

GT Visualisation 2017

Une approche out-of-core entièrement basée GPU
pour la manipulation de gros volumes de données

J. Sarton¹, N. Courilleau^{1,2}, Florent Duguet^{1,3}, Y. Remion¹,
L. Lucas¹

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

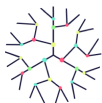
8 Juin 2017



Outline

- 1 Introduction
- 2 Structure de données
- 3 Visualization-driven pipeline
- 4 Résultats et discussion
- 5 Conclusion et perspectives

Motivations



3DNEUROSECURE

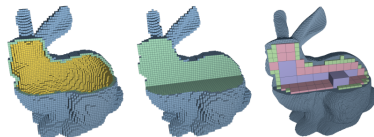
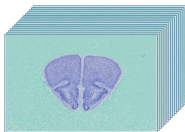
calcul intensif et simulation numérique

Objectif

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

En temps réel ! sur des **données volumique** → Utilisation de **GPU** :)

- Données médicales (scans, IRM ...)
- Voxelisation de maillage



[Schwarz et al., SIGGRAPH Asia, 2010]

- Grille régulière 3D → voxels



Problématique

Mémoire

Représentation voxels → de gros volumes de données.
Dépasse souvent la quantité mémoire GPU et même CPU.
Traitement en temps réel = compliqué.

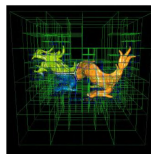
- Voxelisation : 4096^3 en RGBA → ~275 Go
- Histologie → quelques 100aines Go - quelques To
- Microscopie électronique → quelques 100aines de To - quelques Po

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

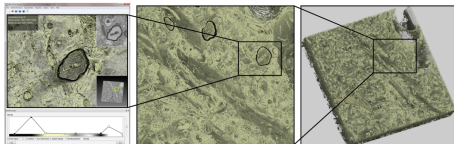
Etat de l'art

Contexte :

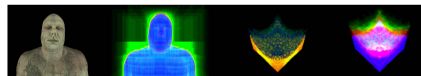
- Ray-casting volumique Out-of-core sur GPU.



[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

Un **pipeline out-of-core** complet du disque jusqu'au GPU pour **accéder, en temps interactif**, à de **très gros volumes** dépassant la quantité mémoire GPU ou CPU.

Hybridation de méthodes

- 1 Structure d'adressage out-of-core sur GPU : hiérarchie de table de pagination multi-résolution multi-niveaux [Hadwiger et al., IEEE SciVis 2012]
- 2 Gestion de la structure sur le GPU [Crassin, PhD Thesis, 2011]

Cadre

- General purpose (pas spécifique au ray-casting volumique)
- Multimodalité

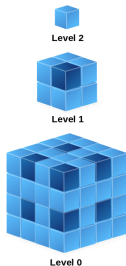
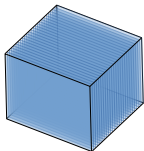
Representation et stockage des données

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

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

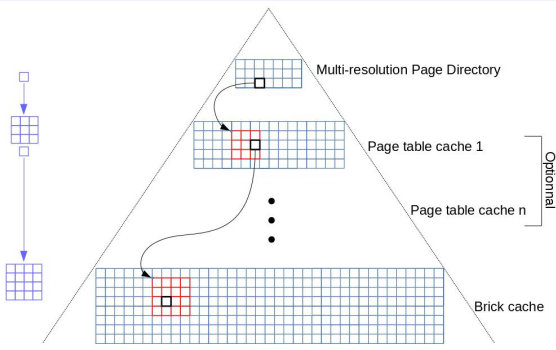
Mipmap 3D

- Pyramide 3D multi-résolution briquée.



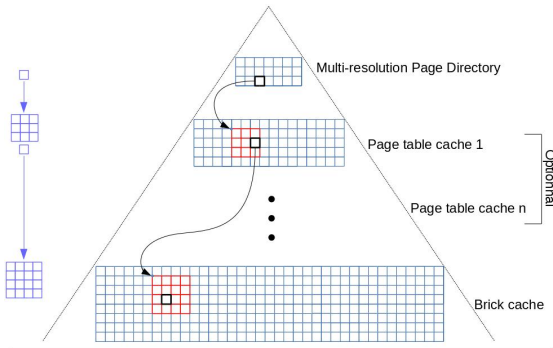
Stockage avec **compression** (Algorithme LZ4, décompression temps-réel).

