution
- $p = (x, y, z)$ position 3D $\in [0, 1[^3$

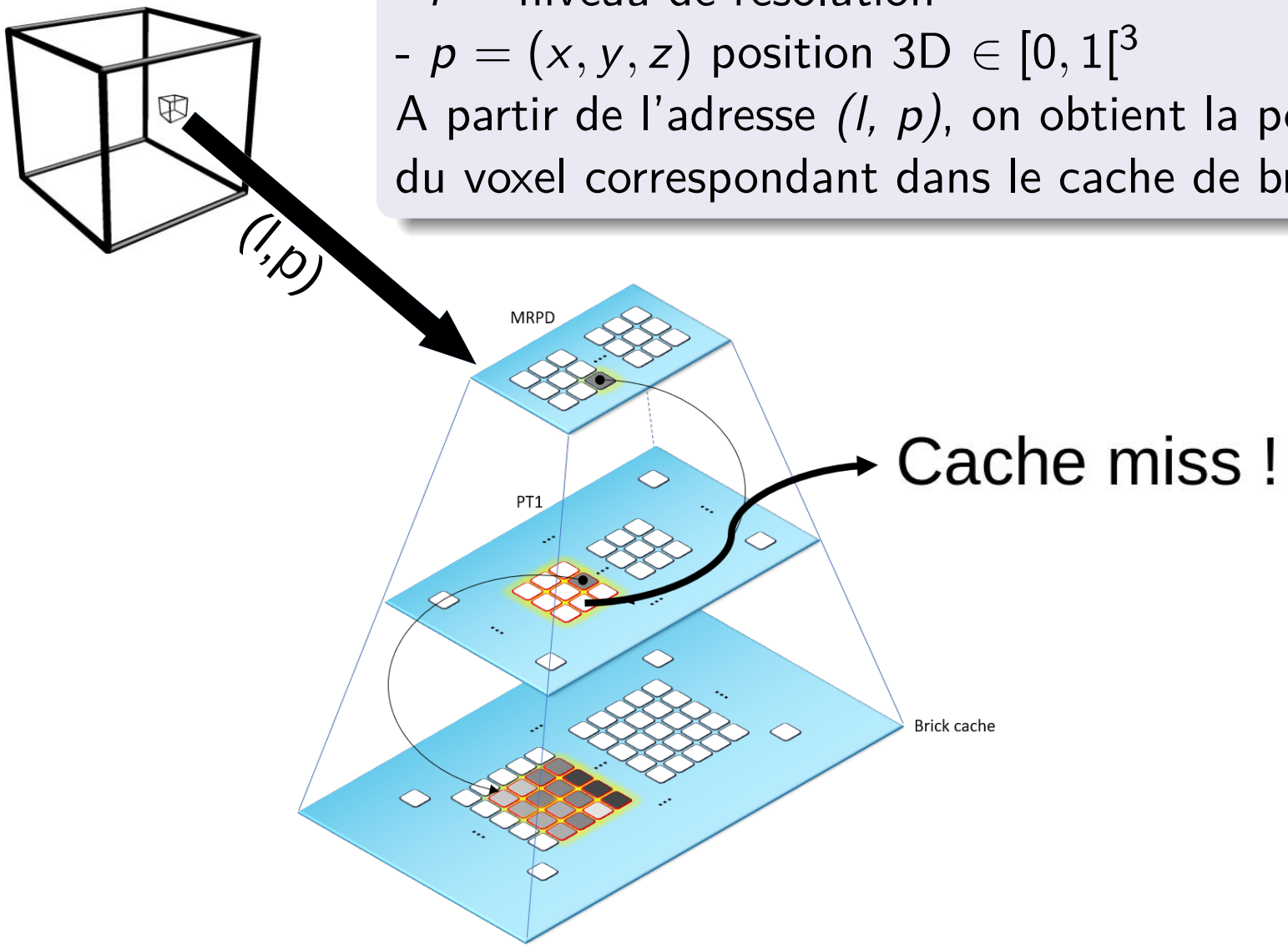
A partir de l'adresse *(l, p)*, on obtient la position 3D du voxel correspondant dans le cache de briques.



