Projet TOMOX

Accélération de la reconstruction tomographique sur processeurs graphiques (GPUs)

Nicolas GAC, Ali Mohammad-Djafari, Alexandre Vabre,
Fanny Buyens, Eric Tordjeman, Karim Tadrist

Colloque GPU du GDR MI2B & CERIMED - Obernai
"Calcul intensif sur carte graphique pour l'imagerie moléculaire"

28/29 Mai 2009

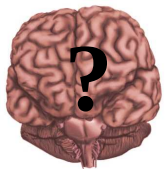
Projet TOMOX : Opération de Maturation Technico-Economique(OMTE)

Equipe opérationnelle

L2S	Ali Mohammad Djafari
CEA LIST	Alexandre Vabre, Fanny Buyens
Digiteo	Eric Tordjeman (Marketing)
	Nicolas Gac (Développement)
Supélec	Karim Tadrist (Juridique)

1. Objectif : accélérer la reconstruction tomographique
   - Projet TOMOX
   - Algorithme bayésien itératif du L2S/CEA
   - Accélération de la reconstruction
2. Tomographie sur GPU : Avant et Après CUDA
   - Avant CUDA ☹
   - Après CUDA ☺
3. Parallélisation des opérateurs de projection/rétroprojection
   - Localité des accès mémoire
   - Découpage en threads
   - Projecteur “voxel-driven” ou “ray-driven”
   - Temps GPU
4. Conclusion et perspectives

Projet TOMOX : Algorithmes innovants pour la reconstruction en nano-tomographie X



Objet à imager

IMAGEUR 3D HAUTE RESOLUTION

① ACQUISITION

− Plateforme de microtomographie X haute résolution

CEA−List

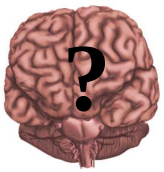
② RECONSTRUCTION

− Algorithme innovant (brevet)
− Accélération sur processeurs graphiques

L2S/Supelec/CNRS/UPS

Structure interne de l'objet

Projet TOMOX : Algorithmes innovants pour la reconstruction en nano-tomographie X



Objet à imager

IMAGEUR 3D HAUTE RESOLUTION

1 ACQUISITION

- Plateforme de microtomographie X haute résolution

CEA−List

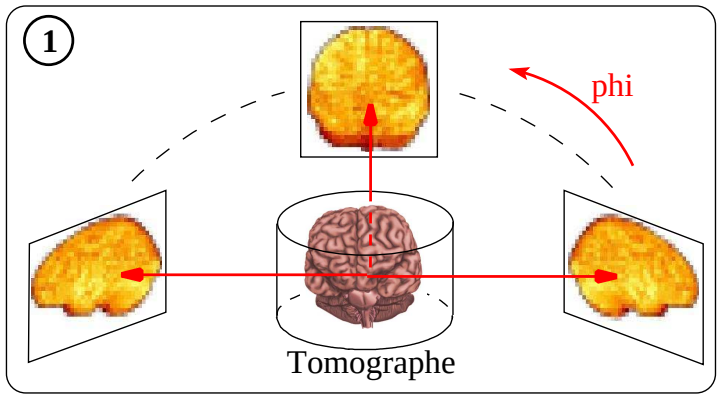
Projections

2 RECONSTRUCTION

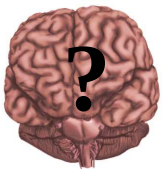
- Algorithme innovant (brevet)
- Accélération sur processeurs graphiques

L2S/Supelec/CNRS/UPS

Structure interne de l'objet



Projet TOMOX : Algorithmes innovants pour la reconstruction en nano-tomographie X



Objet à imager

IMAGEUR 3D HAUTE RESOLUTION

1 ACQUISITION

- Plateforme de microtomographie X haute résolution

CEA−List

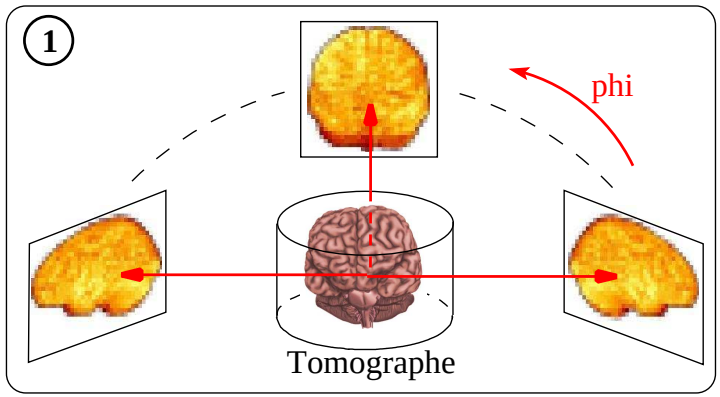
Projections

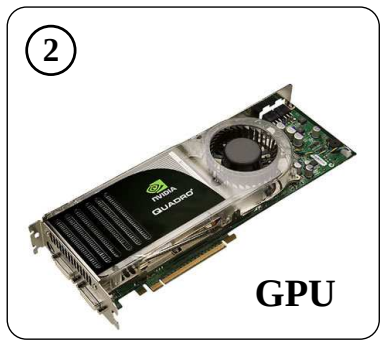
2 RECONSTRUCTION

- Algorithme innovant (brevet)
- Accélération sur processeurs graphiques

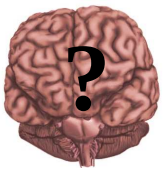
L2S/Supelec/CNRS/UPS

Structure interne de l'objet





Projet TOMOX : Algorithmes innovants pour la reconstruction en nano-tomographie X



Objet à imager

IMAGEUR 3D HAUTE RESOLUTION

1 ACQUISITION

− Plateforme de microtomographie X haute résolution

CEA−List

Projections

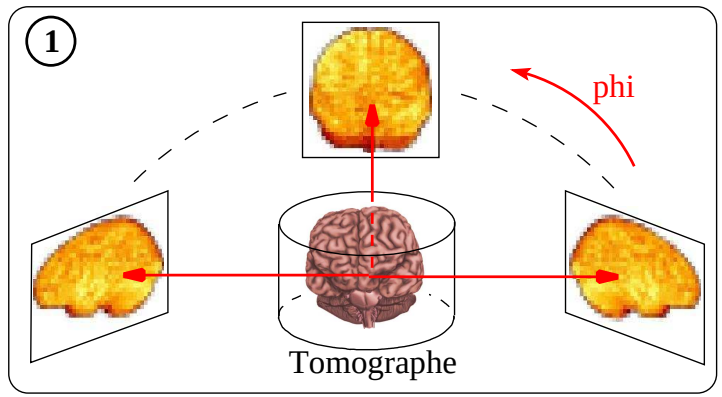
2 RECONSTRUCTION

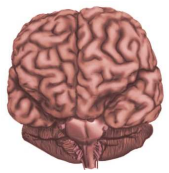
− Algorithme innovant (brevet)
− Accélération sur processeurs graphiques

L2S/Supelec/CNRS/UPS

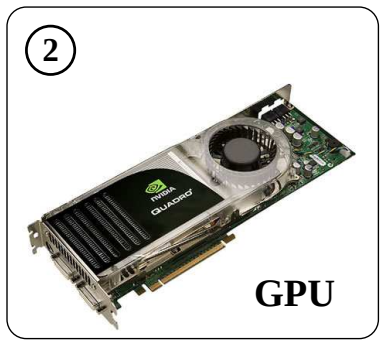
APPLICATIONS

− **Controle non destructif** (réacteurs, pipelines, pièces industrielles...)
− **Imagerie medicale**
− **Transport**
− **Agro−alimentaire**
− **Micro−électronique** (controle qualité)

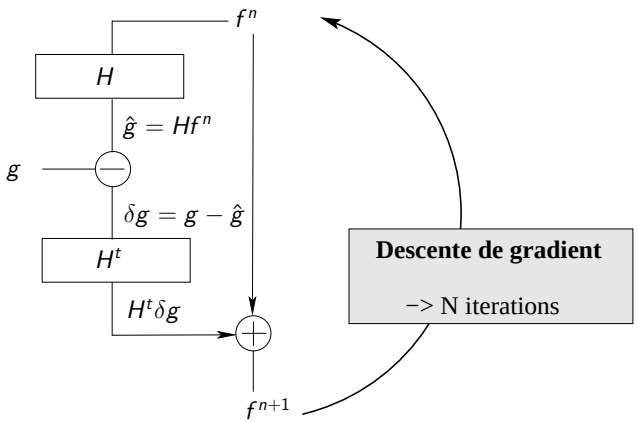




Structure interne de l'objet



SANS régularisation bayésienne



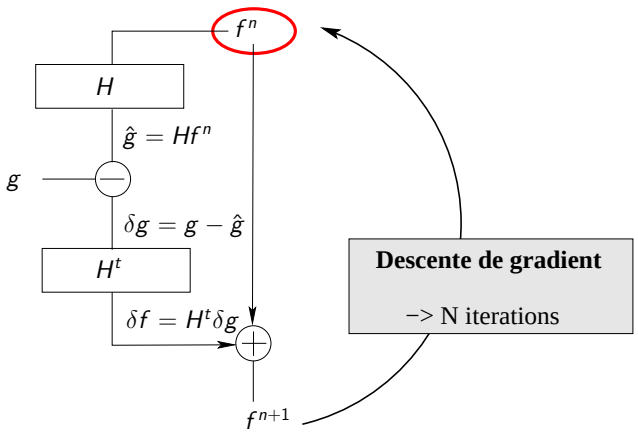
$\mathbf{g = Hf + \epsilon}$

f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

Descente de gradient

$$J(f) = ||g - Hf||^2$$
$$f^{n+1} = f^n - \alpha \cdot \nabla J(f^n)$$
$$\nabla J(f) = -2 \cdot H^t(g - Hf)$$

SANS régularisation bayésienne



f^n : **Estimée du volume**

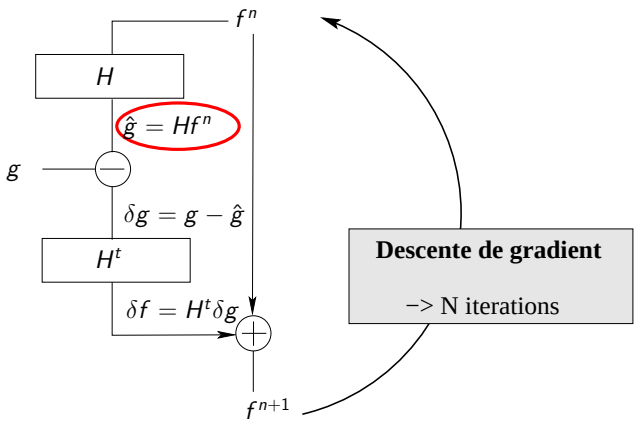
$g = Hf + \epsilon$

f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

Descente de gradient

$$J(f) = ||g - Hf||^2$$
$$f^{n+1} = f^n - \alpha \cdot \nabla J(f^n)$$
$$\nabla J(f) = -2 \cdot H^t(g - Hf)$$

SANS régularisation bayésienne



$\hat{g}$: **Estimée des données**

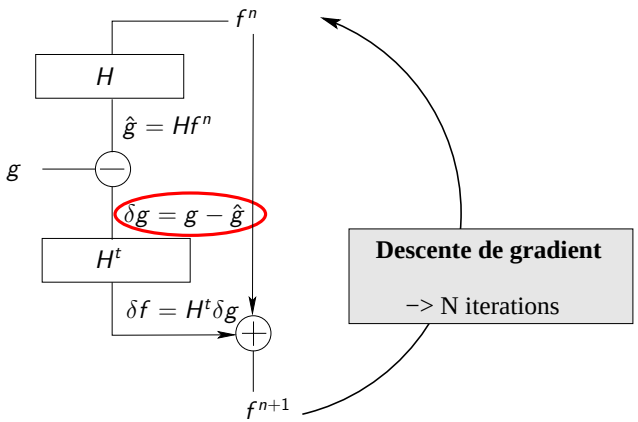
$g = Hf + \epsilon$

f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

Descente de gradient

$$J(f) = ||g - Hf||^2$$
$$f^{n+1} = f^n - \alpha \cdot \nabla J(f^n)$$
$$\nabla J(f) = -2 \cdot H^t(g - Hf)$$

SANS régularisation bayésienne



δg : **Correction des données**

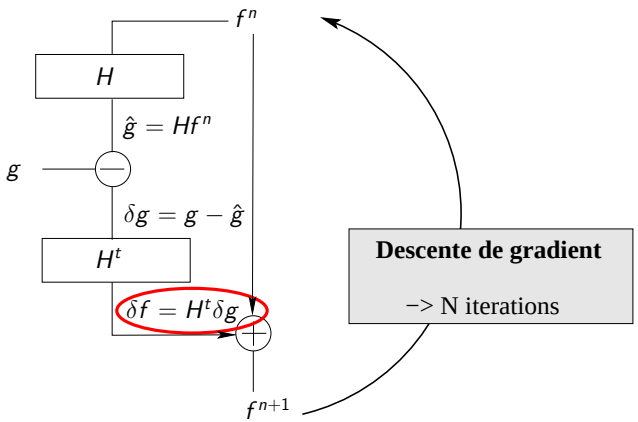
$\mathbf{g = Hf + \epsilon}$

f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

Descente de gradient

$$J(f) = ||g - Hf||^2$$
$$f^{n+1} = f^n - \alpha \cdot \nabla J(f^n)$$
$$\nabla J(f) = -2 \cdot H^t(g - Hf)$$

SANS régularisation bayésienne



δf : **Correction du volume**

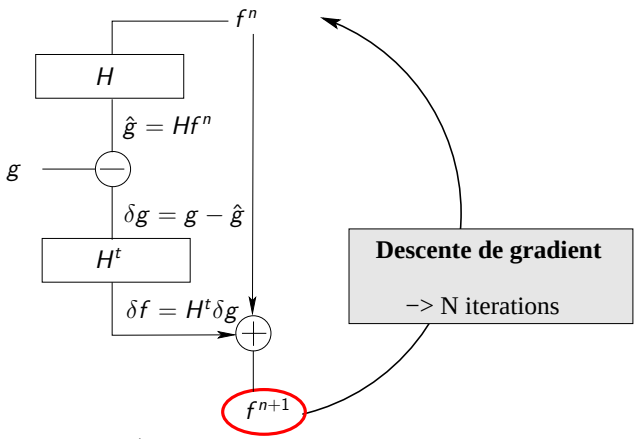
$\mathbf{g = Hf + \epsilon}$

f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

Descente de gradient

$$J(f) = ||g - Hf||^2$$
$$f^{n+1} = f^n - \alpha \cdot \nabla J(f^n)$$
$$\nabla J(f) = -2 \cdot H^t(g - Hf)$$

SANS régularisation bayésienne



f^{n+1} : **Nouvelle estimée du volume**

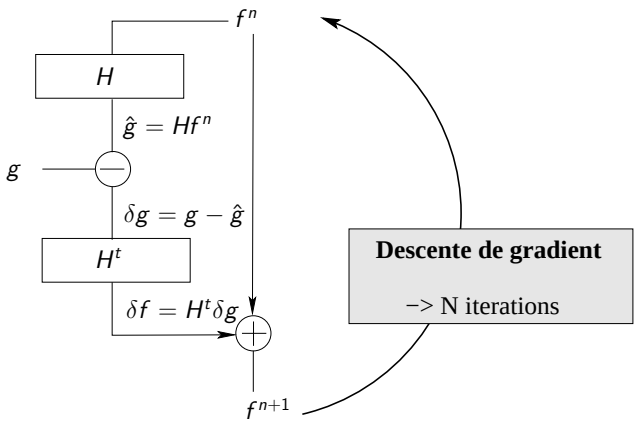
$\mathbf{g = Hf + \epsilon}$

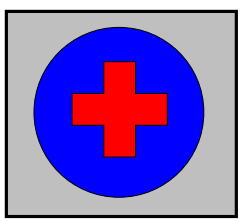
f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

Descente de gradient

$$J(f) = ||g - Hf||^2$$
$$f^{n+1} = f^n - \alpha \cdot \nabla J(f^n)$$
$$\nabla J(f) = -2 \cdot H^t(g - Hf)$$

AVEC régularisation bayésienne





Segmentation z

- $z = 0$: air
- $z = 1$: eau
- $z = 2$: metal

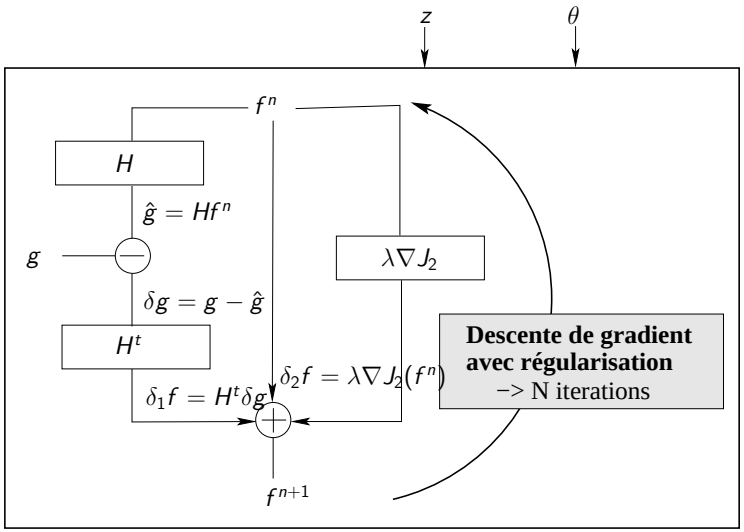
$\mathbf{g} = \mathbf{Hf} + \epsilon$

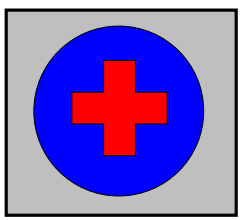
f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