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.

Hierarchie de table de pagination multirésolution multi-niveaux



- Implémentation : API surface de CUDA - texture 3D lecture/écriture
- Algorithme de remplacement des caches : Least Recently Used (LRU)

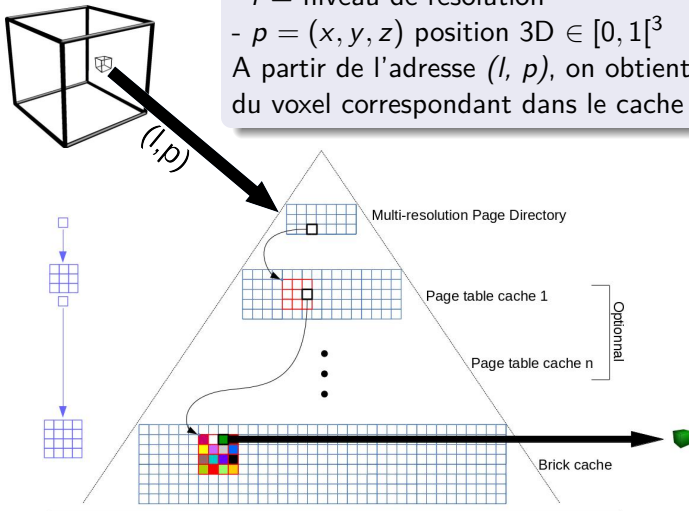
Adressage virtuel

Navigation dans un **volume virtuel** → 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.



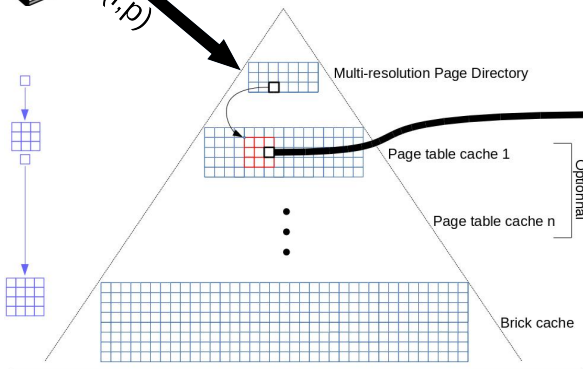
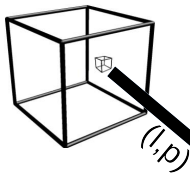
Cache miss

Navigation dans un **volume virtuel** → adresse (l, p)

- l = niveau de résolution

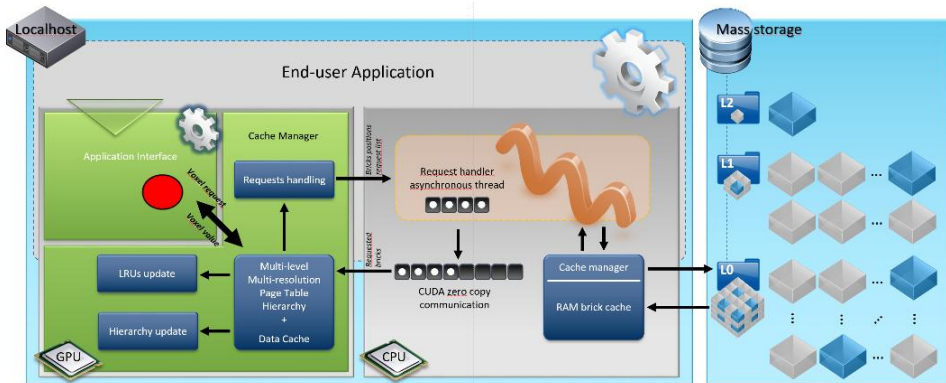
- $p = (x, y, z)$ position 3D $\in [0, 1]^3$