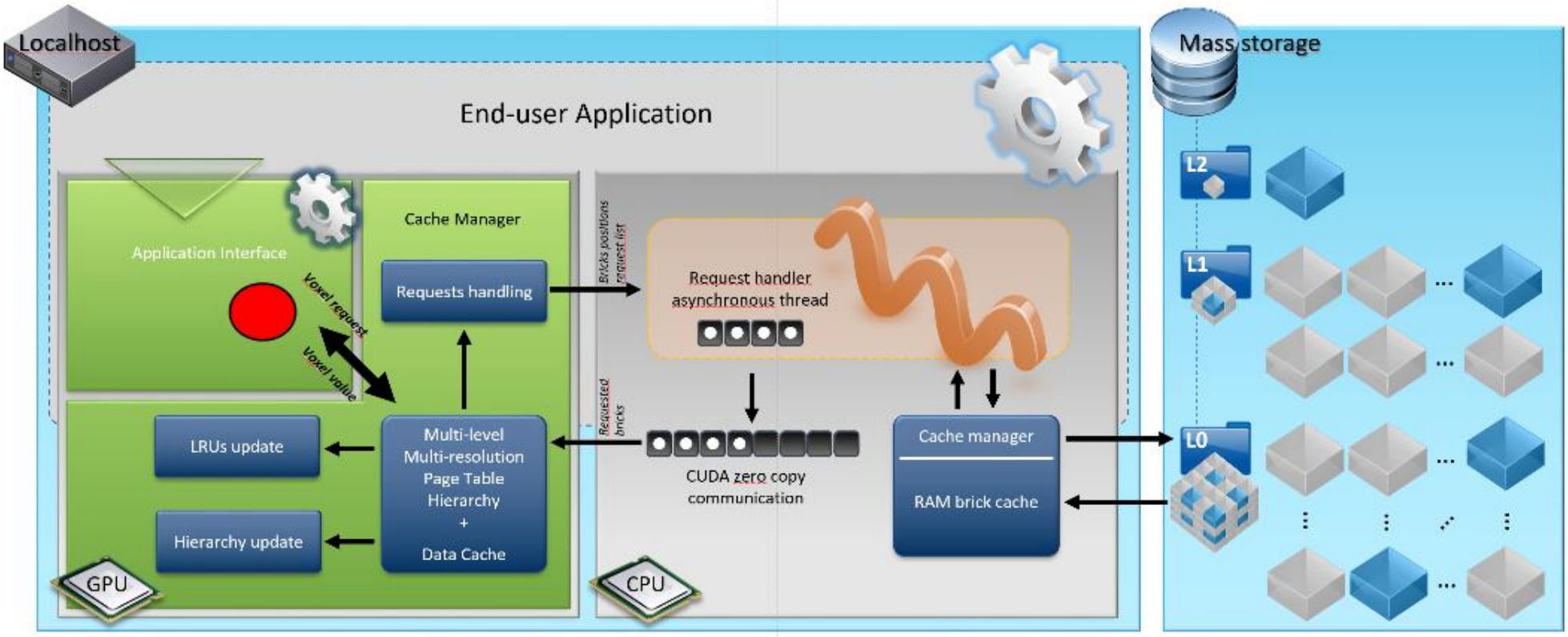
Cache miss

Navigation dans un **volume virtuel** $\rightarrow$ adresse *(l, p)*
- l = niveau de résolution
- $p = (x, y, z)$ position 3D $\in [0, 1[^3$

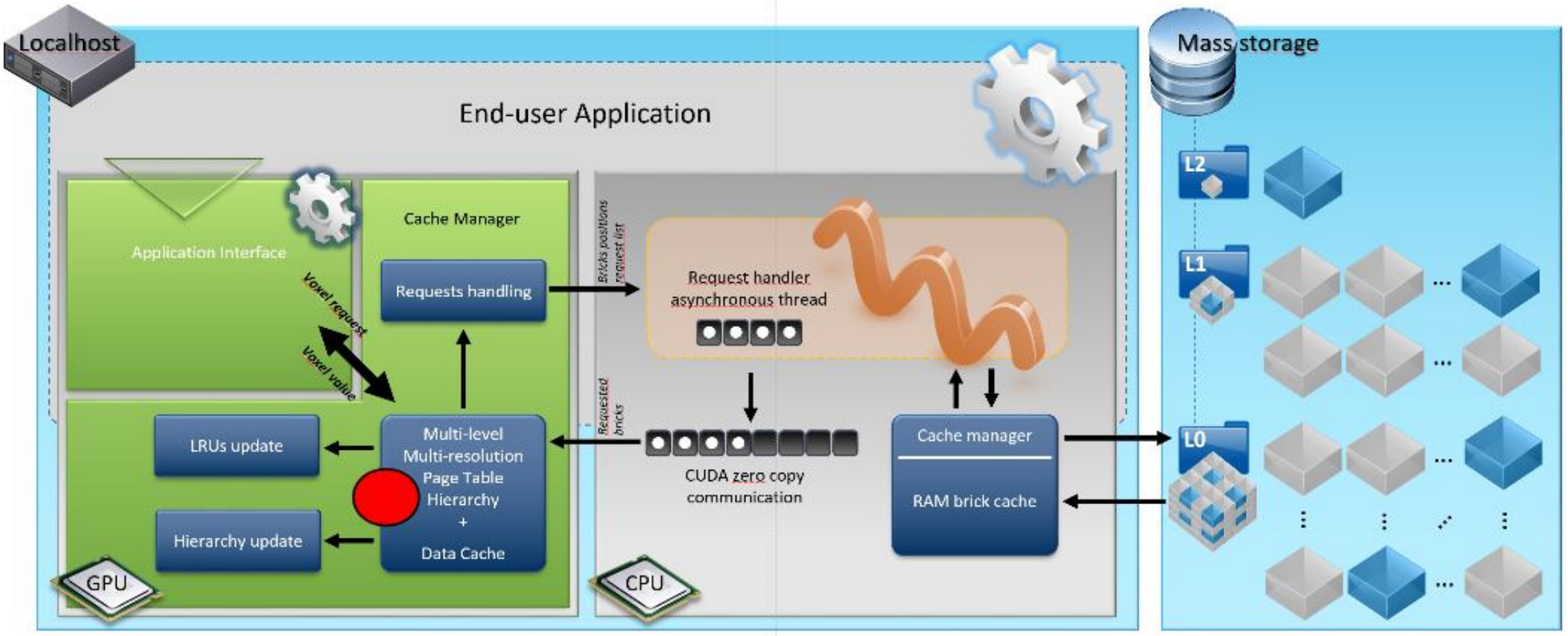
A partir de l'adresse *(l, p)*, on obtient la position 3D du voxel correspondant dans le cache de briques.