Modèle a priori : $\mathbf{f} = \cup_k \mathbf{f_k}$

$f_k = \{r : z(r) = k\}$
$f(r)|z(r) = k \sim \mathcal{N}(m_k, \sigma_k^2)$

AVEC régularisation bayésienne





Segmentation z

- $z = 0$: air
- $z = 1$: eau
- $z = 2$: metal

$\mathbf{g = Hf + \epsilon}$

f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

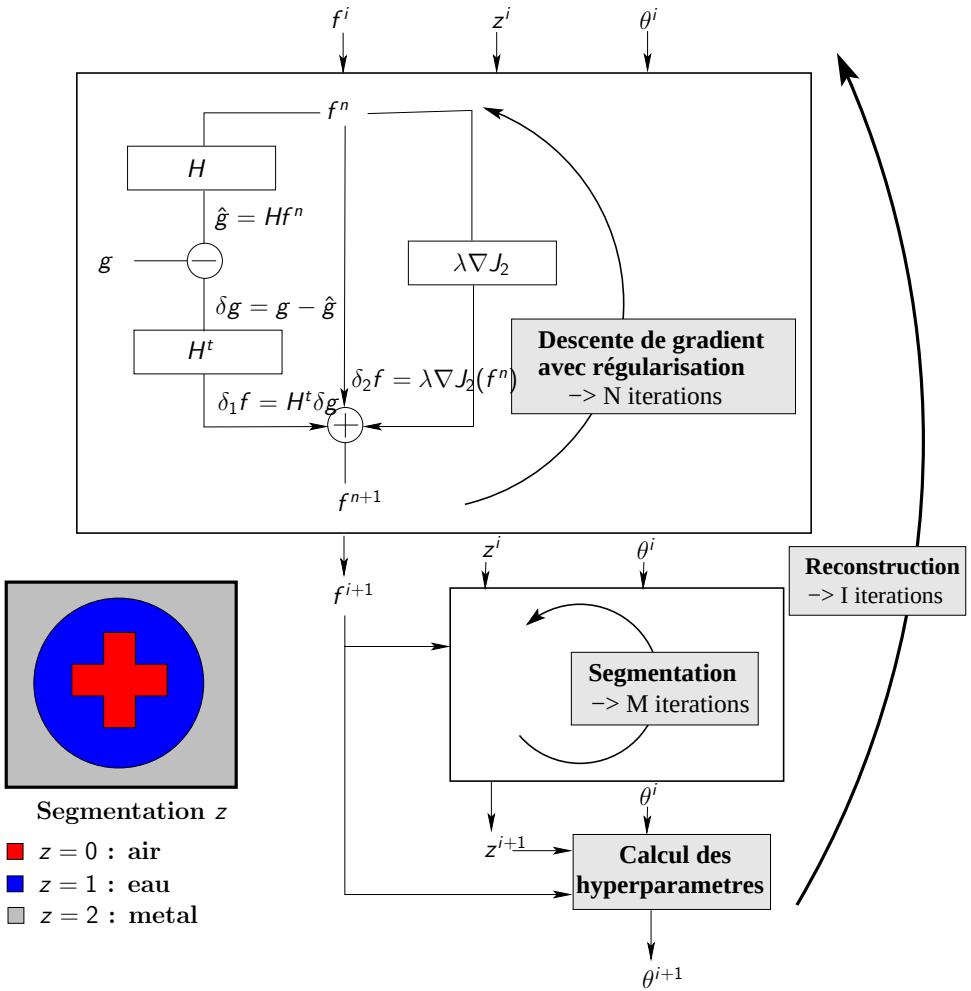
Modèle a priori : $\mathbf{f = \cup_k f_k}$

$f_k = \{r : z(r) = k\}$
$f(r)|z(r) = k \sim \mathcal{N}(m_k, \sigma_k^2)$

Descente de gradient régularisée

$J(f) = J_1(f) + J_2(f)$
$J_1(f) = ||g - Hf||^2$
$J_2(f|z,\theta) = \lambda \sum_k \sum_{r\in\mathcal{R}_k} \sum_{r'\in\mathcal{V}(r)} \phi(\bar{f}(r) - \bar{f}(r'))$
$\bar{f}(r) = \frac{f(r) - m_k}{\sigma_k}$
$f^{n+1} = f^n - \alpha_1 \cdot \nabla J_1(f^n) - \alpha_2 \cdot \nabla J_2(f^n|z,\theta)$

AVEC régularisation bayésienne



Segmentation z

- $z = 0$: air
- $z = 1$: eau
- $z = 2$: metal

$\mathbf{g = Hf + \epsilon}$

f : volume
g : données du tomographe
H : modèle d'acquisition
ϵ : bruit

Modèle a priori : $\mathbf{f = \cup_k f_k}$

$$f_k = \{r : z(r) = k\}$$
$$f(r)|z(r) = k \sim \mathcal{N}(m_k, \sigma_k^2)$$

Descente de gradient régularisée

$$J(f) = J_1(f) + J_2(f)$$
$$J_1(f) = ||g - Hf||^2$$
$$J_2(f|z,\theta) = \lambda \sum_k \sum_{r \in \mathcal{R}_k} \sum_{r' \in \mathcal{V}(r)} \phi(\bar{f}(r) - \bar{f}(r'))$$
$$\bar{f}(r) = \frac{f(r) - m_k}{\sigma_k}$$
$$f^{n+1} = f^n - \alpha_1 \cdot \nabla J_1(f^n) - \alpha_2 \cdot \nabla J_2(f^n|z,\theta)$$

Nécessité d'accélérer la reconstruction

Un problème de plus en plus complexe

- Amélioration de la résolution spatiale des scanners
  ➲ Objectif : *Volume de* $\simeq 2048^3$ *voxels*
- Utilisation de méthodes itératives
  ➲ *10 à 100 itérations nécéssaires*
- Reconstruction dynamique
  ➲ *plusieurs frames à reconstruire*

Temps de reconstruction insuffisant sur PCs

- Plusieurs heures voire jours de calcul

Calcul de Hf et $H^t \delta g$: choix de la méthode

❶ Calcul matriciel

➲ lecture des coefficients h_{ij} dans la mémoire SDRAM

⚠ volume $2048^3 -> $ matrice H $= 1$ To !

Calcul de Hf et $H^t \delta g$: choix de la méthode

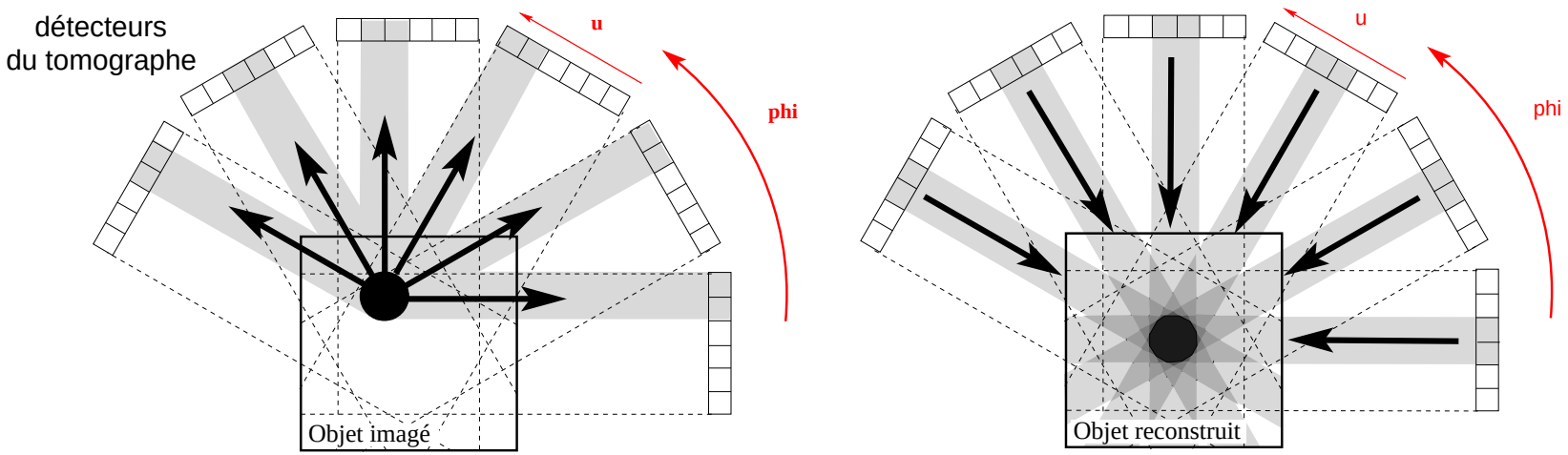
❶ Calcul matriciel

➲ lecture des coefficients h_{ij} dans la mémoire SDRAM
⚠ volume $2048^3 -> $ matrice H $= 1$ To !

❷ Opérateurs géométriques

➲ calcul en ligne des coefficients h_{ij}

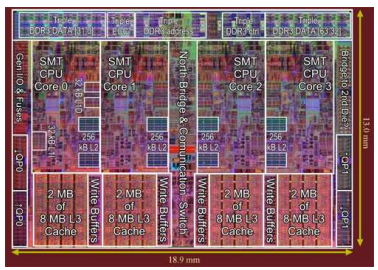
Paire de projection/rétroprojection en tomographie à émission (géométrie parallèle)



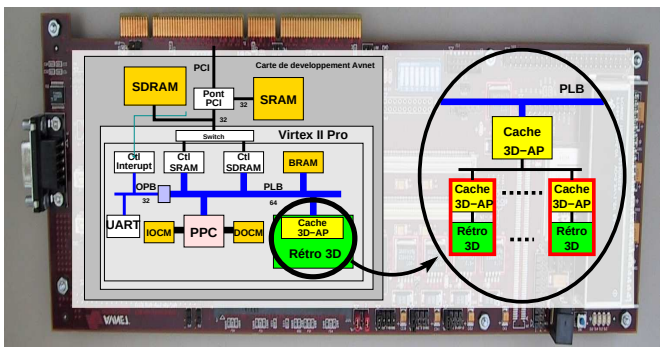
Calcul de Hf et $H^t\delta g$: choix du matériel

High Performance Computing (HPC)

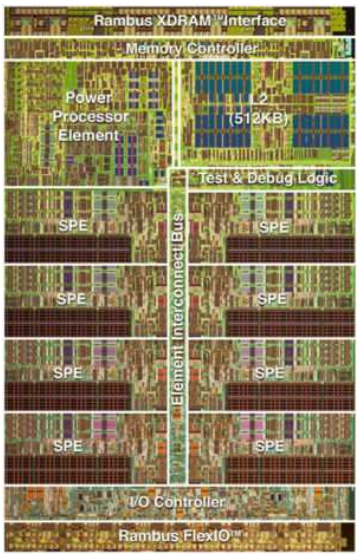
- Parallélisation sur machines multi-processeurs
  ➲ Efficace sur machine à mémoire distribuée
- Noeuds de calculs performants
  ➲ processeurs multi-core, many-core ou FPGA/ASIC



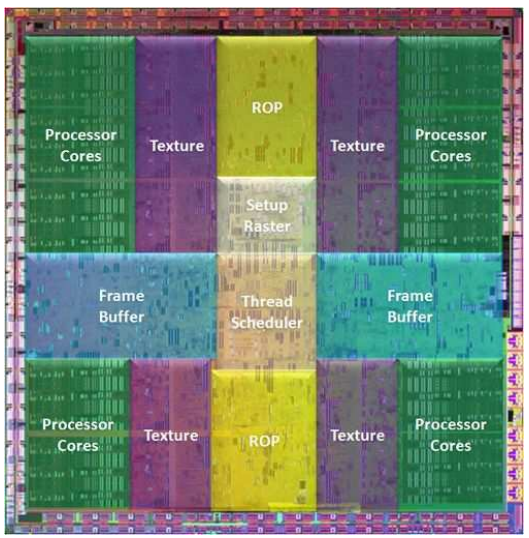
Intel Nehalem (4 coeurs)



SoPC (prototypage)

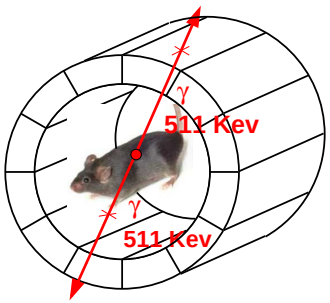


IBM Cell (8+1 coeurs)



Nvidia GTX 200 (240 coeurs)

Thèse : "Adéquation Algorithme Architecture pour la reconstruction 3D en imagerie médicale TEP" (Gipsa-lab, Grenoble-INP)



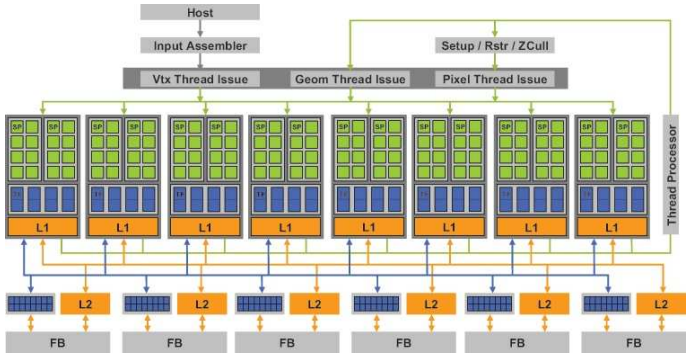
① Injection du traceur ② Acquisition

③ Enregistrement des données (Sinogrammes)

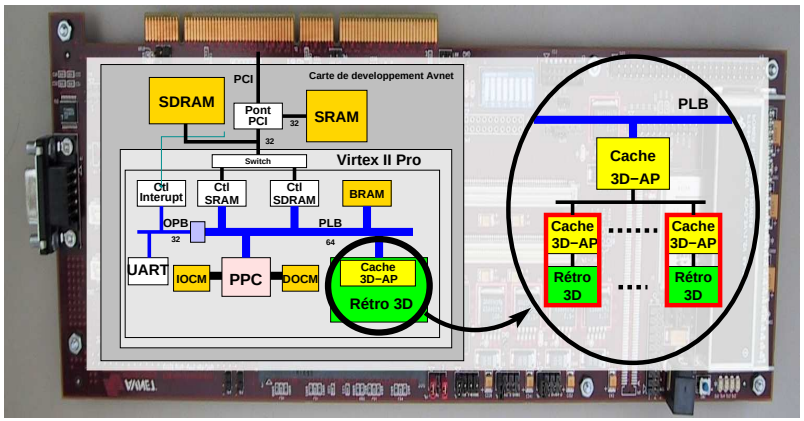
④ Reconstruction

⑤ Visualisation

Tomographie TEP



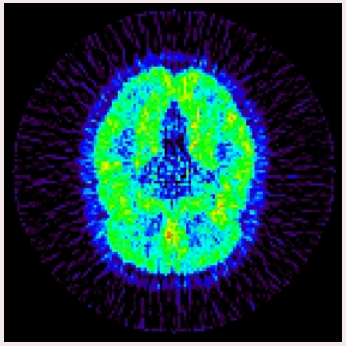
GPU (carte graphique)

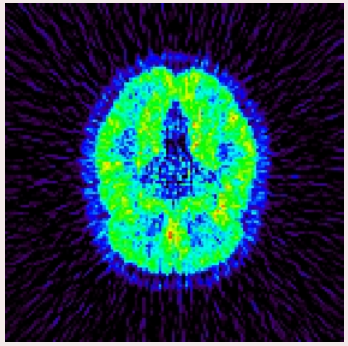


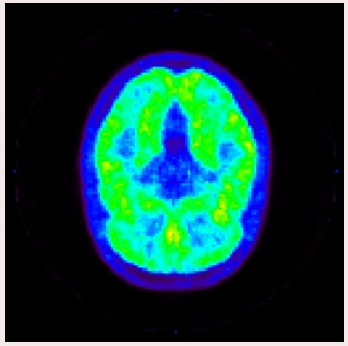
SoPC (prototypage)

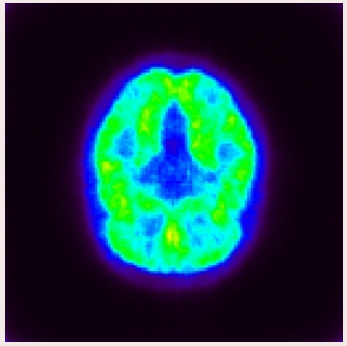
Exploration architecturale

Conclusion de la thèse


CPU
(non itératif)


FPGA
(non itératif)


CPU
(itératif)

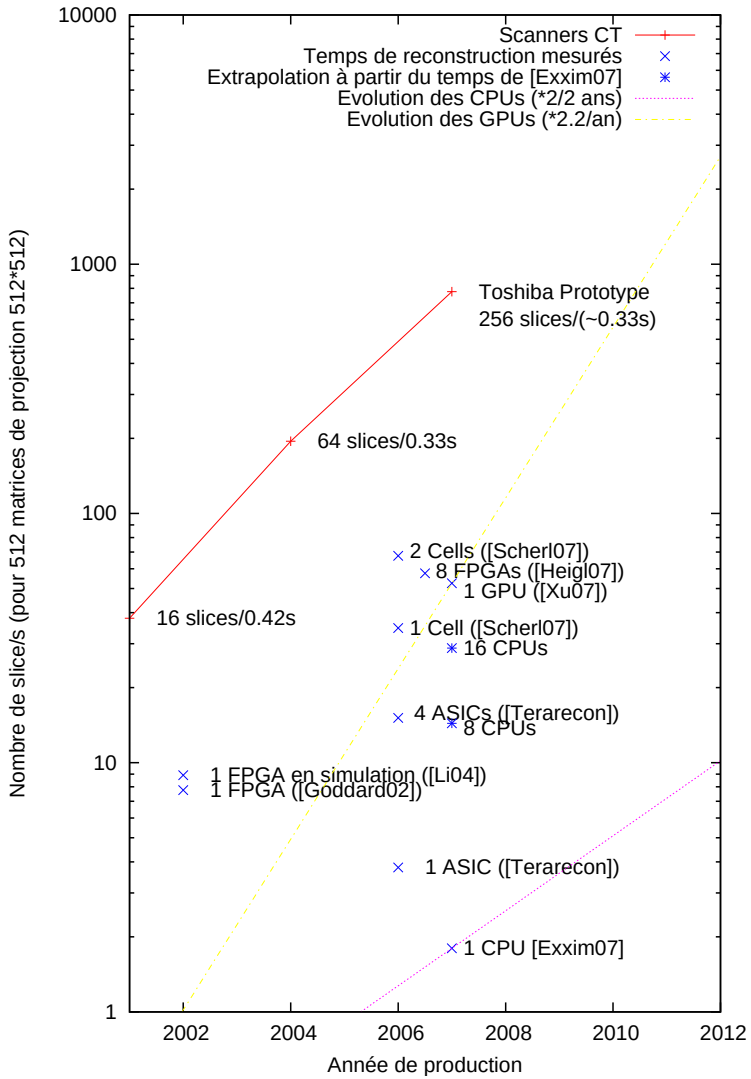

FPGA
(itératif)