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

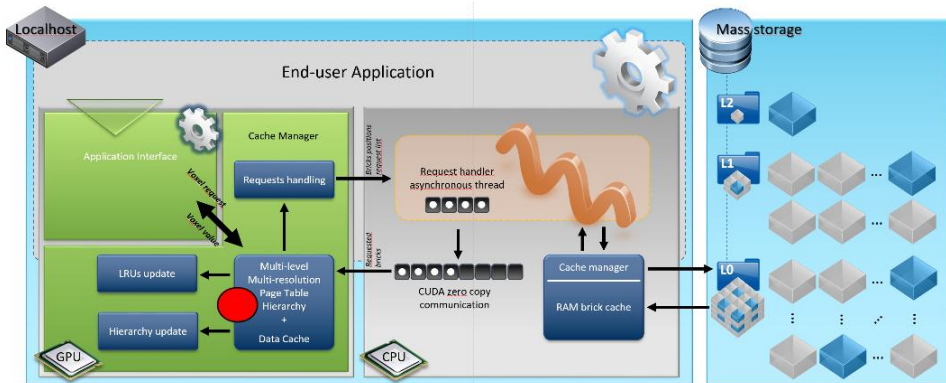


Cache miss !

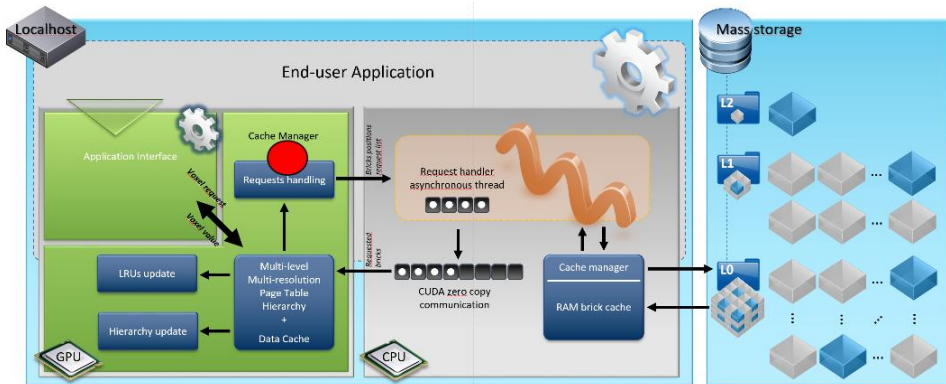
Pipeline



Pipeline



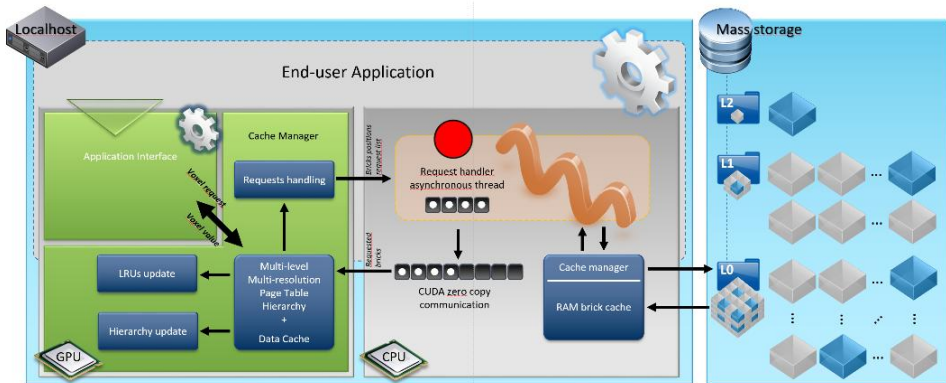
Pipeline



- Mise à jour

- 1 Gestion des requêtes de briques.