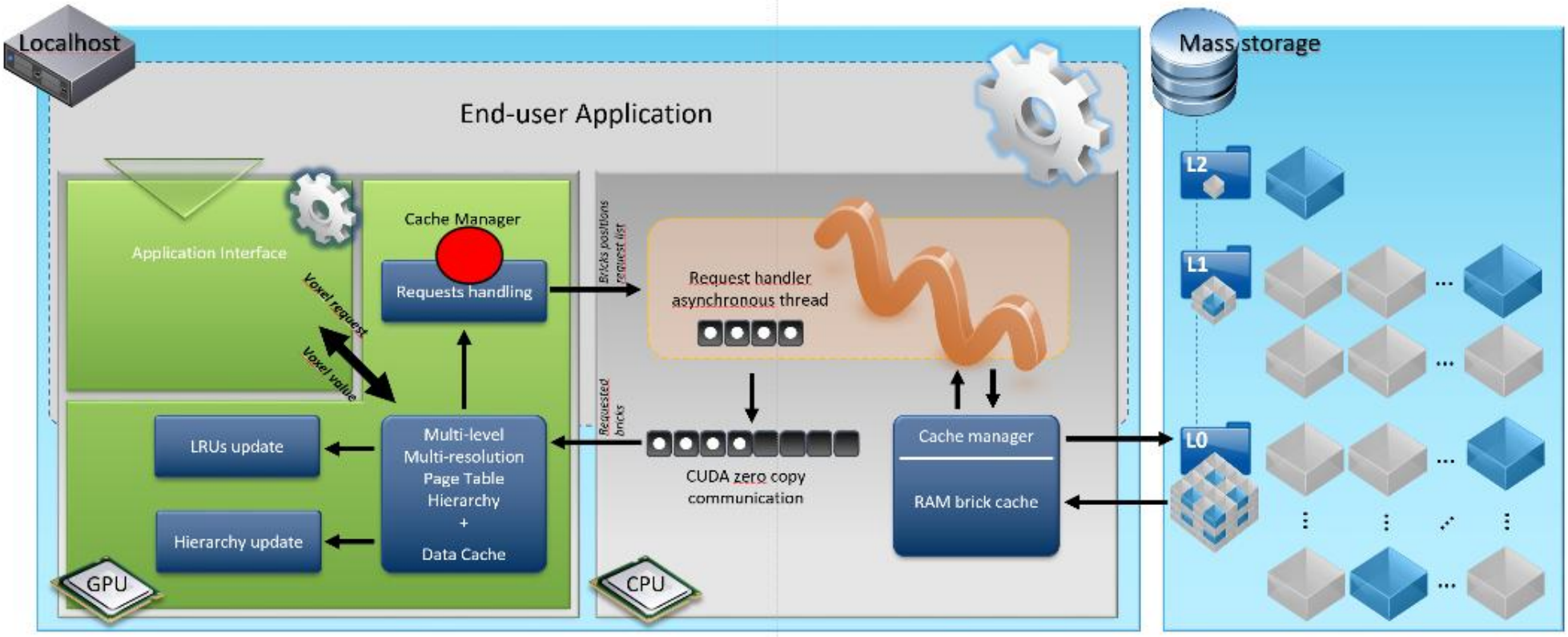
Pipeline


Localhost
End-user Application
Application Interface
Cache Manager
Requests handling
Voxel request
Voxel value
Bricks positions request list
Request handler asynchronous thread
Requested bricks
CUDA zero copy communication
LRUs update
Multi-level Multi-resolution Page Table Hierarchy + Data Cache
Hierarchy update
Cache manager
RAM brick cache
GPU
CPU
Mass storage
L2
L1
L0

Pipeline


Localhost
End-user Application
Application Interface
Cache Manager
Requests handling
Voxel request
Voxel value
Bricks positions request list
Request handler asynchronous thread
Requested bricks
LRUs update
Multi-level Multi-resolution Page Table Hierarchy + Data Cache
Hierarchy update
GPU
CUDA zero copy communication
CPU
Cache manager
RAM brick cache
Mass storage
L2
L1
L0