Comparaison CPU/GPU/FPGA

	CPU	**GPU**	**FPGA**
Temps	3^{eme} (*4 P4)	1^{er} (*50 P4)	2^{eme} (*5 P4)
Efficacité	2^{eme} (7 C/op)	3^{ieme} (14 C/Op)	1^{er} (2 C/Op)

- **GPU est l'accélérateur matériel le plus performant**
- **Validation de notre stratégie d'accès mémoire (cache 3D)**

Vitesse d'acquisition // Vitesse de reconstruction



Avant CUDA : pipeline graphique

Vertex Shader

Transformation géométrique

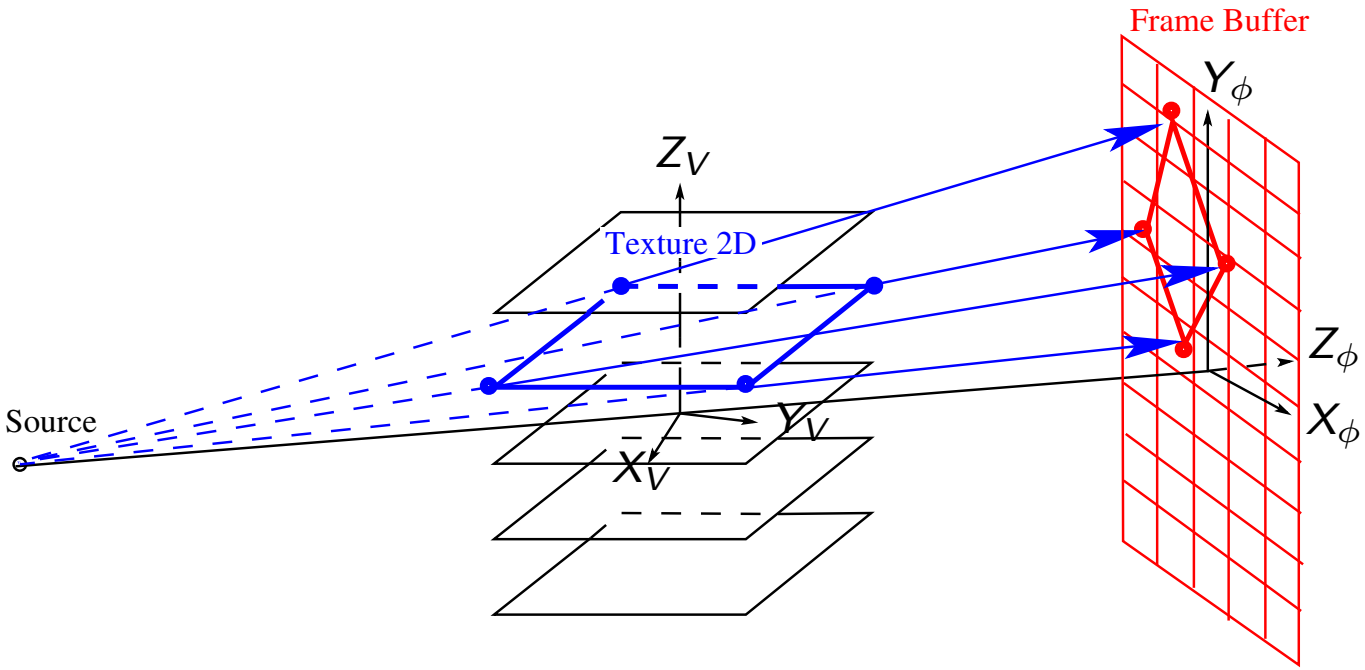
Rasterization

Polygon $->$ Fragments

Fragment Shader

Calcul sur les Pixels

Projection Cone Beam 3D avec le Pipeline Graphique



Vertex Shader

Transformation géométrique

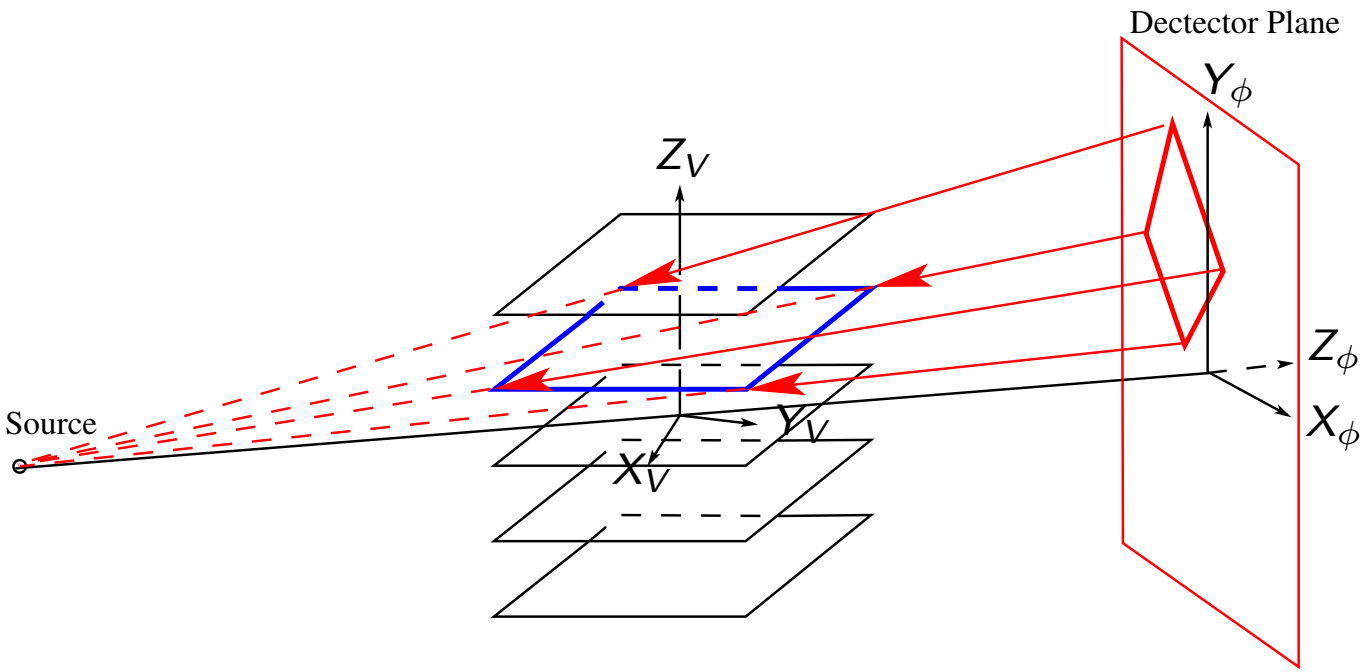
Rasterization

Polygon $->$ Fragments

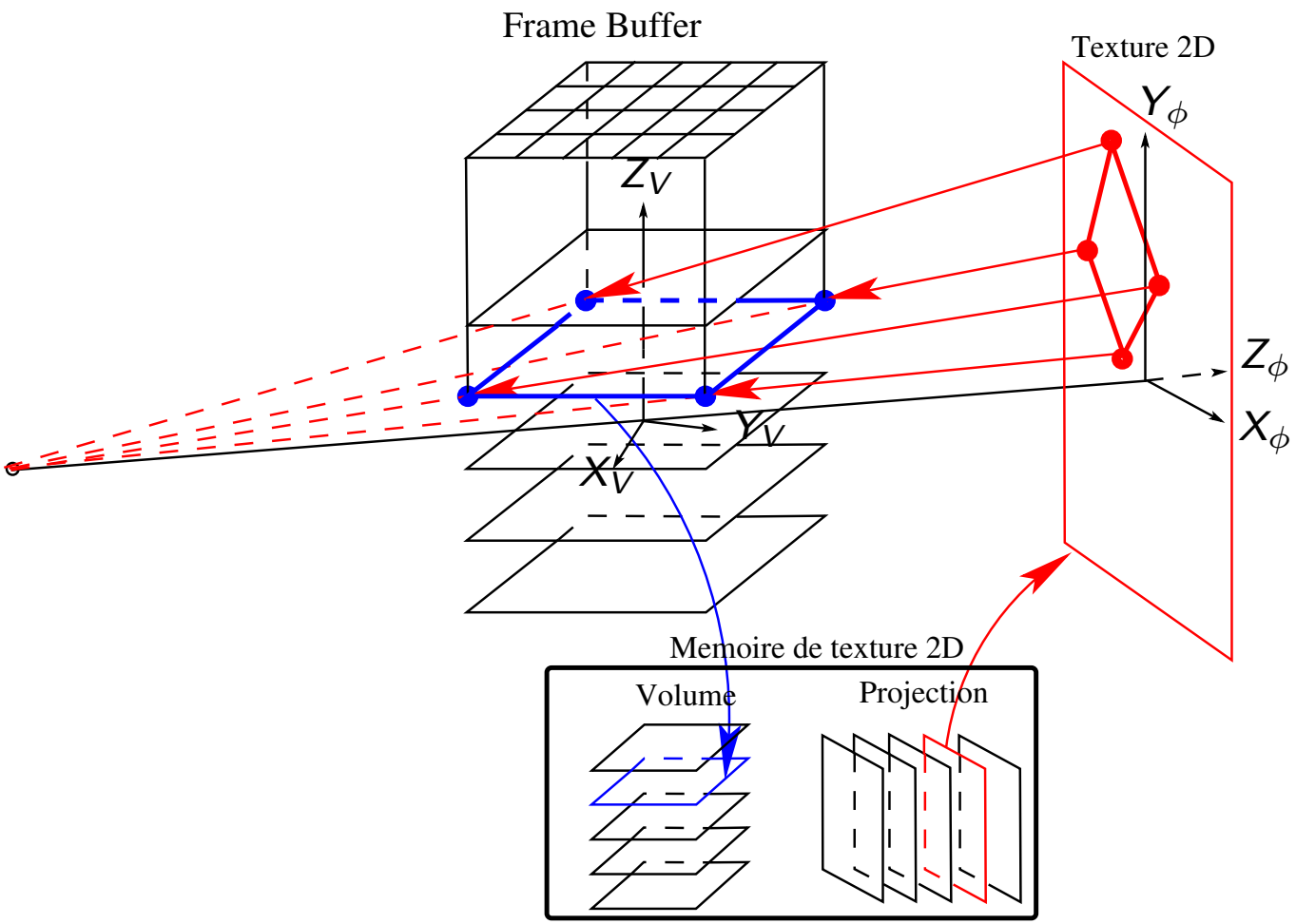
Fragment Shader

Calcul sur les Pixels

Rétroprojection Cone Beam 3D



Rétroprojection Cone Beam 3D avec le Pipeline Graphique



Vertex Shader

Transformation géométrique

Rasterization

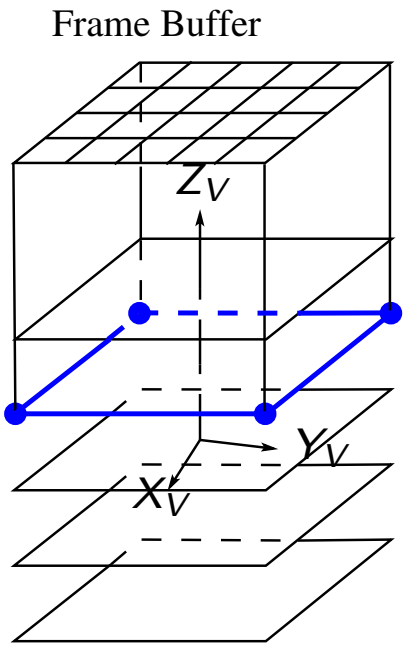
Polygon $->$ Fragments

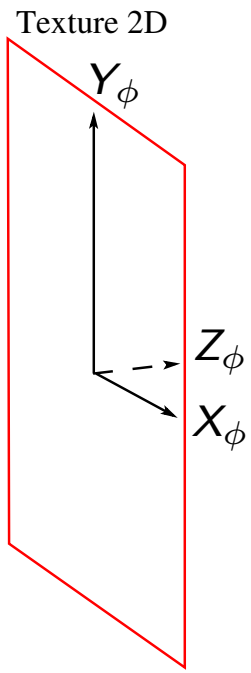
Fragment Shader

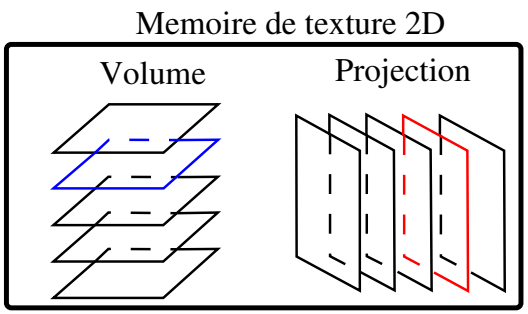
Calcul sur les Pixels

Rétroprojection Cone Beam 3D avec le Pipeline Graphique

Source







Projection des Vertex

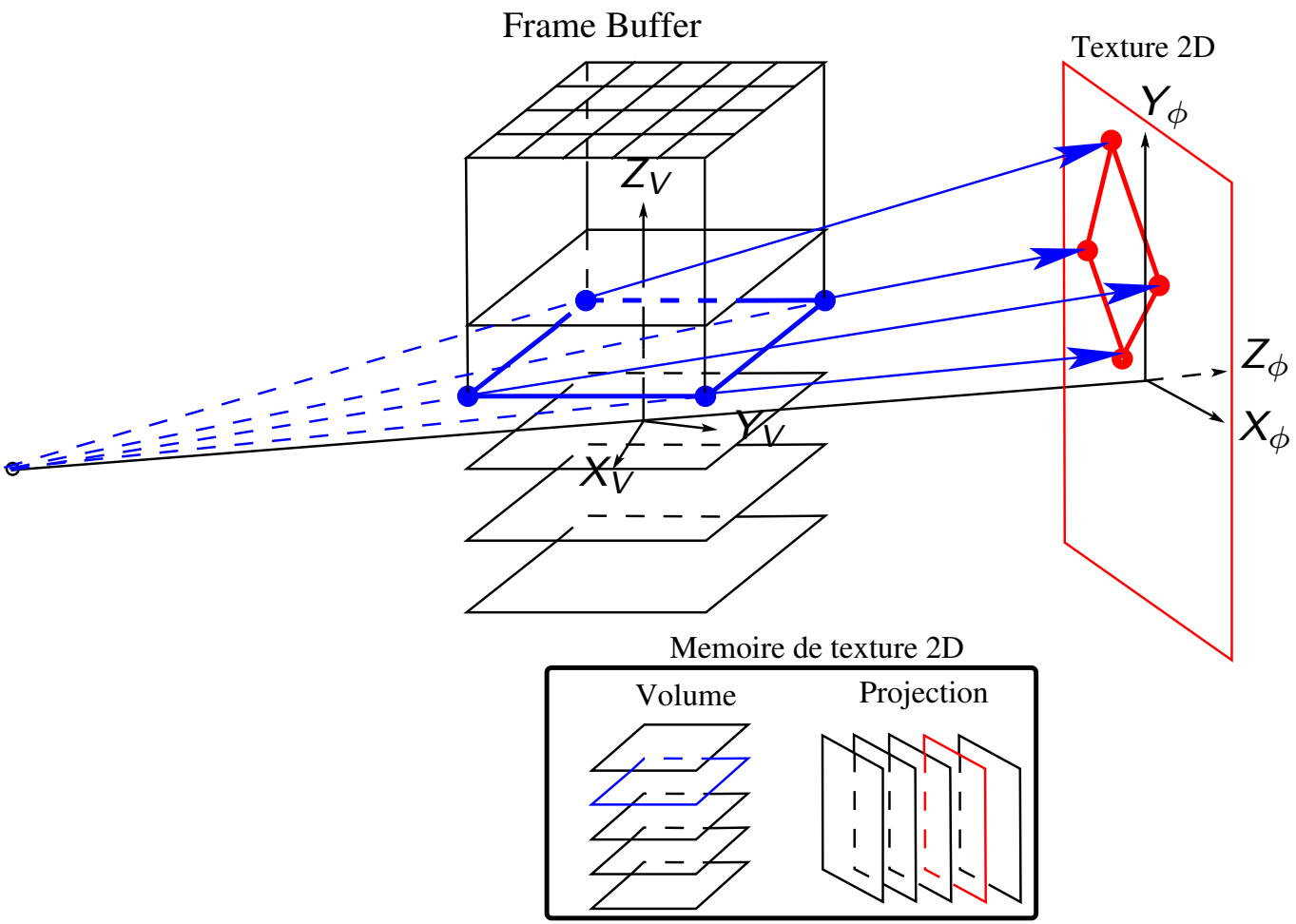