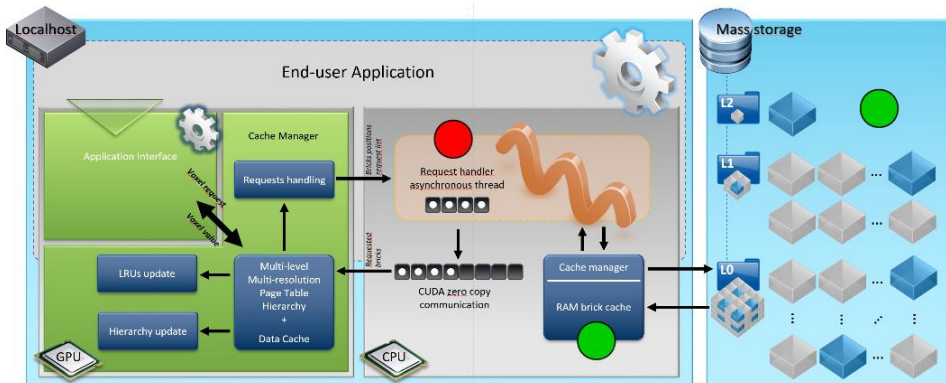
Pipeline



- Mise à jour

- 1 Gestion des requêtes de briques.

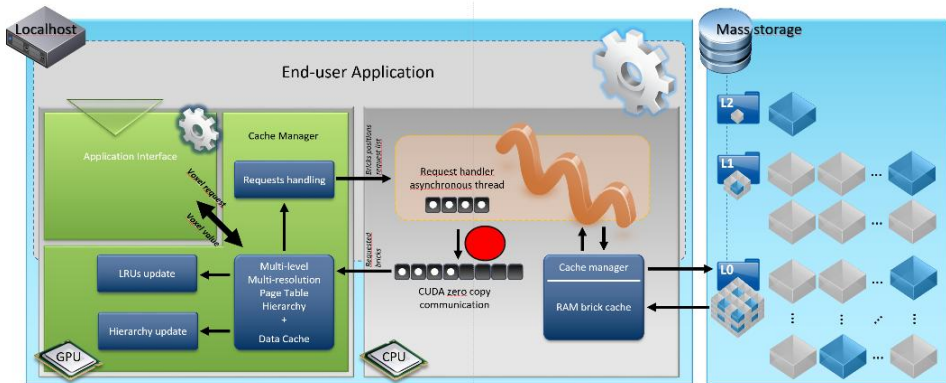
Pipeline



- Mise à jour

- 1 Gestion des requêtes de briques.

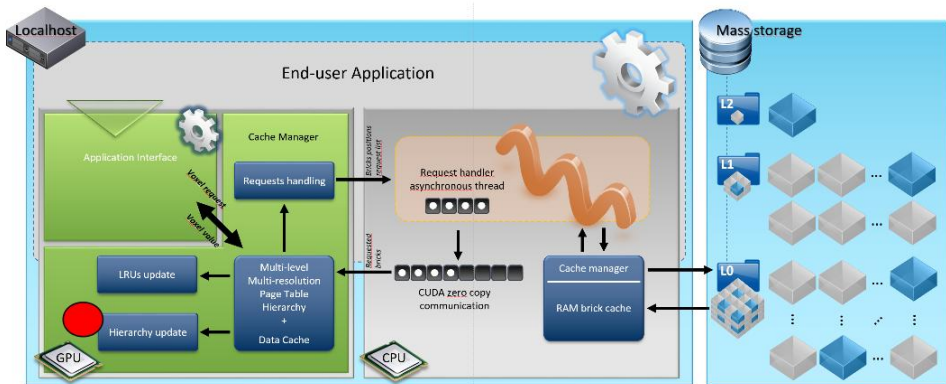
Pipeline



- Mise à jour

- 1 Gestion des requêtes de briques.