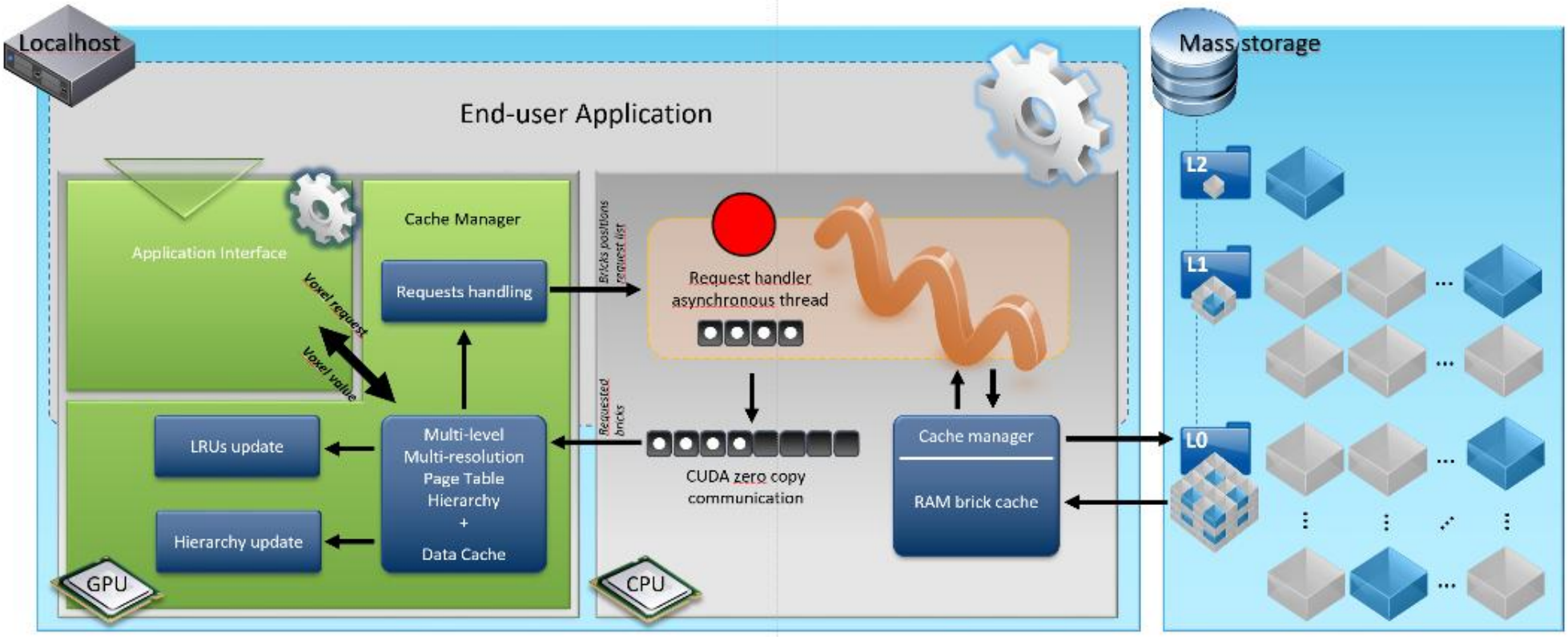
Pipeline



- Mise à jour

1. Gestion des requêtes de briques.

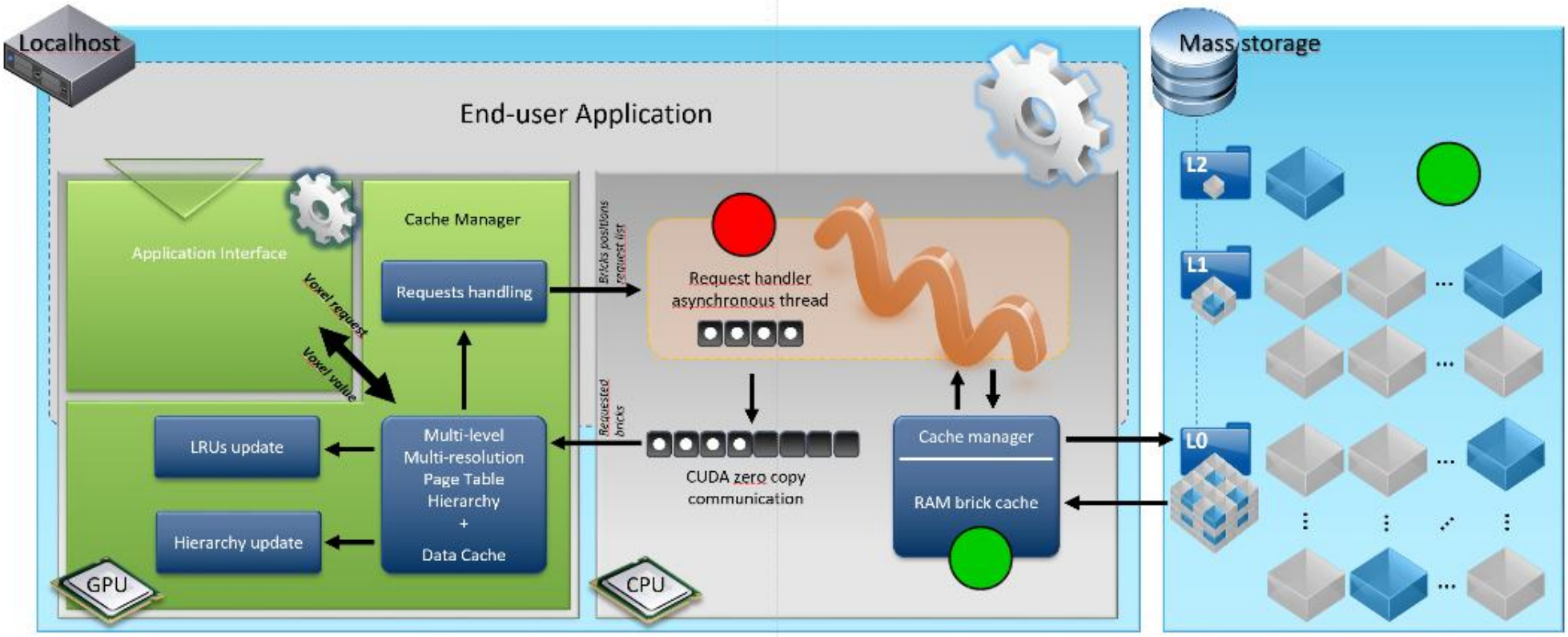
Pipeline



- Mise à jour

1. Gestion des requêtes de briques.

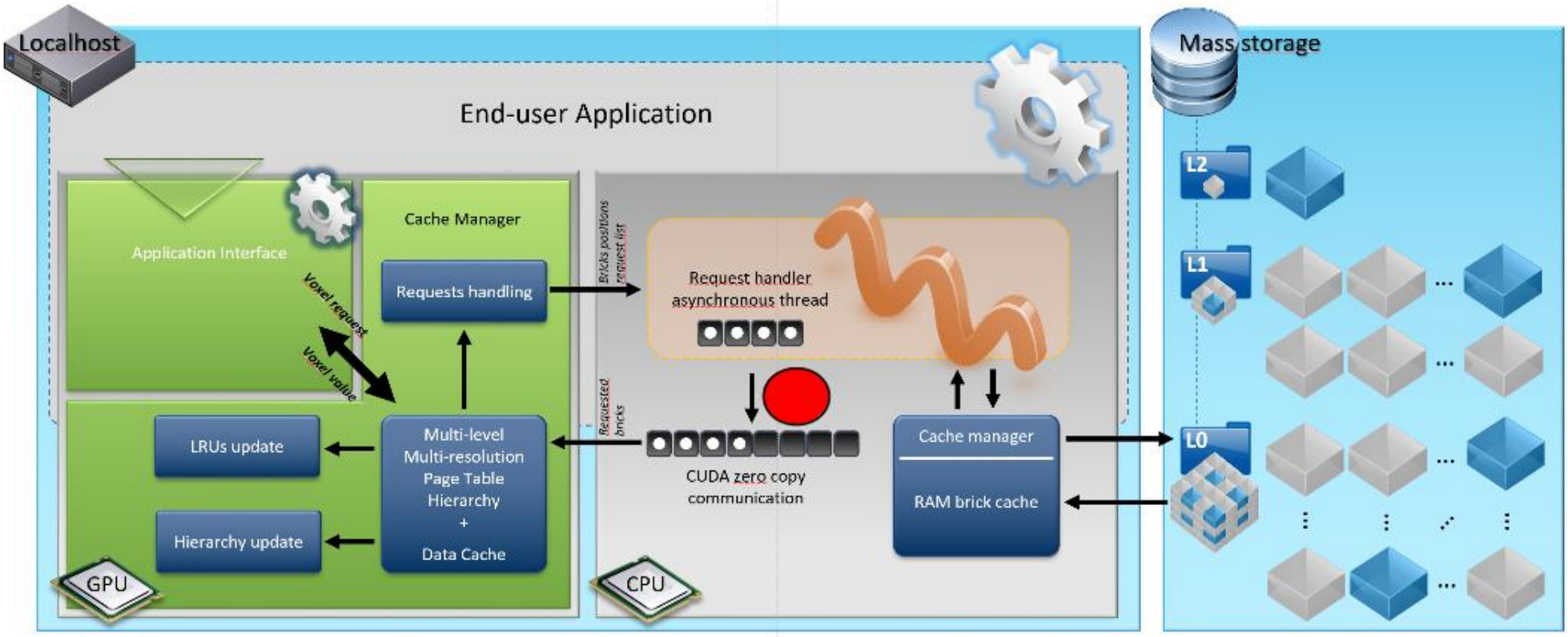