Projection des Fragments

Indice de Texture

Echantillonage de texture

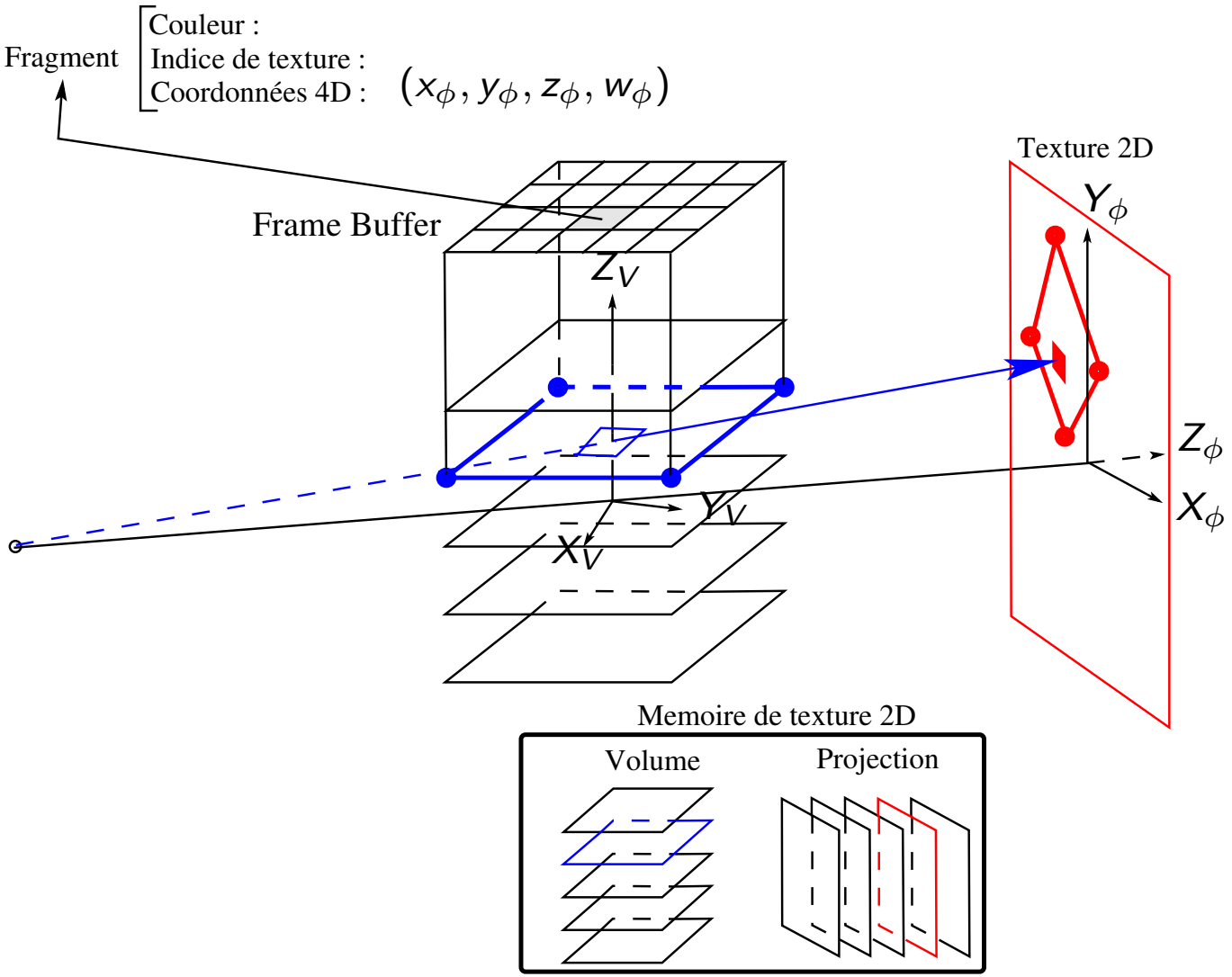
Accumulation

Rétroprojection Cone Beam 3D avec le Pipeline Graphique



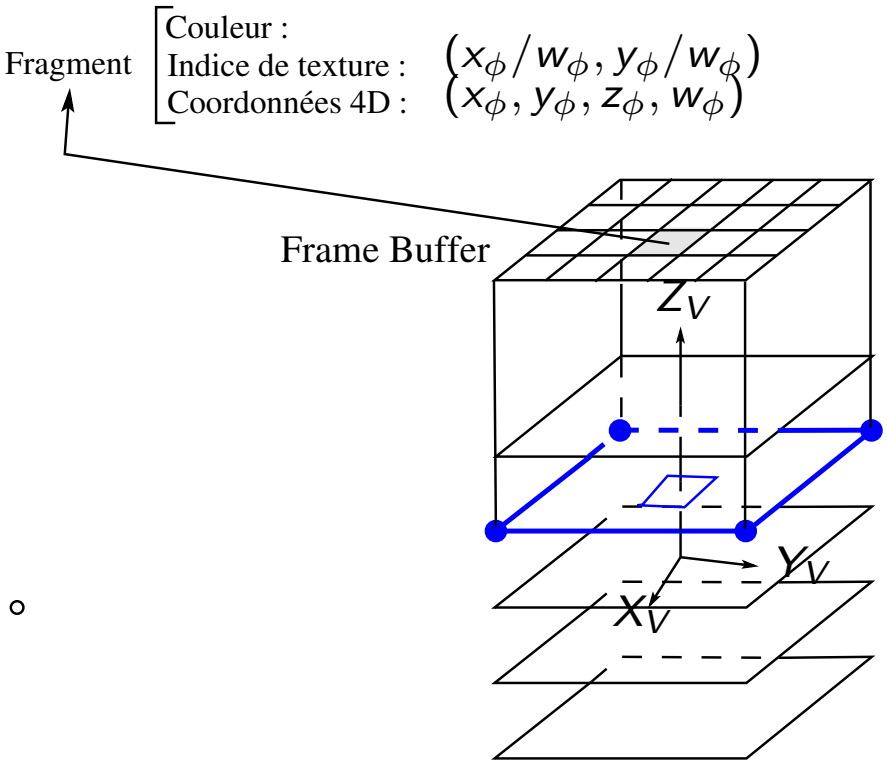
- Projection des Vertex
- Projection des Fragments
- Indice de Texture
- Echantillonage de texture
- Accumulation

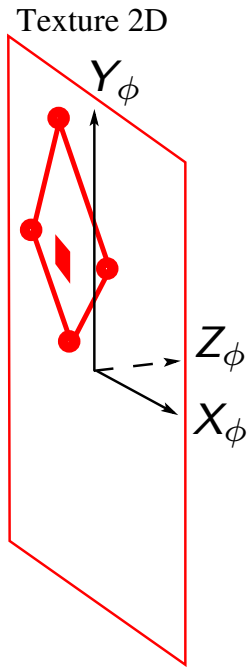
Rétroprojection Cone Beam 3D avec le Pipeline Graphique

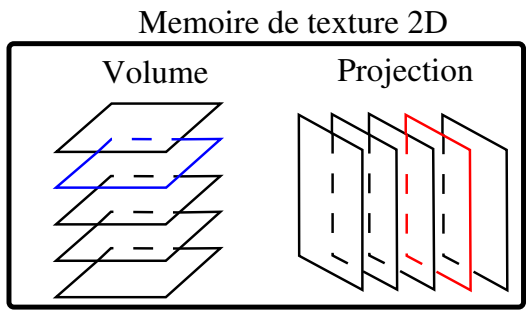


- Projection des Vertex
- Projection des Fragments
- Indice de Texture
- Echantillonage de texture
- Accumulation

Rétroprojection Cone Beam 3D avec le Pipeline Graphique

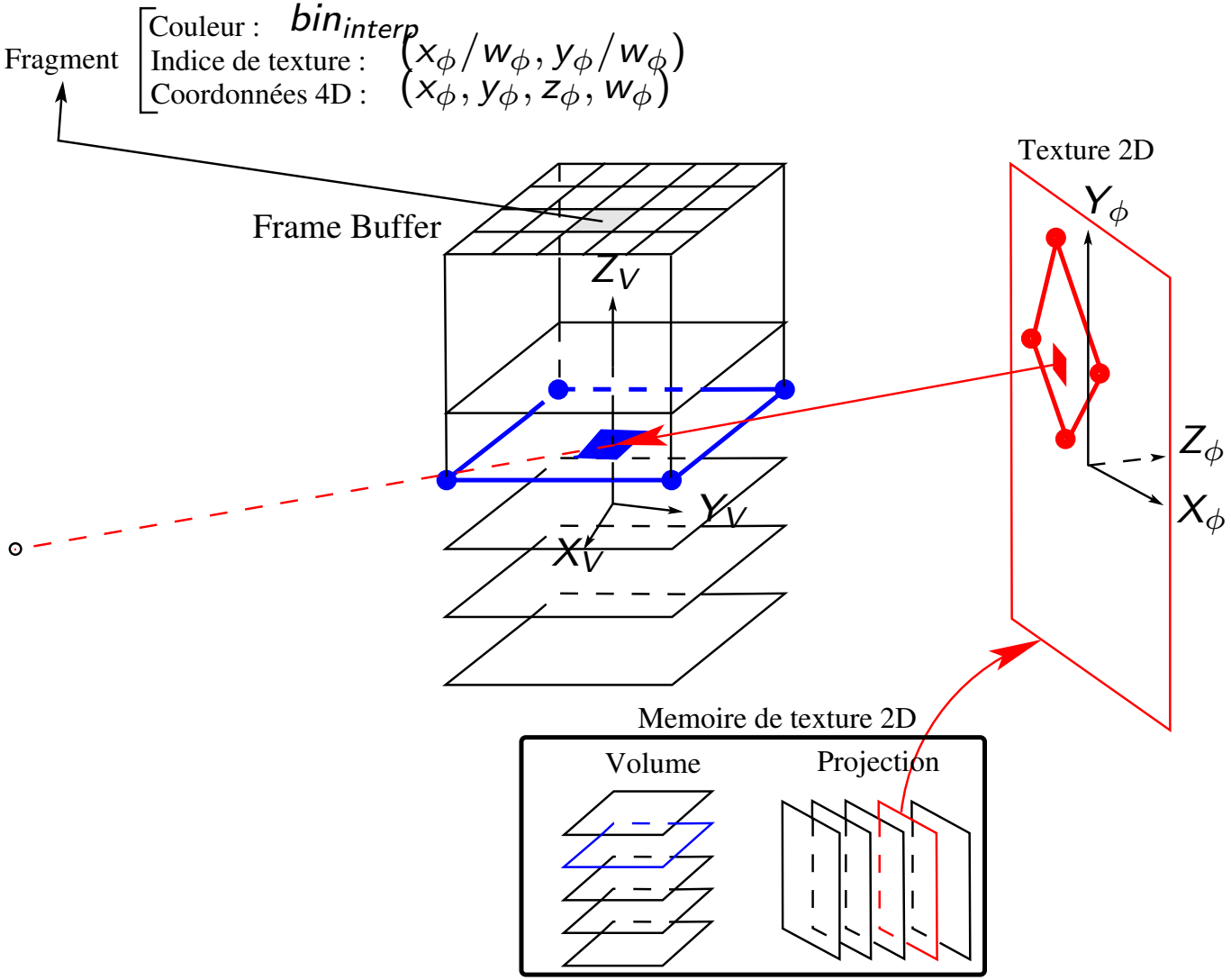






- Projection des Vertex
- Projection des Fragments
- **Indice de Texture**
- Echantillonage de texture
- Accumulation

Rétroprojection Cone Beam 3D avec le Pipeline Graphique



Projection des Vertex

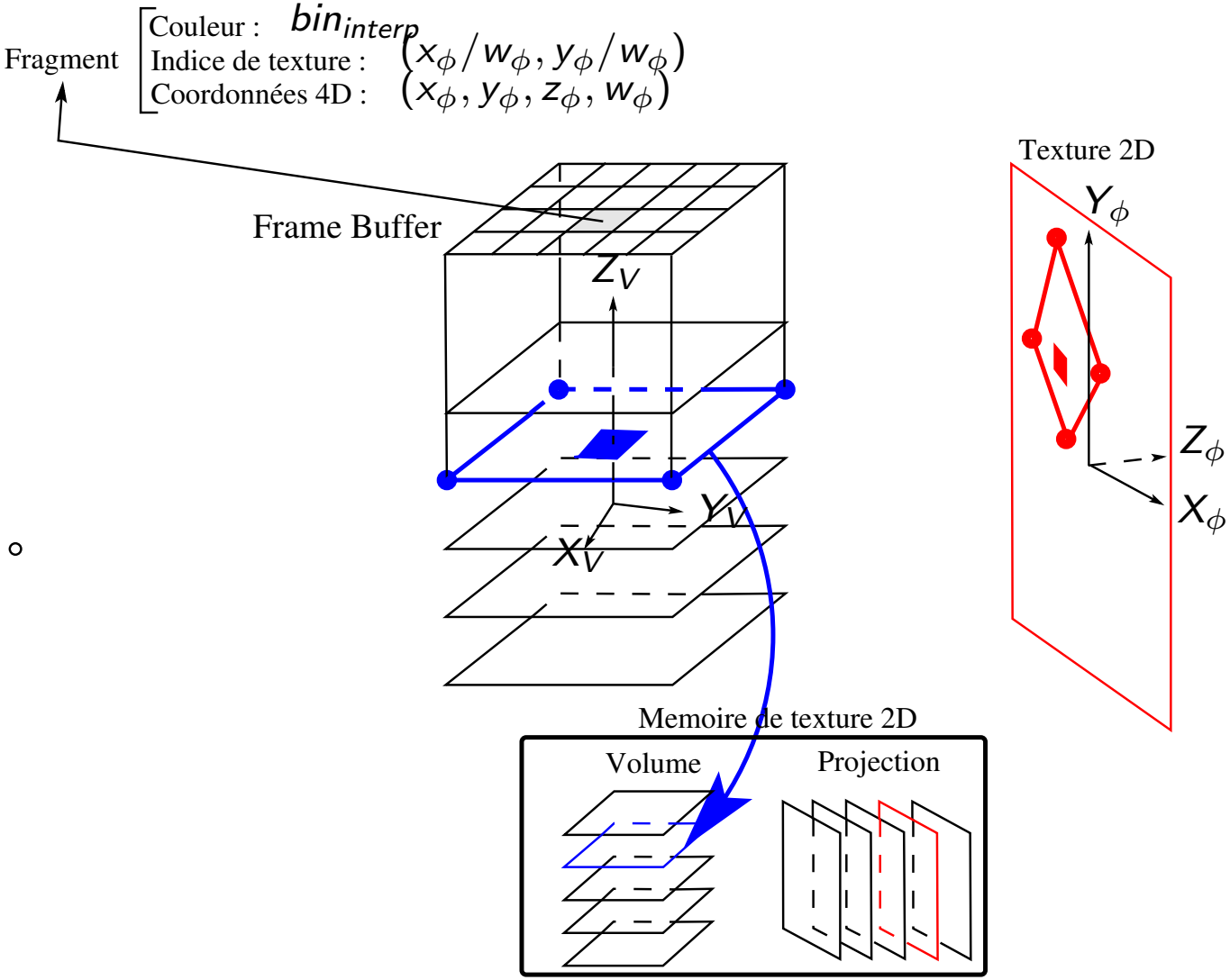
Projection des Fragments

Indice de Texture

Echantillonage de texture

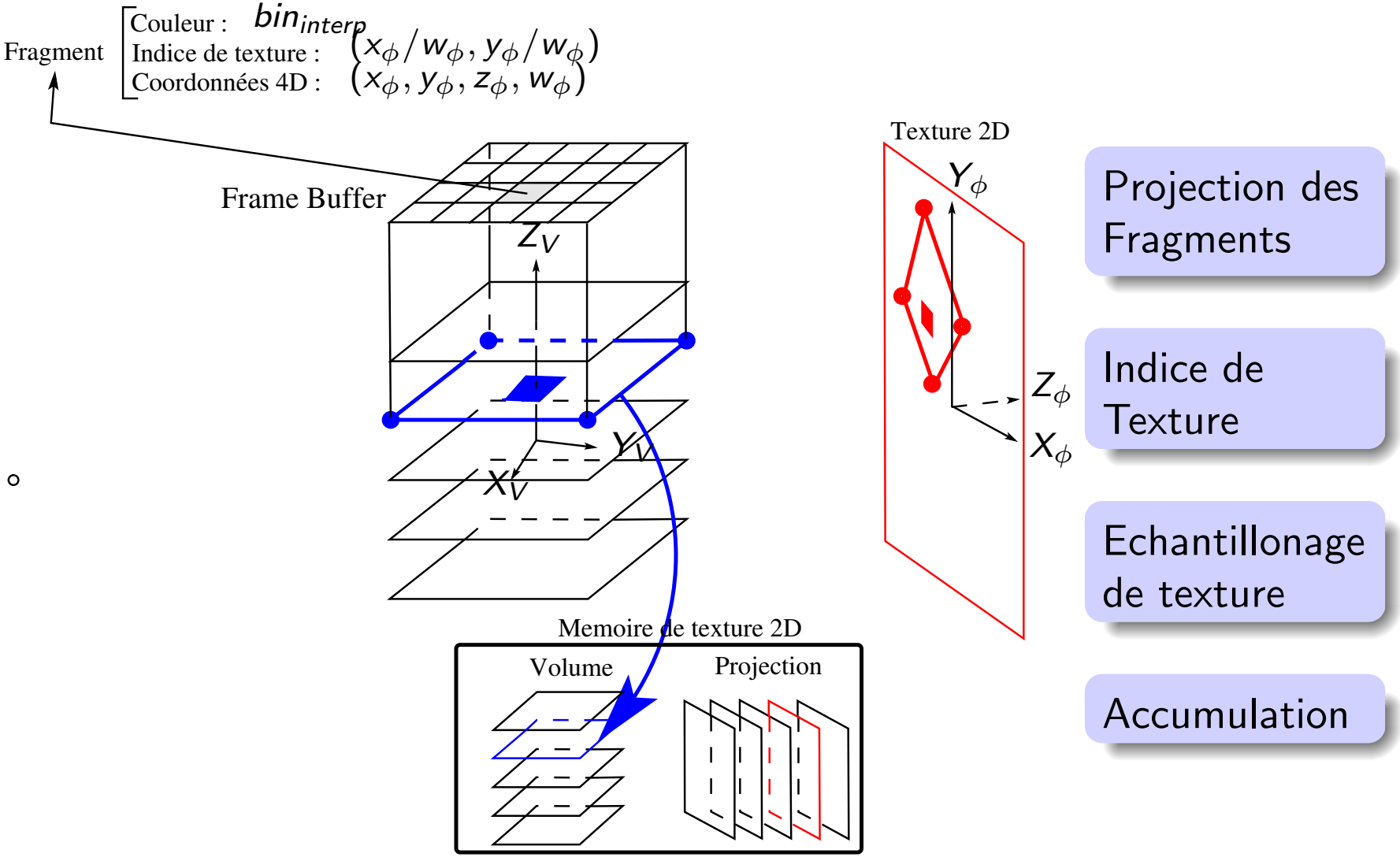
Accumulation

Rétroprojection Cone Beam 3D avec le Pipeline Graphique



- Projection des Vertex
- Projection des Fragments
- Indice de Texture
- Echantillonage de texture
- Accumulation

