UMONS
Université de Mons

Formations CISM/CÉCI

Pour une meilleure exploitation des architectures Multi-CPU/Multi-GPU

Application au traitement d'objets multimédia

Sidi Ahmed Mahmoudi

14 Mars 2013

POLYTECH MONS

OLIMP

cost
EUROPEAN COOPERATION
IN SCIENCE AND TECHNOLOGY

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

VI. Résultats Expérimentaux: cas d'utilisations du modèle

Conclusion

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

VI. Résultats Expérimentaux: cas d'utilisations du modèle

Conclusion

Introduction

Loi de Moore : 1968

Le nombre de transistors composant un circuit électronique à prix constant double tous les 2 ans

✓ Loi bien vérifiée : puissance des CPU doublée tous les 18 mois

✗ Fréquence des CPU plafonnée à 4 GHZ : la loi n'est plus vérifiée

Introduction

- Multiplication d'unités de calcul : **Multi-CPU, Multi-core, GPU**
- GPU : initialement utilisés pour le rendu 2D/3D et jeux vidéos

CPU Intel Core i7 3770K (2012)

4 cores/8 threads

320 Euros

GPU GeForce GTX 580 (2011)

512 CUDA cores

350 Euros

Naissance du GPGPU : General Purpose Graphic Processing Unit

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

VI. Résultats Expérimentaux: cas d'utilisations du modèle

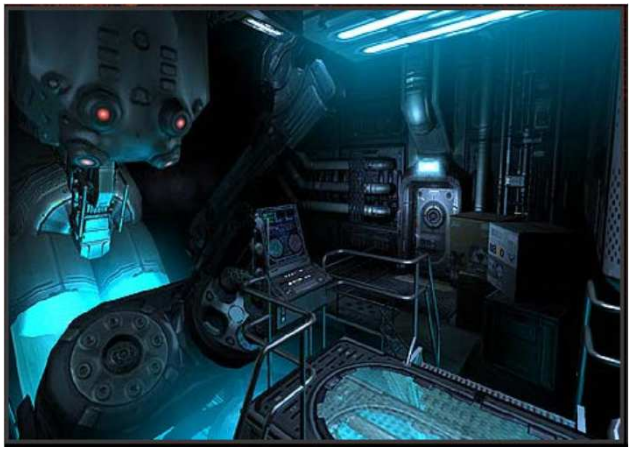