Pipeline



- Mise à jour

1. Gestion des requêtes de briques.

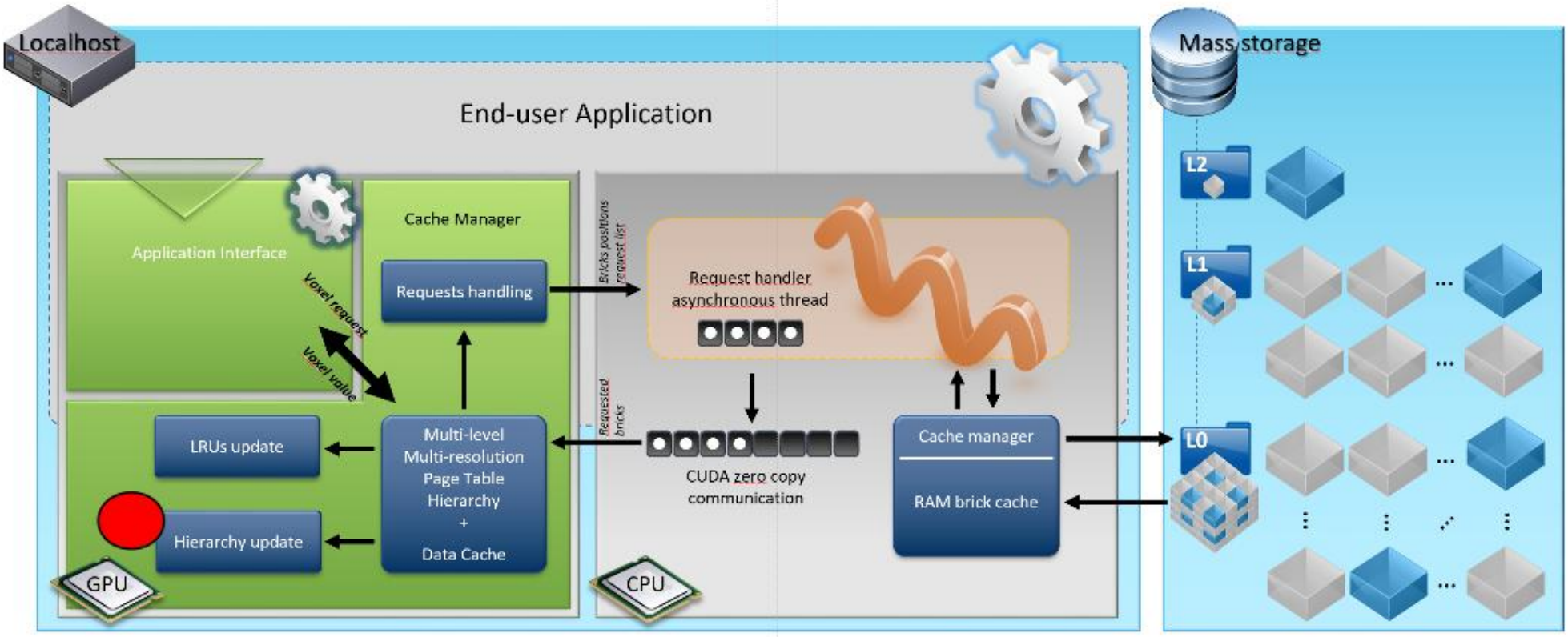
Pipeline



- Mise à jour

1. Gestion des requêtes de briques.

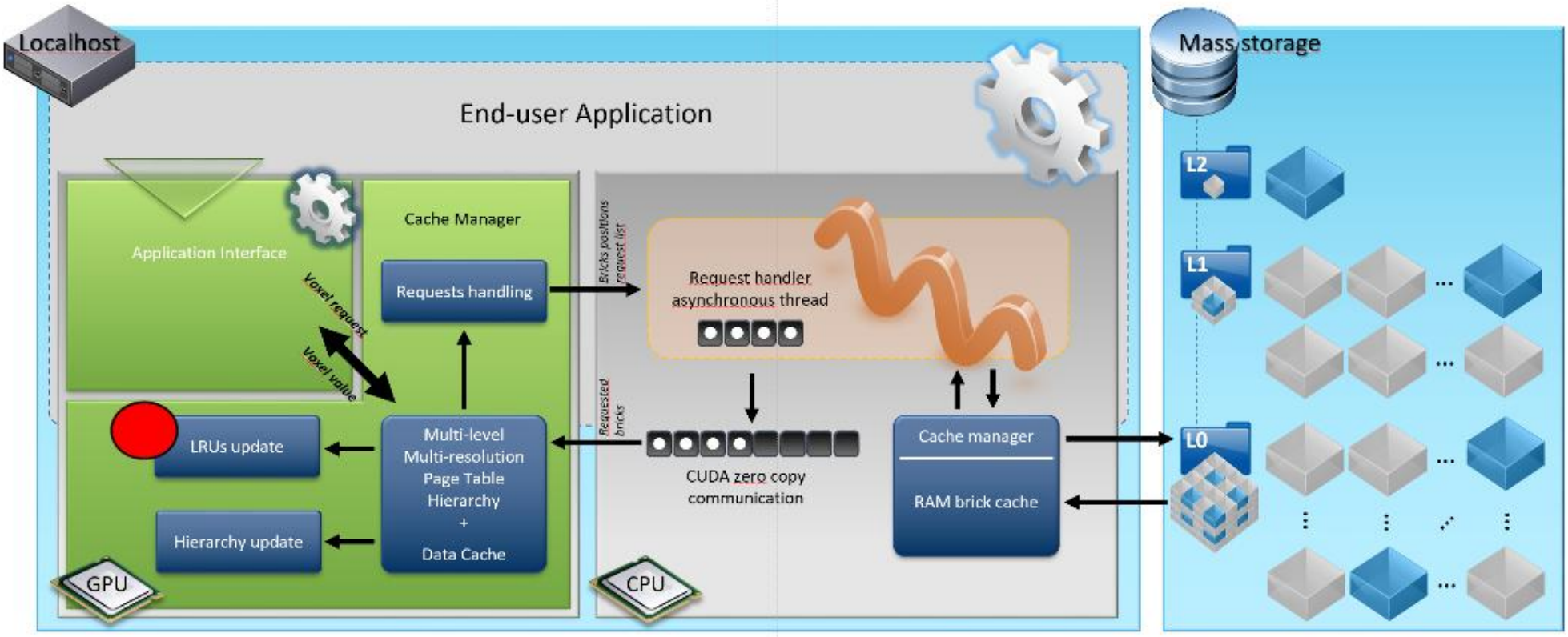
Pipeline



- Mise à jour