Rétroprojection Cone Beam 3D avec CUDA



- Projection des Fragments
- Indice de Texture
- Echantillonage de texture
- Accumulation

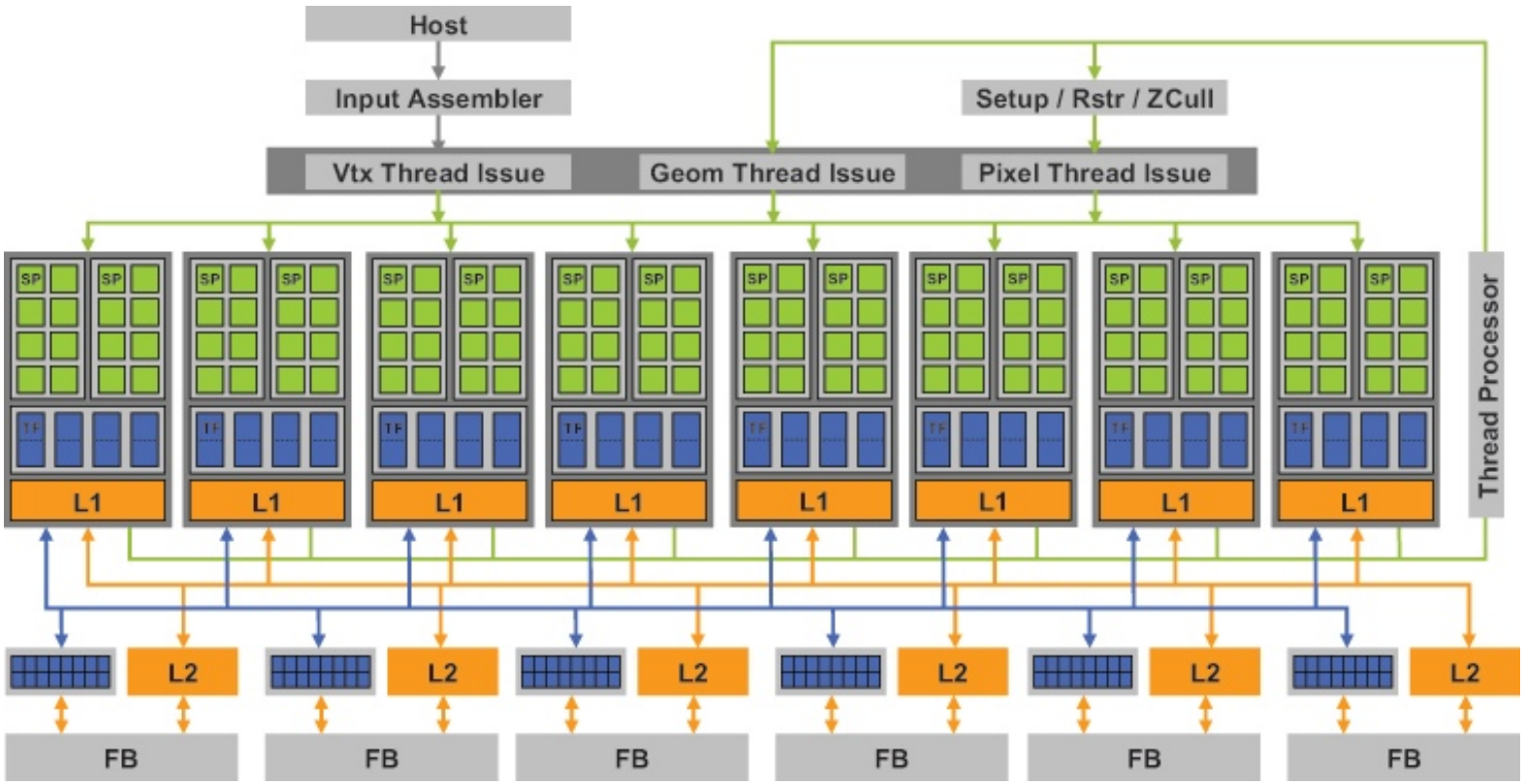
OpenGL (Pipeline Graphique) ou CUDA ?

Intérêt relatif de la programmation OpenGL vs CUDA

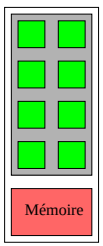
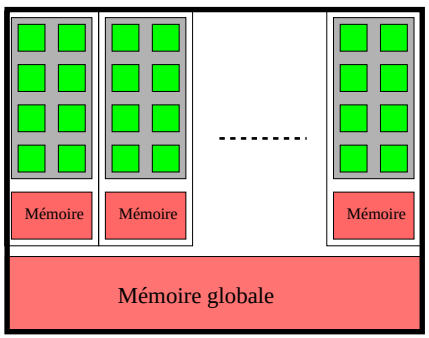
- Plus grande complexité de programmation en OpenGL
- Dans le meilleur des cas, accélération d'un facteur 3

Ref	Feldkamp en **OpenGL/CUDA** (sur 8800 GTX)
[Xu 07]	accélération d'un **facteur 3**
[Knaup 08]	accélération d'un **facteur 1.2**

Après CUDA : plein de threads !



Découpage en threads

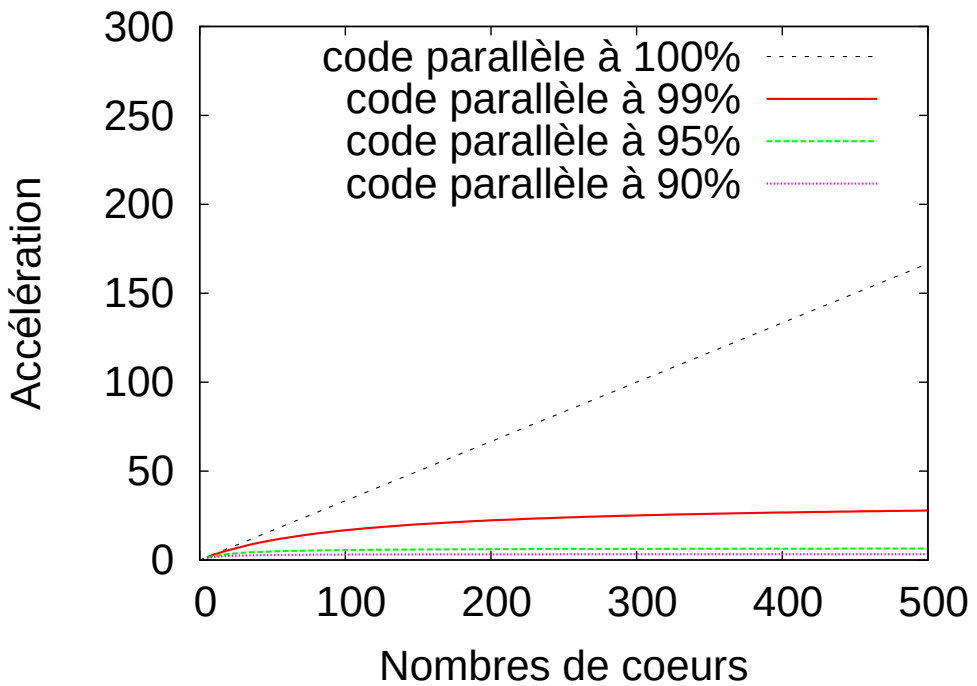
	Matériel	Logiciel	Exécution	
	un Processeur Elémentaire (PE)	un thread	séquentielle	**(a)**
	un processeur multi−threads	un bloc de threads	parallèle (SIMT)	**(b)**
	une carte GPU (device)	une grille de threads (kernel)	parallèle (MIMD) mémoire centralisée	**(c)**

Programmation GPU

❶ Parallélisation de l'algorithme

➲ nourrir en threads (plus ou moins indépendants) le GPU

n coeurs (1 Ghz) vs 1 coeur (3 Ghz)	
taux de parallélisation	accélération GTX 200 (240 coeurs)
100 %	80
99 %	24
95 %	6
90 %	3



Programmation GPU

❶ Parallélisation de l'algorithme

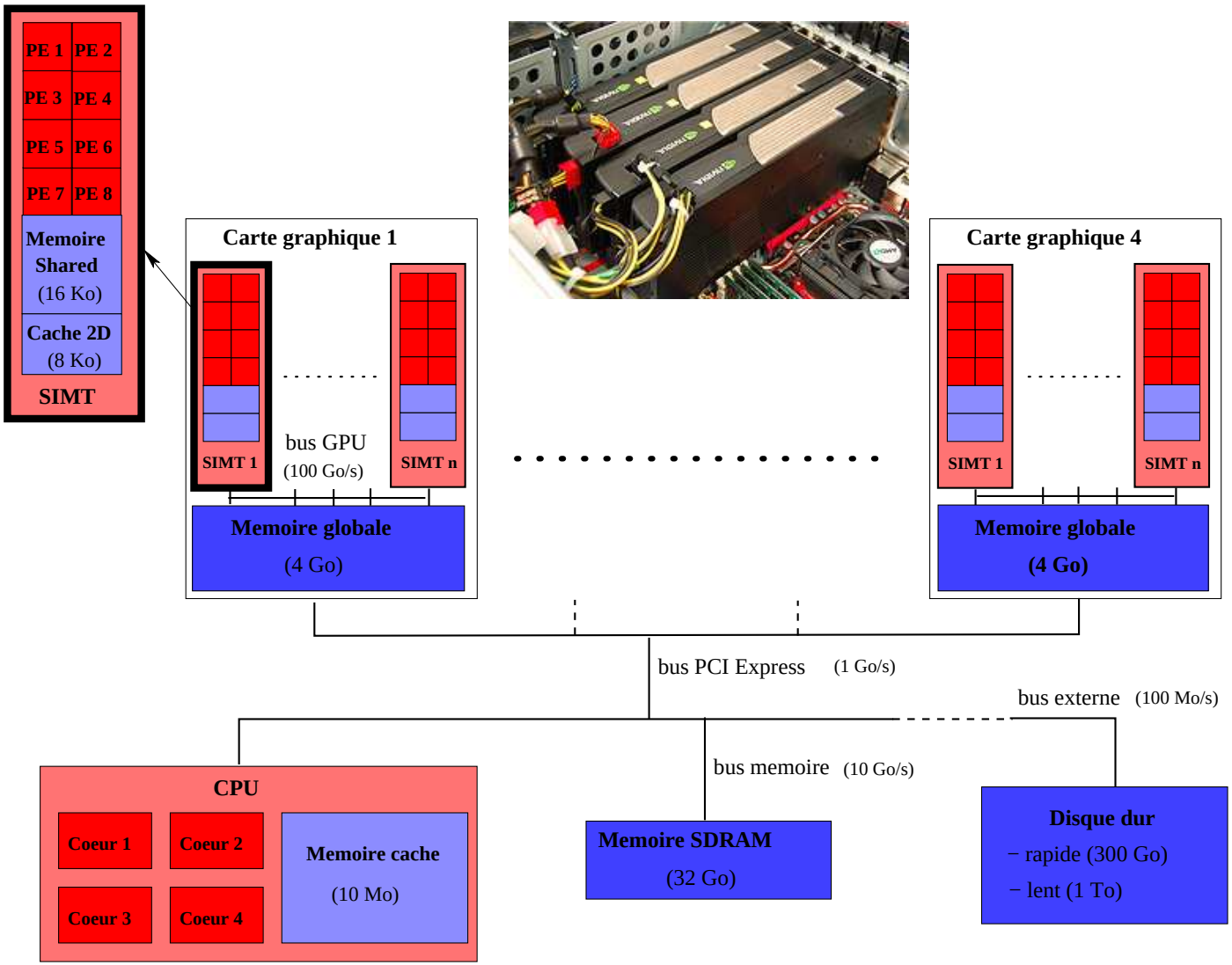
➲ nourrir en threads (plus ou moins indépendants) le GPU

❷ Implémentation GPU

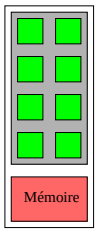
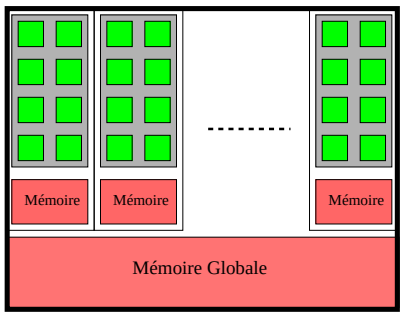
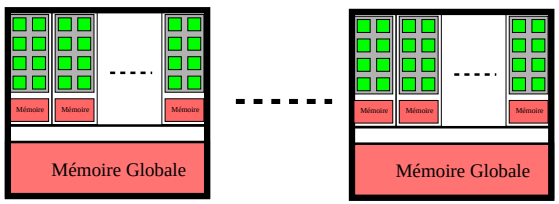
Selon l'intensité arithmétique du code (puissance de calcul exploitée / débit des données), l'execution sera soit *memory bound* soit *computation bound* (ex : calcul X^k [Kirschenmann 08])

➲ optimisation du code portera alors soit sur les **accès mémoire** ou soit sur la **complexité arithmétique**

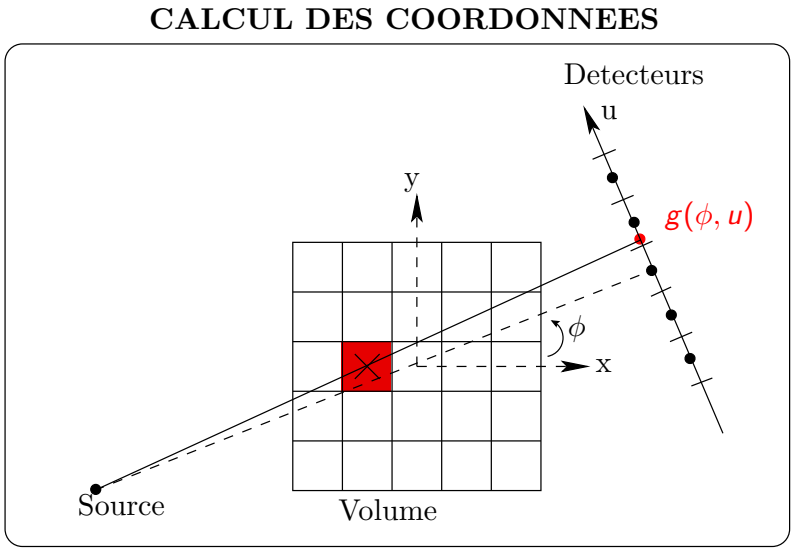
Supercalculateur personnel



Découpage en threads et en **grilles (kernels)**

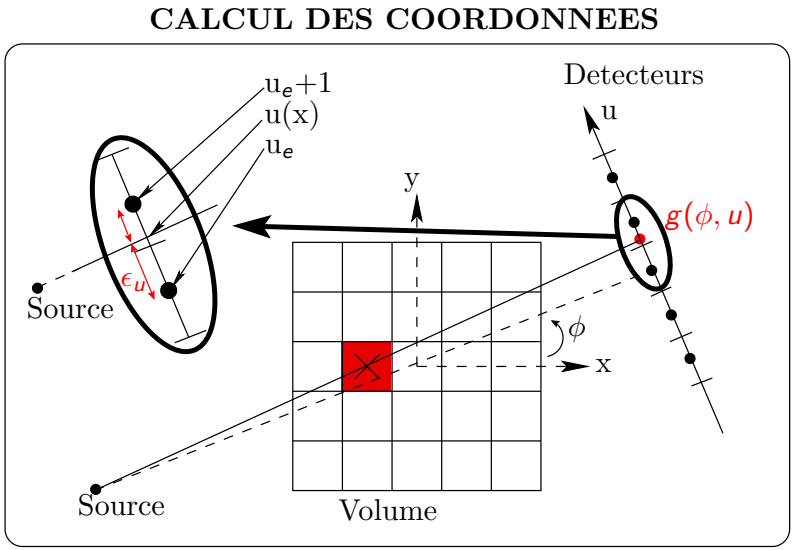
	Matériel	**Logiciel**	**Exécution**	
	un Processeur Elémentaire (PE)	un thread	séquentielle	**(a)**
	un processeur multi−threads	un bloc de threads	parallèle (SIMT)	**(b)**
	une carte GPU (device)	une grille de threads (kernel)	parallèle (MIMD) mémoire centralisée	**(c)**
	PC multi−carte	threads du PC hote via librairie pthread (un thread CPU = un kernel GPU)	parallèle (MIMD) mémoire distribuée	**(d)**