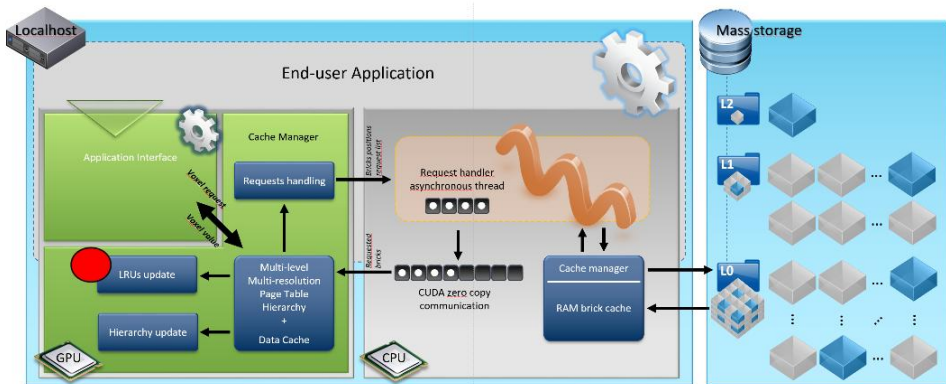
Pipeline



- 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

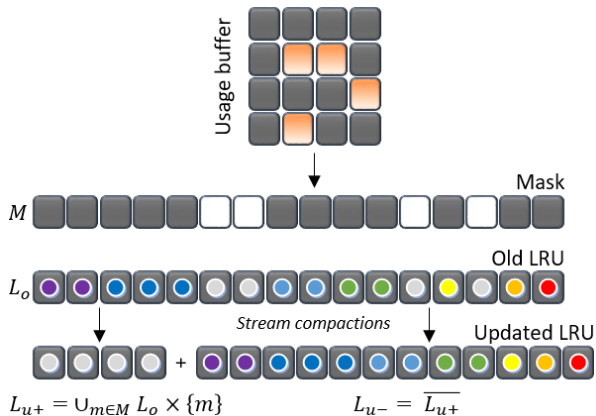
- 1 Mise à jour des LRUs
- 2 Gestion des requêtes de briques

Une gestion des caches GPU entièrement sur GPU !

Avantages

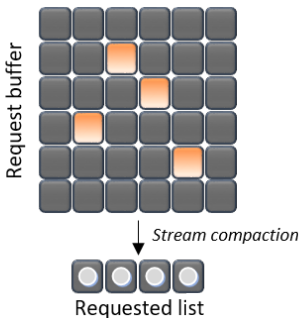
- Evite beaucoup de transferts de données entre CPU et GPU
- Permet de profiter de l'environnement massivement parallèle des GPUs

Mise à jour des LRUs



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

Gestion des requêtes de briques

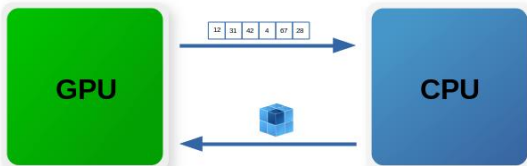


- Taille du nombre de briques dans le **volume**
- Marquage via un timestamp

Transferts entre CPU et GPU

Communications GPU → CPU

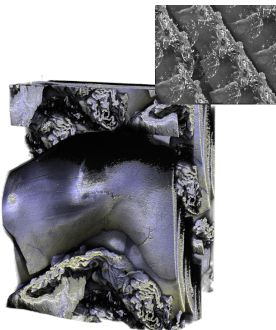
Une simple liste avec les identifiants des briques requêtées !



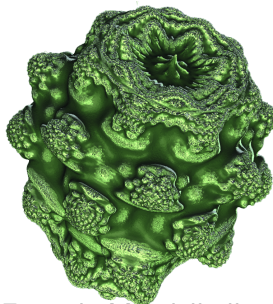
Communications GPU ← CPU

Uniquement les briques ! (Avec CUDA Zero Copy !)

Résultats



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



Fractale Mandelbulb
 4352^3 ~330 GB

Application

Visualiseur 2D : navigation (zoom et déplacement) avec rendu multirésolution 2D.

Rendu ~300 FPS.

Résultats

Utilisation mémoire GPU

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

Nombre de caches

3 niveaux de virtualisation et blocs de 64^3
→ Adressage de ~ 70 milliards de briques

Analyse de performance

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.
- Hybridation de 2 approches :
 - 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-CPU.
- Application de visualisation avec ray-casting volumique.
- Application de traitement avec deep-learning.

Merci

Questions ?