1. Gestion des requêtes de briques.

Pipeline



- Mise à jour

1. Gestion des requêtes de briques.
2. Gestion de l'utilisation des briques du cache.

Cache manager

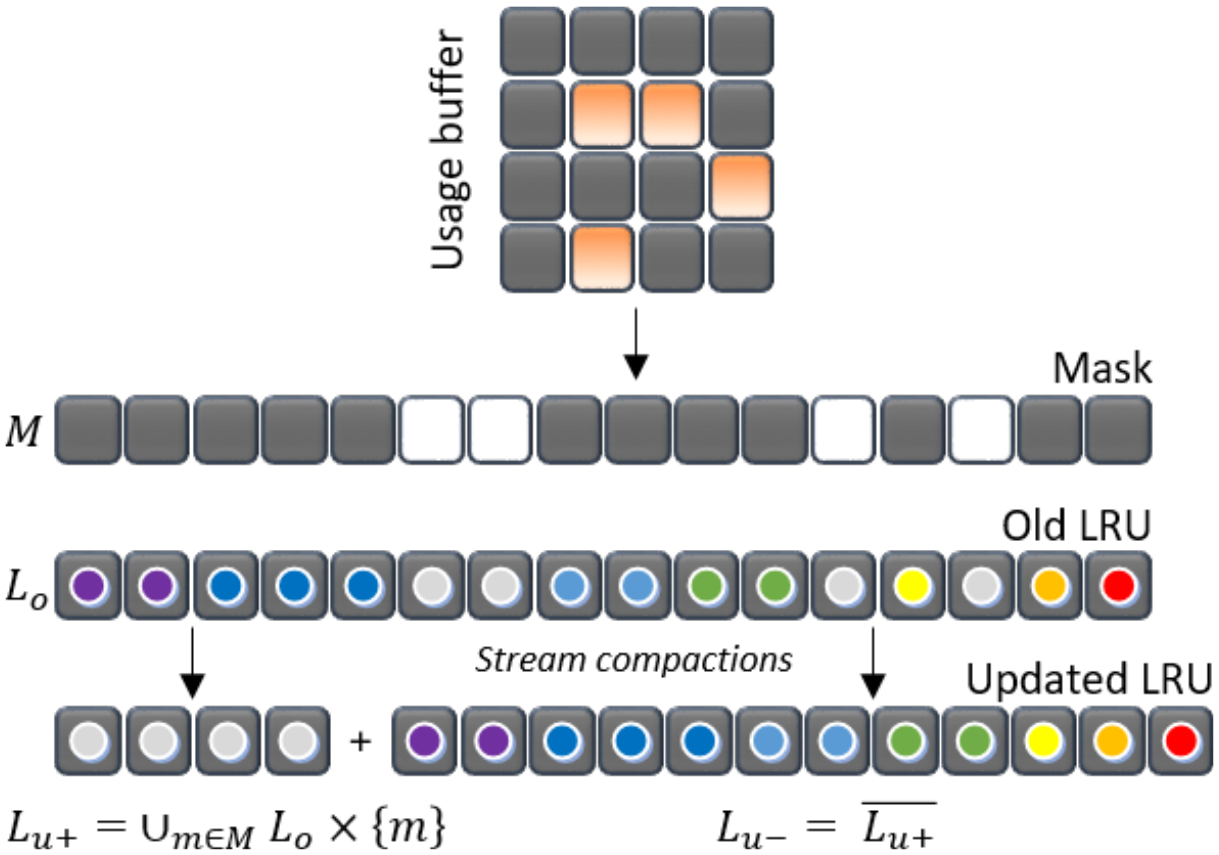
1. Mise à jour des LRUs
2. Gestion des requêtes de briques