Rétroprojection 2D : algorithme



for $(\texttt{xn}, \texttt{yn})$ in Volume **do**
  for $\texttt{phi} = 0$ to $\texttt{phi}_{\texttt{max}} - 1$ **do**
    // Calcul des coordonnées
    $\texttt{u}(\texttt{phi}, \texttt{xn}, \texttt{yn}) = \ldots$
    // Accumulation
    $f^*(\texttt{xn}, \texttt{yn}) += g(u, \phi)$
  end for
end for

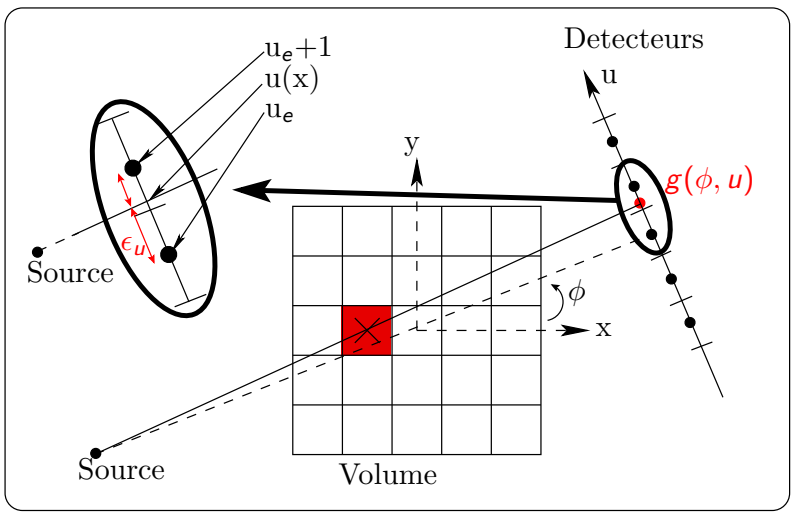
Rétroprojection 2D : interpolation linéaire



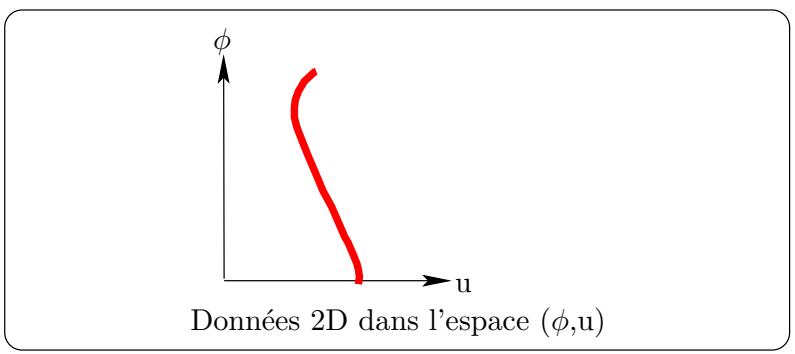
for $(\mathtt{xn}, \mathtt{yn})$ in Volume **do**
  for $\mathtt{phi} = 0$ to $\mathtt{phi}_{\mathtt{max}} - 1$ **do**
    // Calcul des coordonnées
    $\mathtt{u}(\mathtt{phi}, \mathtt{xn}, \mathtt{yn}) = \ldots$
    // Interpolation linéaire
    $g_{interp} = (1 - \epsilon_u) \cdot g(\mathtt{phi}, \mathtt{u_e}) +$
    $\epsilon_u \cdot g(\mathtt{phi}, \mathtt{u_e} + 1)$
    // Accumulation
    $f^*(\mathtt{xn}, \mathtt{yn}) += g_{interp}$
  end for
end for

Rétroprojection 2D : accès aux données dispersés

CALCUL DES COORDONNEES



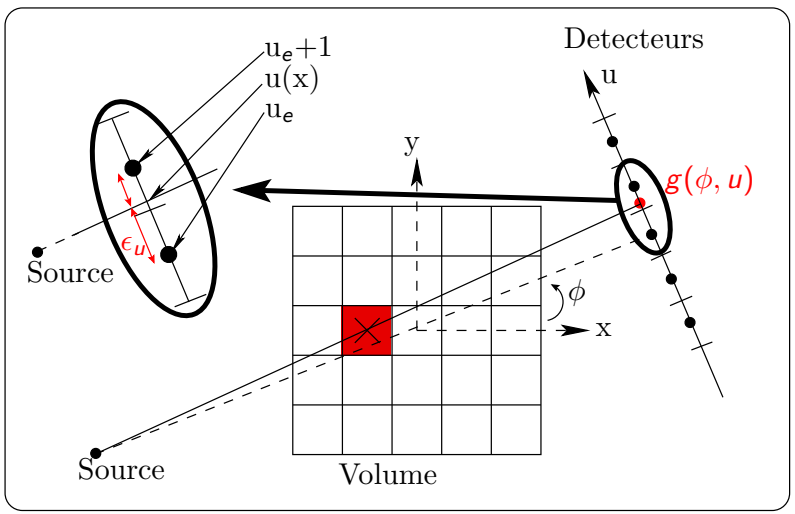
ACCES AUX DONNEES $g(\phi, u)$



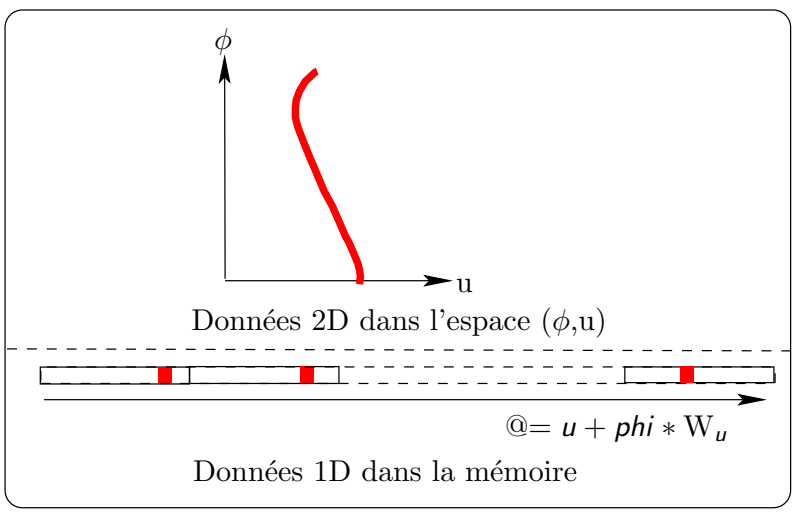
for $(\mathtt{xn}, \mathtt{yn})$ in Volume **do**
  for $\mathtt{phi} = 0$ to $\mathtt{phi}_{\mathtt{max}} - 1$ **do**
    // Calcul des coordonnées
    $\mathtt{u}(\mathtt{phi}, \mathtt{xn}, \mathtt{yn}) = \ldots$
    // Interpolation linéaire
    $g_{interp} = (1 - \epsilon_u) \cdot g(\mathtt{phi}, \mathtt{u_e}) +$
    $\epsilon_u \cdot g(\mathtt{phi}, \mathtt{u_e} + 1)$
    // Accumulation
    $f^*(\mathtt{xn}, \mathtt{yn}) += g_{interp}$
  end for
end for

Rétroprojection 2D : accès aux données dispersés

CALCUL DES COORDONNEES



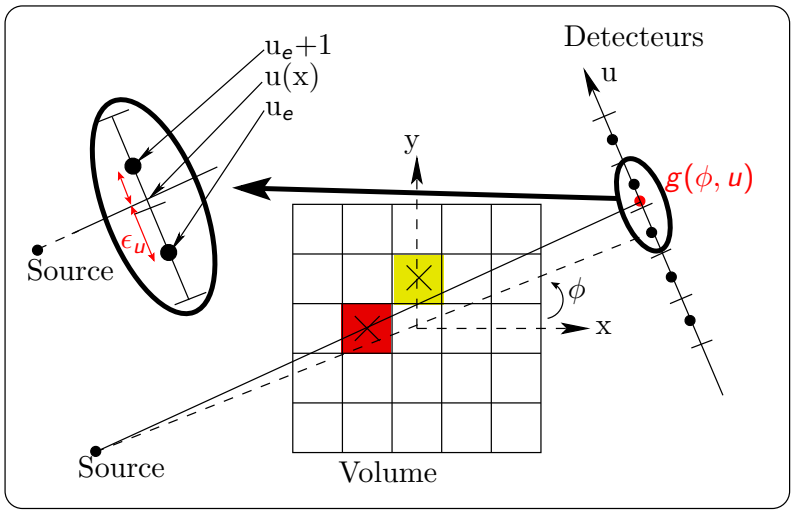
ACCES AUX DONNEES $g(\phi, u)$



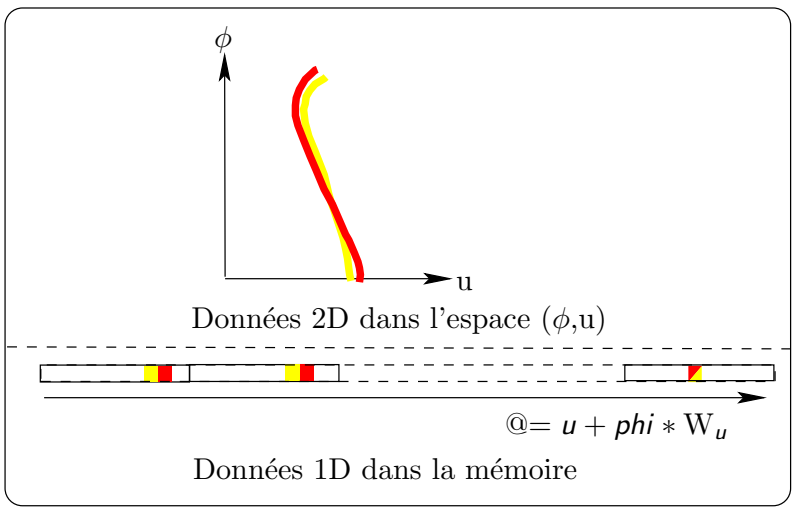
for (xn, yn) in Volume **do**
  for phi $= 0$ to $\texttt{phi}_{\texttt{max}} - 1$ **do**
    // Calcul des coordonnées
    $\texttt{u}(\texttt{phi}, \texttt{xn}, \texttt{yn}) = \ldots$
    // Interpolation linéaire
    $g_{interp} = (1 - \epsilon_u) \cdot g(\texttt{phi}, \texttt{u}_\texttt{e}) +$
    $\epsilon_u \cdot g(\texttt{phi}, \texttt{u}_\texttt{e} + 1)$
    // Accumulation
    $f^*(\texttt{xn}, \texttt{yn}) += g_{interp}$
  end for
end for

Rétroprojection 2D : accès aux données dispersés

CALCUL DES COORDONNEES



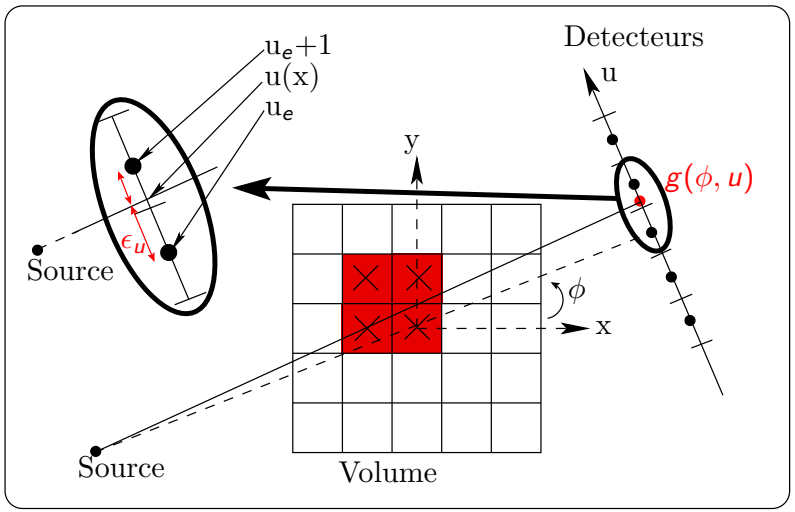
ACCES AUX DONNEES $g(\phi, u)$



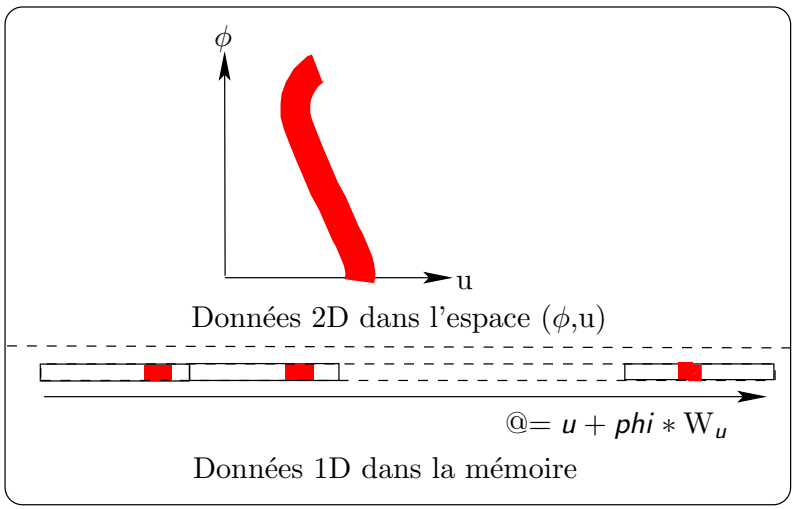
for (xn, yn) in Volume **do**
  for phi $= 0$ to $\texttt{phi}_{\texttt{max}} - 1$ **do**
    // Calcul des coordonnées
    $\texttt{u}(\texttt{phi}, \texttt{xn}, \texttt{yn}) = \ldots$
    // Interpolation linéaire
    $g_{interp} = (1 - \epsilon_u) \cdot g(\texttt{phi}, \texttt{u}_\texttt{e}) +$
    $\epsilon_u \cdot g(\texttt{phi}, \texttt{u}_\texttt{e} + 1)$
    // Accumulation
    $f^*(\texttt{xn}, \texttt{yn}) += g_{interp}$
  end for
end for

Rétroprojection 2D par bloc : accès aux données localisés

CALCUL DES COORDONNEES

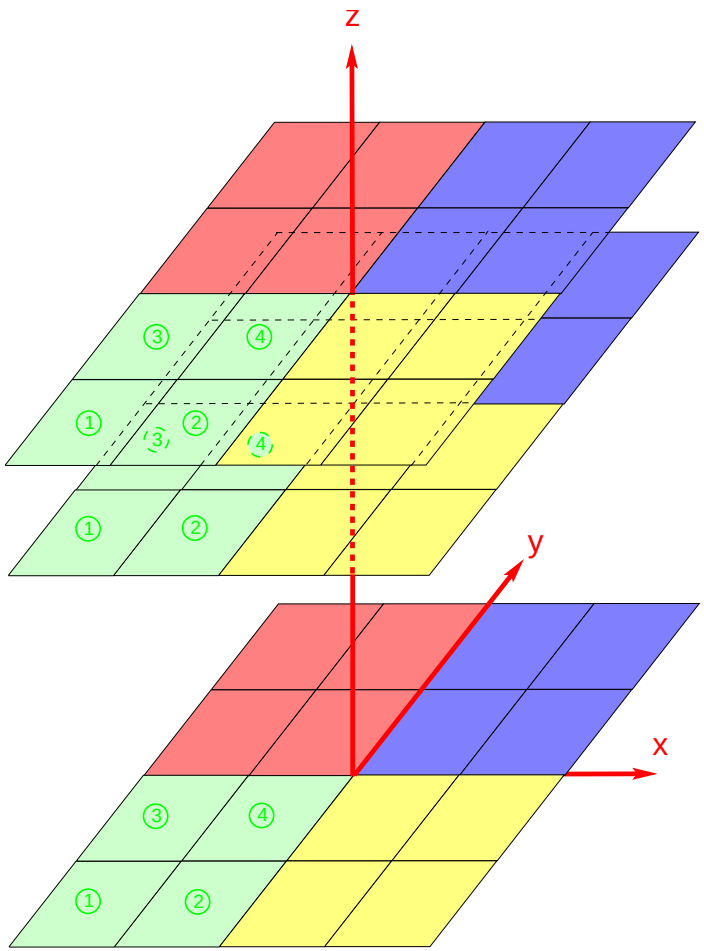


ACCES AUX DONNEES $g(\phi, u)$



for (Bx, By) in Volume **do**
  for phi $= 0$ to $\text{phi}_{\text{max}} - 1$ **do**
    for (xn, yn) in Bloc **do**
      // *Calcul des coordonnées*
      u(phi, xn, yn) = ...
      // *Interpolation linéaire*
      $g_{interp} = (1 - \epsilon_u) \cdot$
      $g(\text{phi}, \text{u}_\text{e}) + \epsilon_u \cdot g(\text{phi}, \text{u}_\text{e} + 1)$
      // *Accumulation*
      $f^*(\text{xn}, \text{yn}) += g_{interp}$
    end for
  end for
end for