Une gestion des caches GPU entièrement sur GPU !

Avantages

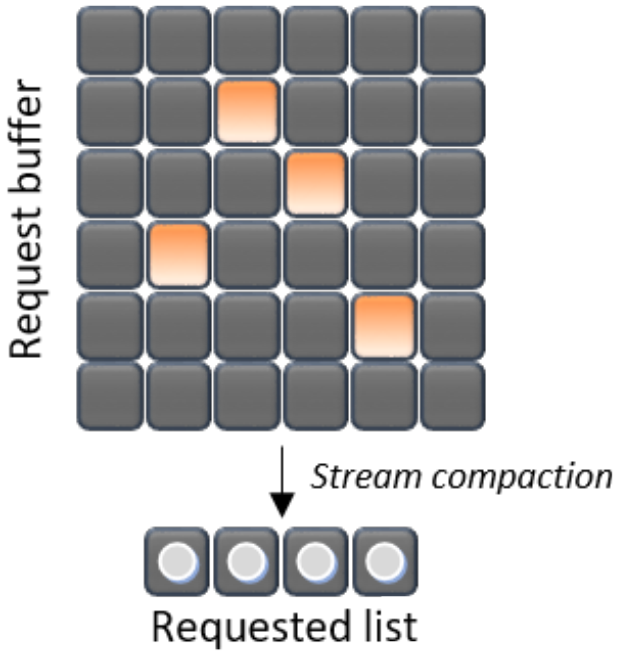
- Evite beaucoup de tranferts de données entre CPU et GPU.
- Permet de profiter de l'environnement massivement parallèle des GPUs.
- Libère le CPU pour d'autres traitements éventuels.

Mise à jour des LRUs



- Taille du nombre de briques dans le **cache**
- Marquage via un timestamp

Gestion des requêtes de briques

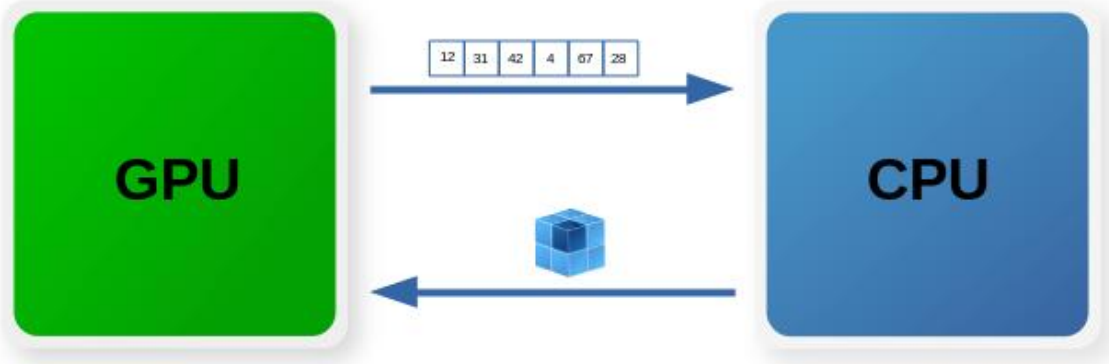


- Taille du nombre de briques dans le **volume**
- Marquage via un timestamp

Transferts entre CPU et GPU

Communications GPU → CPU

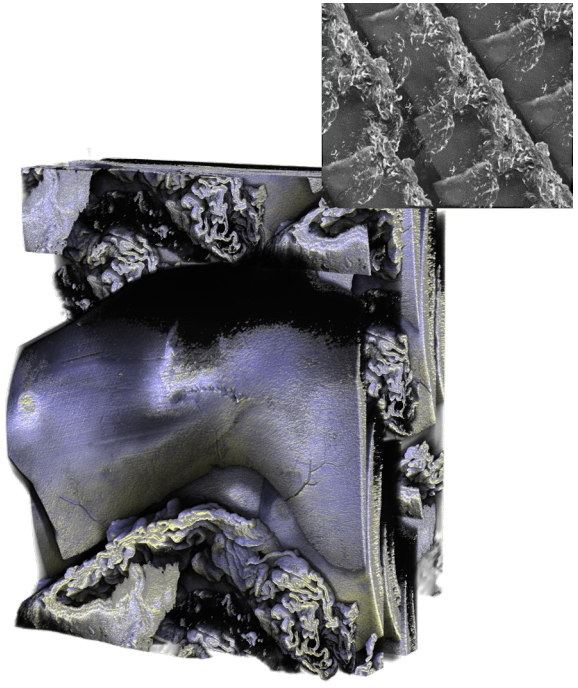
Une simple liste avec les identifiants des briques requétées !