Découpage en threads de la rétroprojection 3D



(a) Calcul séquentiel sur un PE

- Boucle sur z
- Boucle sur ϕ

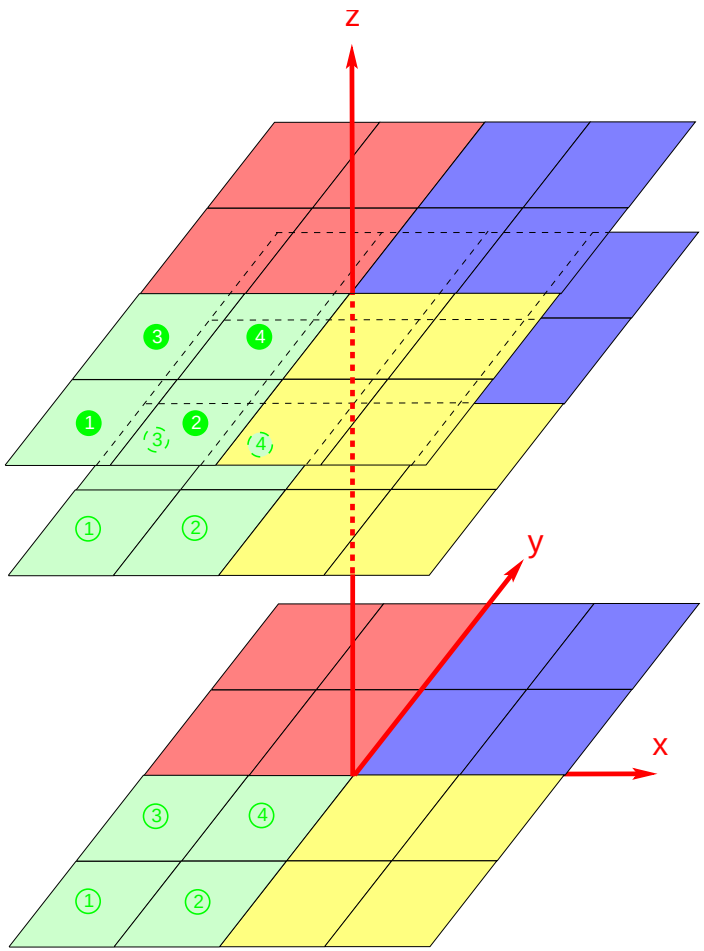
(b) Calcul parallèle sur un proc. SIMT

- Boucle sur (x,y)

(c) Calcul parallèle sur une carte

- Boucle sur les blocs (Bx,By,Bz)

Découpage en threads de la rétroprojection 3D



(a) Calcul séquentiel sur un PE

- Boucle sur z
- Boucle sur ϕ

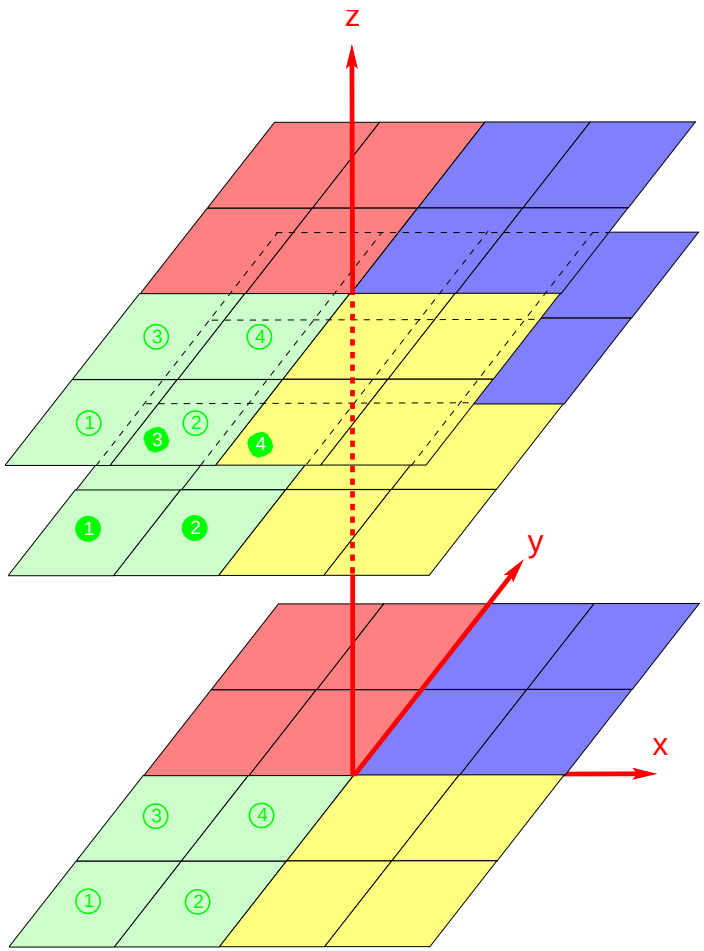
(b) Calcul parallèle sur un proc. SIMT

- Boucle sur (x,y)

(c) Calcul parallèle sur une carte

- Boucle sur les blocs (Bx,By,Bz)

Découpage en threads de la rétroprojection 3D



(a) Calcul séquentiel sur un PE

- Boucle sur z
- Boucle sur ϕ

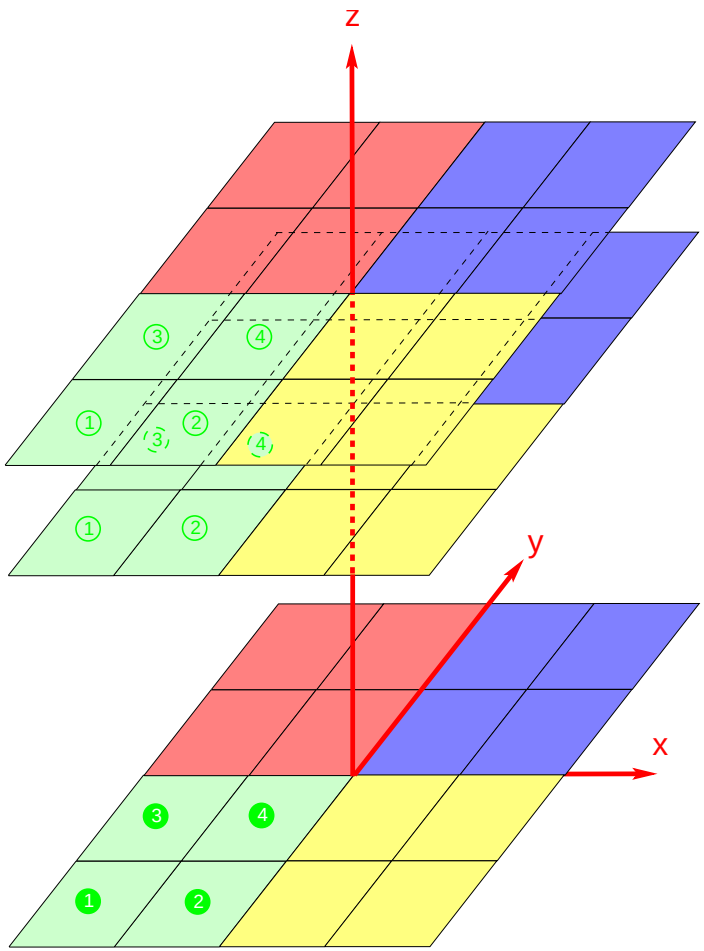
(b) Calcul parallèle sur un proc. SIMT

- Boucle sur (x,y)

(c) Calcul parallèle sur une carte

- Boucle sur les blocs (Bx,By,Bz)

Découpage en threads de la rétroprojection 3D



(a) Calcul séquentiel sur un PE

- Boucle sur z
- Boucle sur ϕ

(b) Calcul parallèle sur un proc. SIMT

- Boucle sur (x,y)

(c) Calcul parallèle sur une carte

- Boucle sur les blocs (Bx,By,Bz)

Pseudo-code parallèle de la rétroprojection 3D

```
for   (Bx.id,By.id,Bz.id)=(0,0) to (N_Bx − 1, N_By − 1, N_Bz − 1)                    c
    for   (Tx.id,Ty.id)=(0,0) to (N_Tx-1,N_Ty-1)                                    b
        xn=Bx.id ·N_Tx+Tx.id                                                        a
        yn=By.id ·N_Ty+Ty.id

        for   zn = zn_first to zn_stop
              for   phi = phi_start to phi_stop

                    un(xn, yn, phi) = ...
                    A_{x,y,φ} = ...
                    vn(xn, yn, zn, phi) = A_{x,y,φ} · zn
                    @tex_{un,phi} = un + phi · N_un
                    B(xn, yn, zn)+ = tex2D(@tex_{un,phi}, vn)
                    zn + +
              end

              // Accumulation du Bloc B_{Bx.id,By.id}
              // de la mémoire shared dans la mémoire globale
        end
    end
end
```

Calcul incrémental de vn

for (Bx.id,By.id,Bz.id)=(0,0) to $(N_{Bx}-1, N_{By}-1, N_{Bz}-1)$ **c**

  for (Tx.id,Ty.id)=(0,0) to (N_{Tx}-1,N_{Ty}-1) **b**

    xn=Bx.id $\cdot N_{Tx}$+Tx.id **a**

    yn=By.id $\cdot N_{Ty}$+Ty.id

    for phi $=$ phi_{start} to phi_{stop}

      $\text{un}(\text{xn}, \text{yn}, \text{phi}) = \ldots$

      $A_{x,y,\phi} = \ldots$

      $\text{vn}(\text{xn}, \text{yn}, \text{zn}, \text{phi}) = A_{x,y,\phi} \cdot \text{zn}_0$

      $@tex_{\text{un},\text{phi}} = \text{un} + \text{phi} \cdot N_{\text{un}}$

      for $\text{zn} = \text{zn}_{first}$ to zn_{stop}

        $B(\text{xn}, \text{yn}, \text{zn}) += tex2D(@tex_{\text{un},\text{phi}}, \text{vn})$

        $\text{vn} += A_{x,y,\phi}$

        $\text{zn}++$

      end

      // Accumulation du Bloc $B_{Bx.id,By.id}$

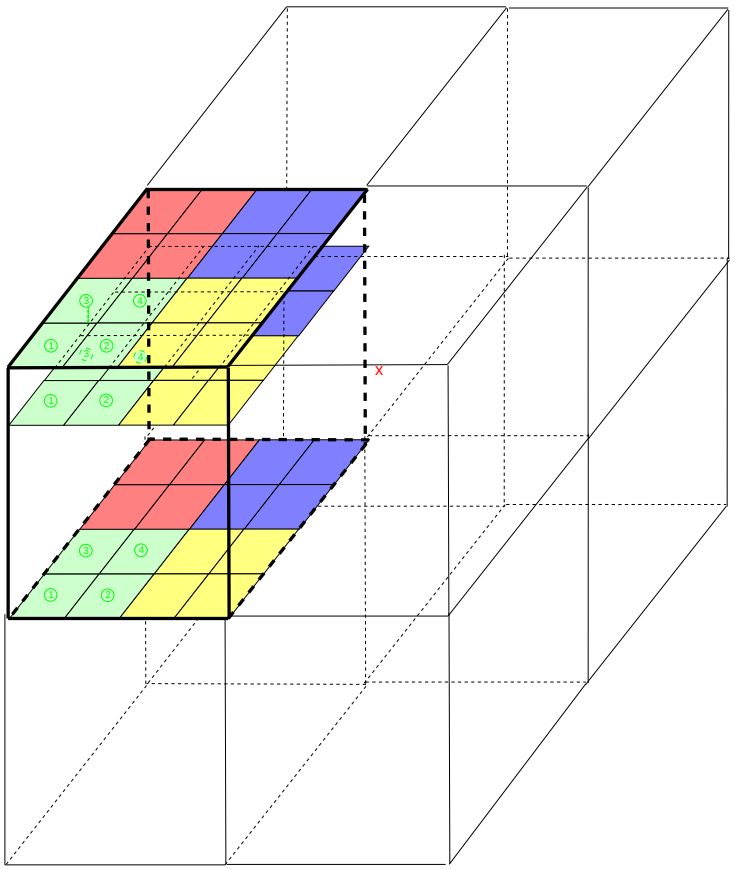
      // de la mémoire shared dans la mémoire globale

    end

  end

end

Découpage en threads et en **grilles** de la rétroprojection 3D



(a) Calcul séquentiel sur un PE

- Boucle sur z
- Boucle sur ϕ

(b) Calcul parallèle sur un proc. SIMT

- Boucle sur (x,y)

(c) Calcul parallèle sur une carte

- Boucle sur les blocs (Bx,By,Bz)

(d) Calcul parallèle sur PC multi-cartes

- Boucle sur les grilles (Gx,Gy)
- Boucle sur la grille Gϕ

Pseudo-code parallèle de la rétroprojection 3D

```
for (Gx.id,Gy.id)=(0,0) to (N_Gx − 1, N_Gy − 1)                                   d
    for Gφ.id=0 to N_Gφ − 1
        //Copie des Plans P_phi_start ..P_phi_stop de la mémoire CPU dans la mémoire globale du GPU
        for (Bx.id,By.id,Bz.id)=(0,0) to (N_Bx − 1, N_By − 1, N_Bz − 1)           c
            for (Tx.id,Ty.id)=(0,0) to (N_Tx-1,N_Ty-1)                            b
                xn=Bx.id ·N_Tx+Tx.id                                              a
                yn=By.id ·N_Ty+Ty.id

                for phi = phi_start to phi_stop

                    un(xn, yn, phi) = ...
                    A_{x,y,φ} = ...
                    vn(xn, yn, zn, phi) = A_{x,y,φ} · zn_0
                    @tex_{un,phi} = un + phi · N_un

                    for zn = zn_first to zn_stop
                        B(xn, yn, zn)+ = tex2D(@tex_{un,phi}, vn)
                        vn+ = A_{x,y,φ}
                        zn + +
                    end

                    // Accumulation du Bloc B_{Bx.id,By.id,Bz.id}
                    // de la mémoire shared dans la mémoire globale
                end
            end
        end
    end
    //Accumulation de la grille G_{(Gx.id,Gy.id)} de la memoire globale dans la mémoire du PC hote
end
```

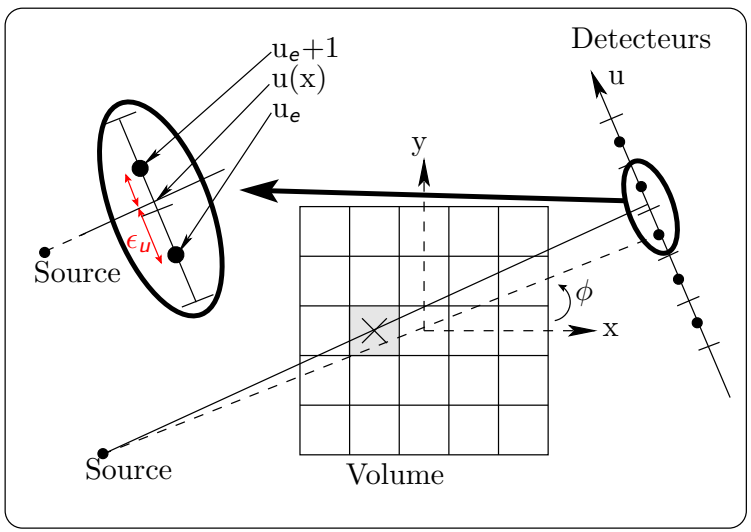
Compromis pour la taille des boucles

Boucles	*Commentaires*	
(Bx,By)/(Tx,Ty)	(Tx,Ty) $\nearrow$	parallélisation sur un proc. SIMT
	(Tx,Ty) $\nearrow$	acces local via le cache 2D (8Ko)
	(Tx,Ty) $\searrow$	taille du cache 2D (8Ko) $->$ dimension u de la tuile
	(Tx,Ty) $\searrow$	stockage du bloc dans la mémoire shared (16Ko)
(Gx,Gy)/(Bx,By)	(Bx,By) $\searrow$	stockage de la grille dans la mémoire globale (4Go)
	(Bx,By) $\nearrow$	minimisation des transferts mémoire CPU$<-> $GPU
Gϕ/ϕ	ϕ $\nearrow$	augmentation de la taille du thread
	ϕ $\nearrow$	accumulation dans la mémoire shared (pas dans la mémoire CPU)
	ϕ $\searrow$	stockage des plans dans la mémoire globale (4Go)
	ϕ $\searrow$	minimisation des transferts mémoire CPU$<-> $GPU
Bz/zn	zn $\searrow$	stockage du bloc dans la mémoire shared (16Ko)
	zn $\nearrow$	minimisation transfert mem. shared$<-> $mem. globale
	zn $\nearrow$	bénéfice du calcul incrémental
Tx,Ty	Tx$\cdot$Ty$=$ 256 threads	
Bx,By	Bx$\cdot$By $=$ multiple de 32	

Ordres de grandeur

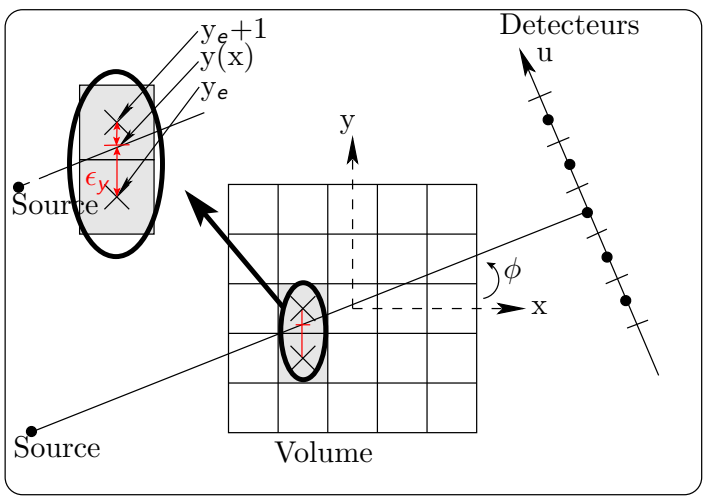
Volume	512^3	1024^3	2048^3
zn		16	
Tx,Ty		16·16	
Bx,By	32·32	32·32	32·32
Bz	16	32	64
Gx,Gy	1·1	2·2	4·4
un bloc	16 Ko	16 Ko	16 Ko
une grille	256 Mo	512 Mo	1 Go
ϕ	512	512	128
Gϕ	1	2	16
Tuile	16·16=1 Ko		
Plan	$512 \cdot 256$ = 512 Ko	$1024 \cdot 512$ = 2 Mo	$2048 \cdot 1024$ = 8 Mo
Memoire globale pour projection	256 Mo	1 Go	1Go