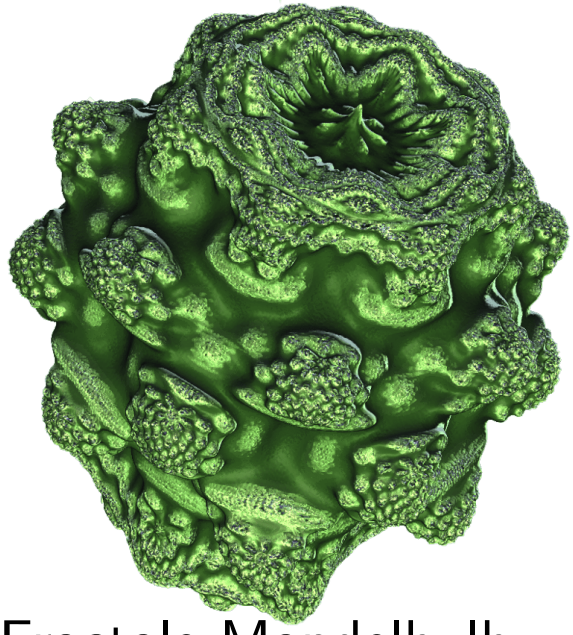
Communications GPU ← CPU

Uniquement les briques ! (Avec CUDA Zero Copy !)

Résultats



Microscope "light sheet"
$2160 \times 2560 \times 1072$ 16bits ~11 GB



Fractale Mandelbulb
4352^3 RGBA ~330 GB

Visualiseur de coupe 2D

- Navigation (zoom et déplacement) avec rendu multirésolution 2D.
- Morphologie mathématique basique.

Résultats

Utilisation mémoire GPU

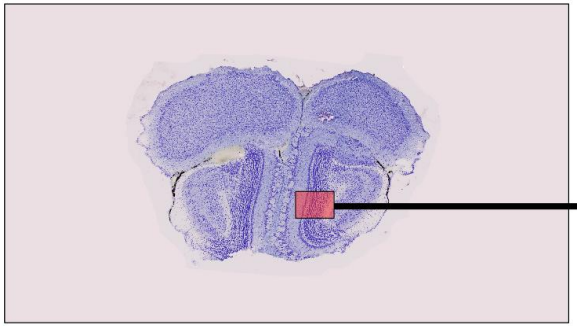
- Microscope "light sheet" $2160 \times 2560 \times 1072$ ~11 GB
  - Taille de brique : 64^3 => 27000 briques
  - Configuration 1 niveau de virtualisation
    → Requiert **1.5 MB** sur le GPU
- Mandelbulb 4352^3 ~330 GB
  - Taille de brique : 64^3 => 360000 briques
  - Configuration 2 niveaux de virtualisation
    → Requiert **3 MB** sur le GPU

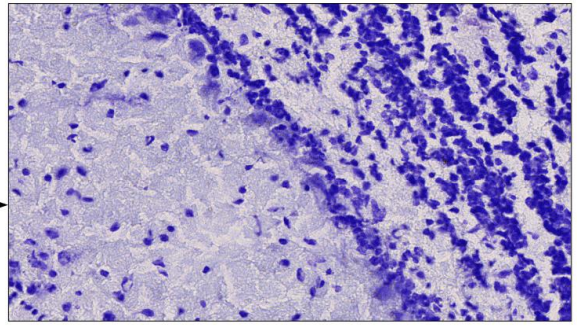
Nombre de caches

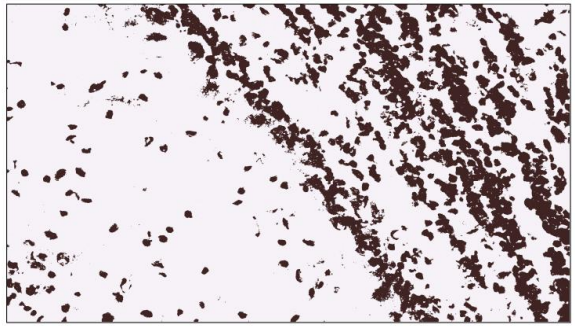
3 niveaux de virtualisation et blocs de 64^3

→ Adressage de ~70 milliards de briques

Résultats







Utilisation du GPU

- Env. 95% de temps libre

Steps	mean in µs
Brick loading	1996
LRU update	932
Request handling	443
Hierarchy update	195

Conclusion et perspectives

- Pipeline out-of-core avec interface sur GPU pour "tout type" d'application.
- Système permettant la **visualisation** et des **traitements** interactifs
  1. Utilisation d'une structure d'adressage virtuel out-of-core : hiérarchie de table de pagination
  2. Maintien de la structure en parallèle sur le GPU
- Communications entre CPU et GPU réduites
- Gestion de la multimodalité des données

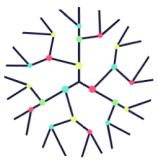
Perspectives

- Passage à l'échelle sur super-calculateur multi-GPUs multi-CPUs.
- Application de visualisation avec ray-casting volumique.
- Application de traitement avec deep-learning.

Merci

Questions ?

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE



3DNEUROSECURE
calcul intensif et simulation numérique