Projecteur 2D "voxel-driven"



for $(\texttt{xn}, \texttt{yn})$ in Volume **do**
  for $\texttt{phi} = 0$ to $\texttt{phi}_{\texttt{max}} - 1$ **do**
    // Calcul des coordonnées
    $\texttt{u}(\texttt{phi}, \texttt{xn}, \texttt{yn}) = \ldots$
    // Calcul des coefficents d'interpolation
    $C_0 = 1 - \epsilon_u$
    $C_1 = \epsilon_u$
    // Dépôt de la dose
    $g(\texttt{u}_\texttt{e}, \texttt{phi}) += C_0 \cdot V(\texttt{xn}, \texttt{yn})$
    $g(\texttt{u}_\texttt{e} + 1, \texttt{phi}) += C_1 \cdot V(\texttt{xn}, \texttt{yn})$
  end for
end for

Projecteur 2D "ray-driven" (Algorithme 2D de [Joseph 82])



for $(\texttt{un}, \texttt{phi})$ in Projection **do**
  for $\texttt{xn} = 0$ to $\texttt{xn}_{\texttt{max}} - 1$ **do**
    // Calcul des coordonnées
    $\texttt{yn}(\texttt{xn}, \texttt{un}, \texttt{phi}) = \ldots$
    // Interpolation linéaire
    $f_{interp} = (1 - \epsilon_y) \cdot f(\texttt{xn}, \texttt{yn}_\texttt{e}) +$
    $\epsilon_y \cdot f(\texttt{xn}, \texttt{yn}_\texttt{e} + 1)$
    // Accumulation
    $g^*(\texttt{un}, \texttt{phi}) += f_{interp}$
  end for
end for

Débit de calcul de la paire projection/rétroprojection

Reconstruction sur une GTX 295 (240 SPs @1.2 Ghz)

- volume de 264^3 voxels (champs de vue cylindrique)
- 64 projections de 256^2 pixels

	Temps	Giga opérations par seconde (Gop/s)
Rétroprojecteur VIB	63 ms	9,9 Gop/s
Projecteur de Joseph	63 ms	15,9 Gop/s

Conclusion : Atteindre l'équilibre entre débit de calcul et débit mémoire

❶ Exploiter au maximum la puissance de calcul

- Parallélisation de l'algorithme
- Réduction de la complexité arithmétique
- Utilisation de l'interpolateur *hardware* de texture

Conclusion : Atteindre l'équilibre entre débit de calcul et débit mémoire

❶ Exploiter au maximum la puissance de calcul

- Parallélisation de l'algorithme
  ➲ **Découpage en threads et en kernels**
- Réduction de la complexité arithmétique
- Utilisation de l'interpolateur *hardware* de texture

Conclusion : Atteindre l'équilibre entre débit de calcul et débit mémoire

❶ Exploiter au maximum la puissance de calcul

- Parallélisation de l'algorithme
  ➲ **Découpage en threads et en kernels**
- Réduction de la complexité arithmétique
  ➲ **Calcul incrémental des coordonnées**
- Utilisation de l'interpolateur *hardware* de texture

Conclusion : Atteindre l'équilibre entre débit de calcul et débit mémoire

❶ Exploiter au maximum la puissance de calcul

- Parallélisation de l'algorithme
  ➲ **Découpage en threads et en kernels**
- Réduction de la complexité arithmétique
  ➲ **Calcul incrémental des coordonnées**
- Utilisation de l'interpolateur *hardware* de texture
  ➲ **Interpolation bi-linéaire**

❷ Favoriser la localité spatio-temporelle des données

- Lecture des données via le cache 2D de texture
  - réutilisation au maximum des données
  - accès à des données 2D
- Ecriture dans la mémoire shared

Conclusion : Atteindre l'équilibre entre débit de calcul et débit mémoire

❶ Exploiter au maximum la puissance de calcul

- Parallélisation de l'algorithme
  ➲ **Découpage en threads et en kernels**
- Réduction de la complexité arithmétique
  ➲ **Calcul incrémental des coordonnées**
- Utilisation de l'interpolateur *hardware* de texture
  ➲ **Interpolation bi-linéaire**

❷ Favoriser la localité spatio-temporelle des données

- Lecture des données via le cache 2D de texture
  - réutilisation au maximum des données
  - accès à des données 2D
- Ecriture dans la mémoire shared
  ➲ **Harmoniser l'ordre des boucles avec la structure des données**

Travaux en cours

❶ Algorithme itératif avec paire projection/rétroprojection GPU

Interface Matlab

❷ Cohérence paire projection/rétroprojection

Etude de la convergence de l'algorithme itératif pour plusieurs projecteurs couplés avec un rétroprojecteur voxel driven

❸ Projecteur ray driven avec échantillonage régulier

Parallélisation sur GPU à l'aide des textures 3D

Perspectives

Parallélisation sur serveur 4 Tesla (S1070)

- Décomposition de l'Espace Image (DEI) : volume distribué et projections centralisées
- Décomposition de l'Espace de Projection (DEP) : volume centralisé et projections distribuées

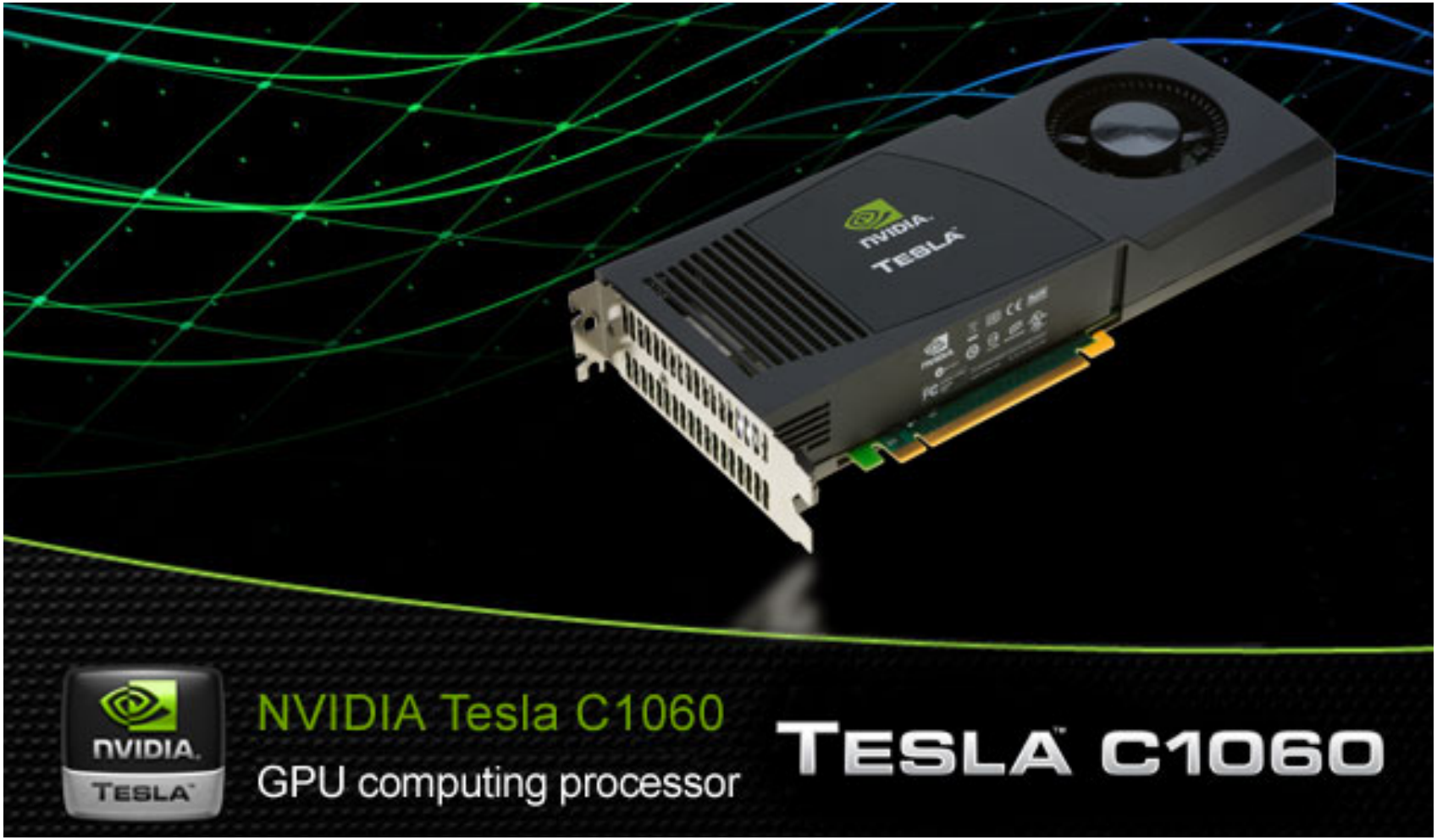
Données rélles du CEA

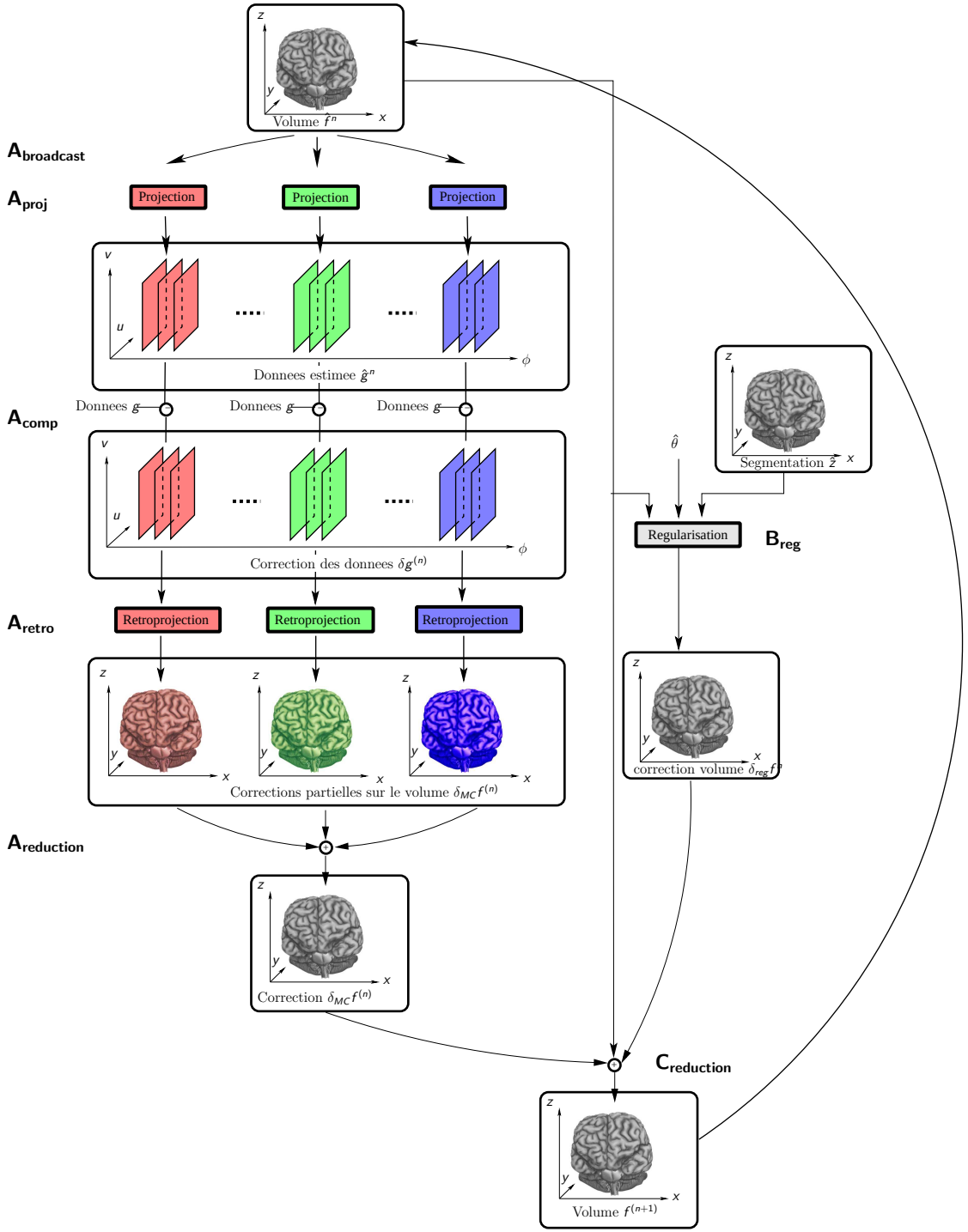
Objectif : reconstruction de volumes 2048^3

Etape de segmentation sur GPU

Utile ou pas utile ?

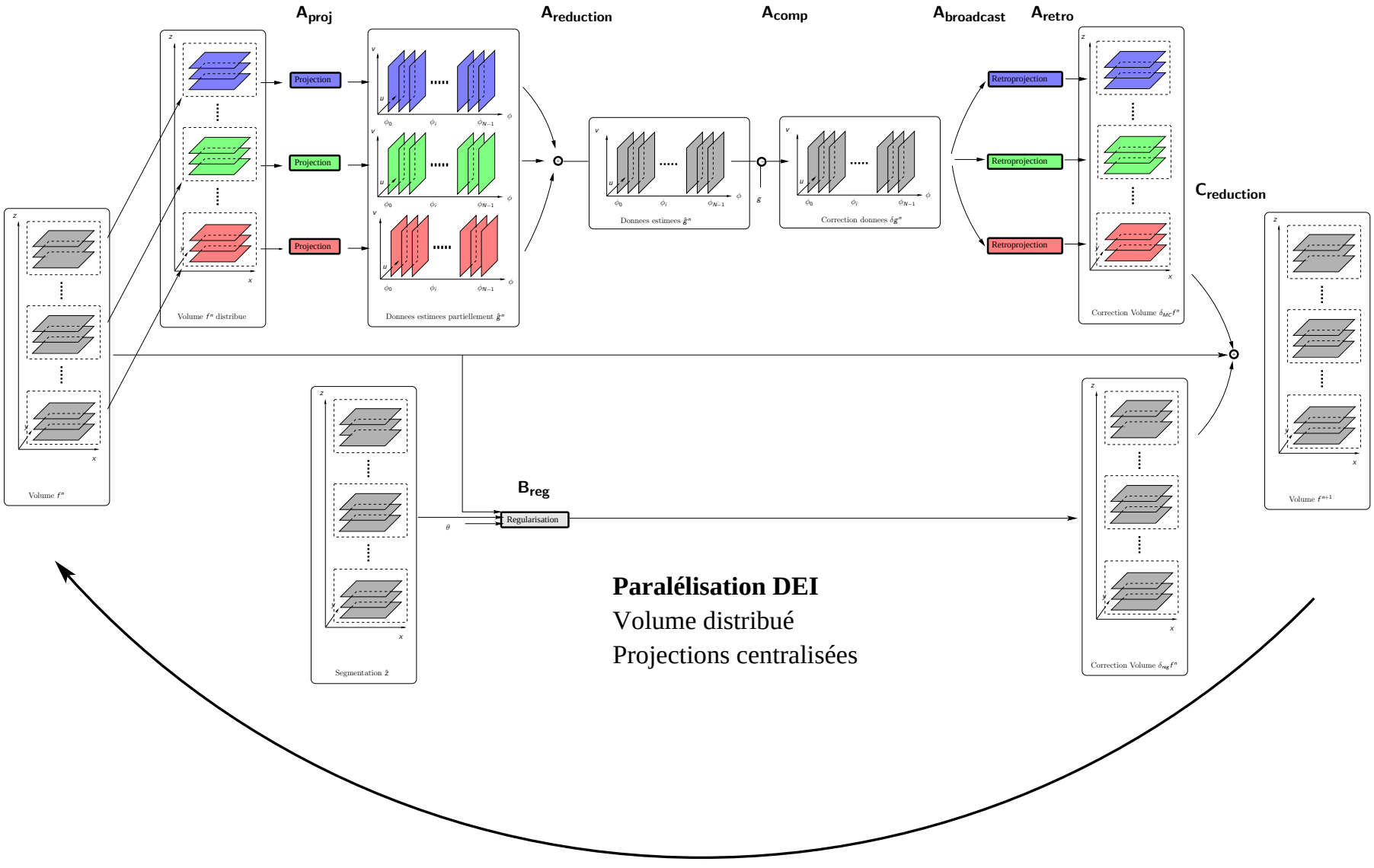
Merci de votre attention !





Parallélisation DEP

Volume centralisé
Projections centralisées


A_{proj}
$A_{reduction}$
A_{comp}
$A_{broadcast}$
A_{retro}
$C_{reduction}$
B_{reg}
Projection
Projection
Projection
Retroprojection
Retroprojection
Retroprojection
Regularisation
Volume f^n distribue
Donnees estimees partiellement $\hat{g}^n$
Donnees estimees $\hat{g}^n$
Correction donnees δg^n
Correction Volume $\delta_{MC} f^n$
Volume f^n
Volume f^{n+1}
Segmentation $\hat{z}$
Correction Volume $\delta_{reg} f^n$
Paralélisation DEI
Volume distribué
Projections centralisées

Références

N. Gac, S.Mancini, M.Desvignes & D.Houzet.

High Speed 3D Tomography on CPU, GPU and FPGA.
EURASIP Journal on Embedded systems, vol. Special issue : Design and Architectures for Signal Image Processing, 2008.

Peter M. Joseph.

An Improved Algorithm for Reprojecting Rays through Pixel Images.
vol. 1, no. 3, pages 192–196, Nov. 1982.

Wilfried Kirschenmann.

Parallélisation d'un solveur de neutronique sur GPU.
In 2ième Journée Thème Émergeant GPGPU (GDR ASR), December 2008.

M. Knaup & M. Kachelrieß.

GPU-Based Parallel-Beam and Cone-Beam GPU Forward- and Backprojection using CUDA.
In MIC, 2008.

Fang Xu & Klaus Mueller.

Real-time 3D computed tomographic reconstruction using commodity graphics hardware.
Physics in Medicine and Biology, vol. 52, no. 12, pages 3405–3419, 2007.