Conclusion

Présentation des GPU

- Unités de calcul graphiques

NVIDIA

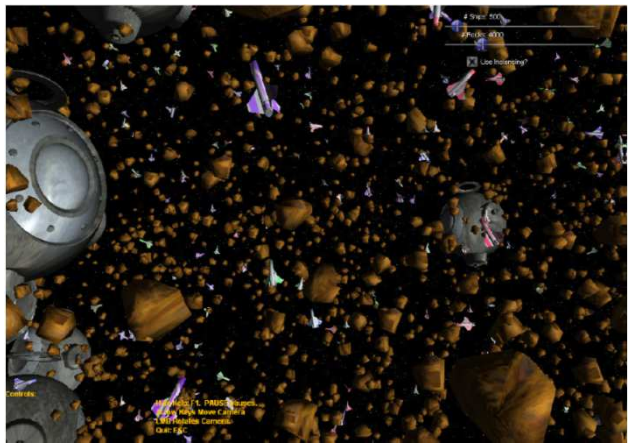
ATI



- Placés dans des cartes graphiques

GeForce GTX 580

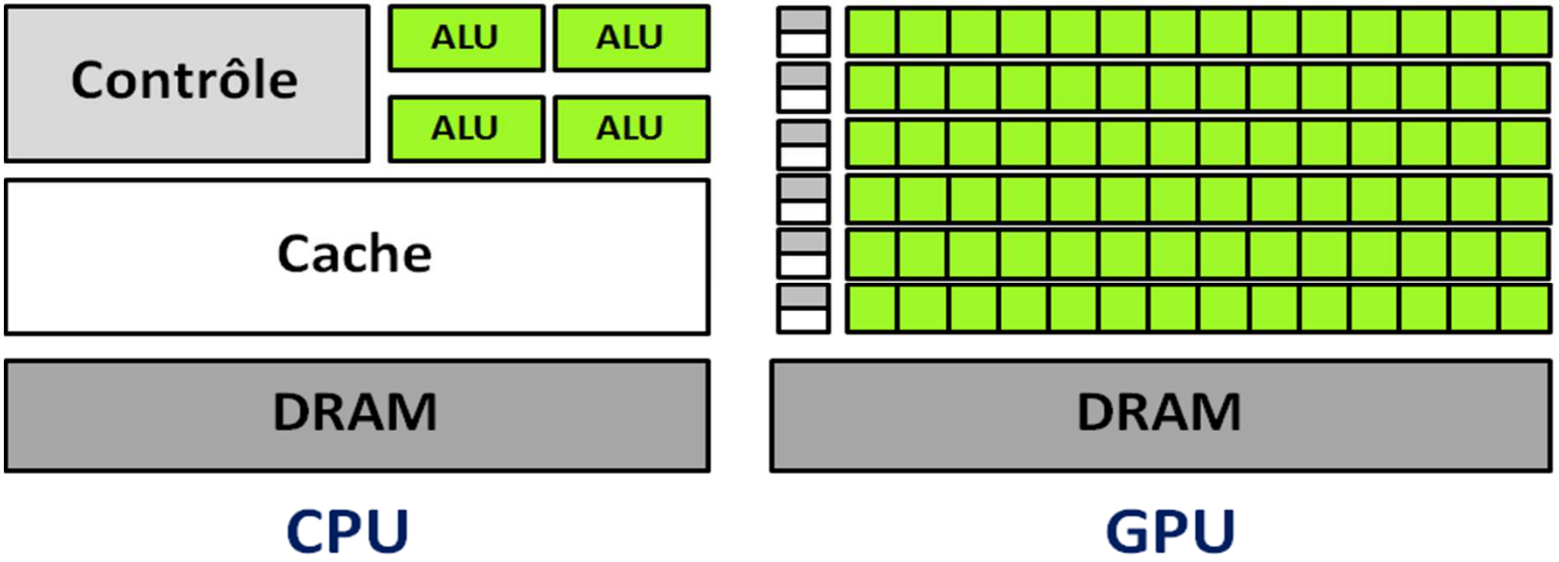
Radeon HD4870



- Initialement utilisés pour les applications 3D et jeux vidéos

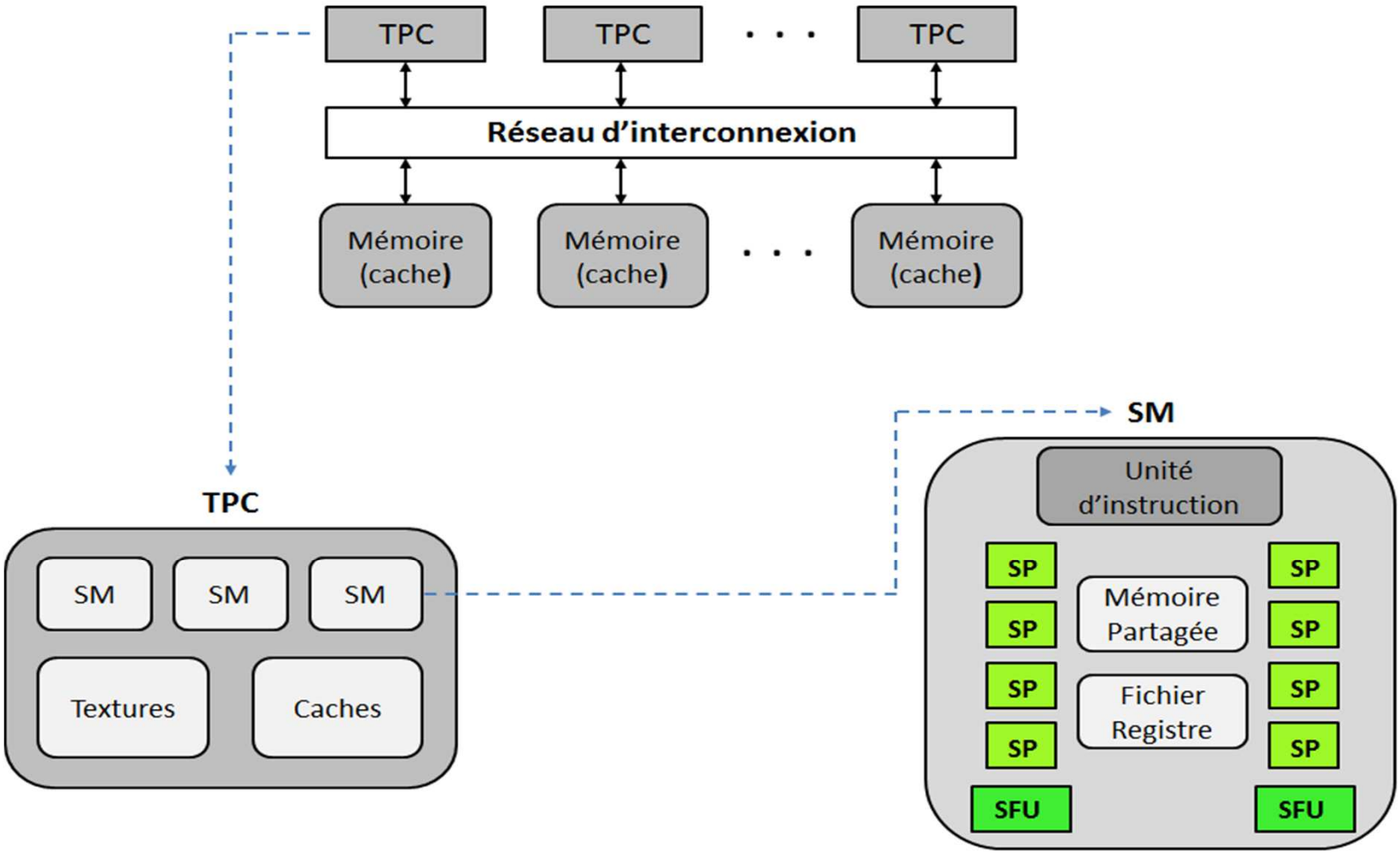
Présentation des GPU

- GPU spécialisés pour les calculs massivement parallèles
- Plus de transistors dédiés aux unités de traitement
- Peu d'unités de contrôle et de cache

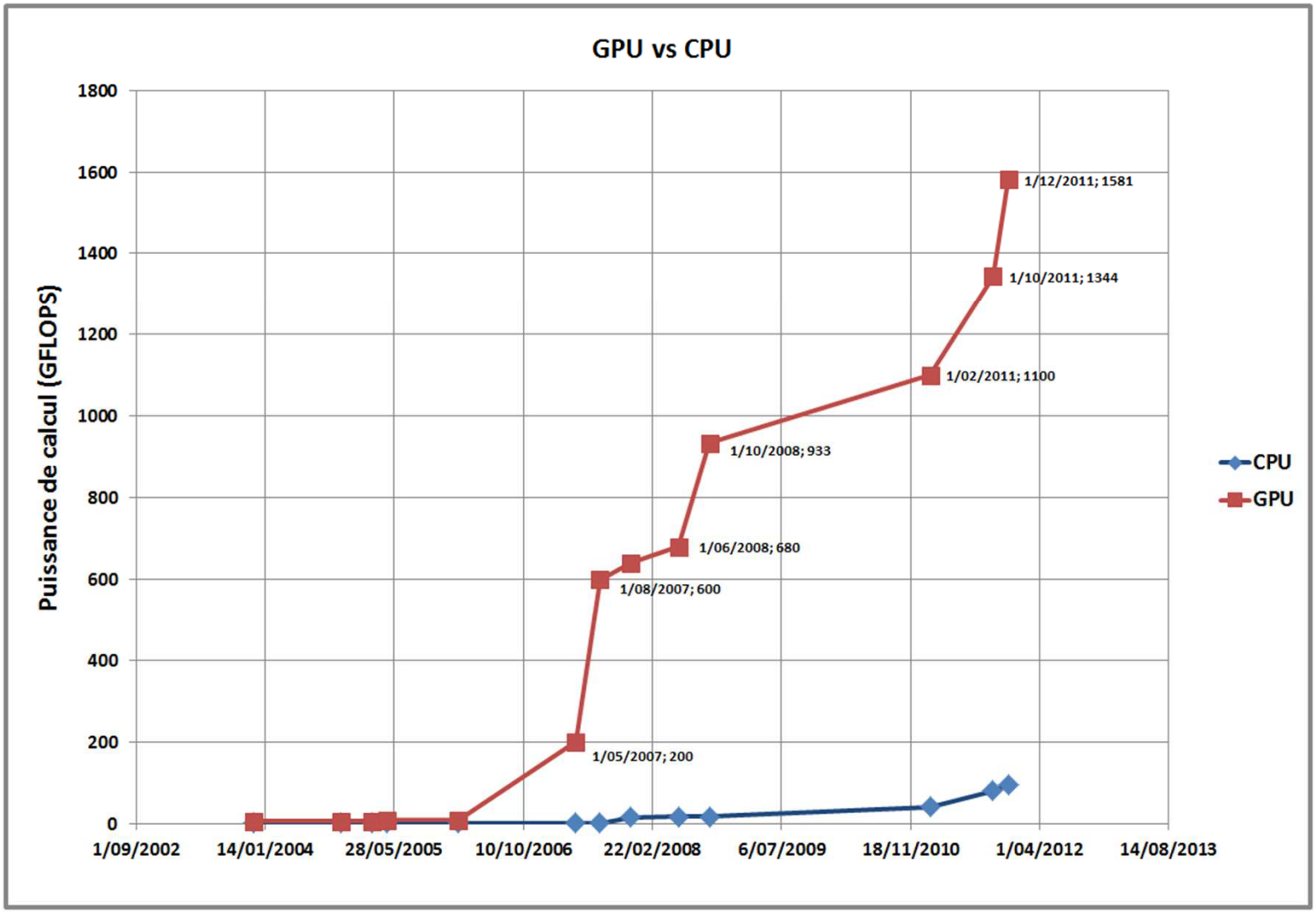


- **GPU:** Multiplication des unités de traitement

Présentation des GPU

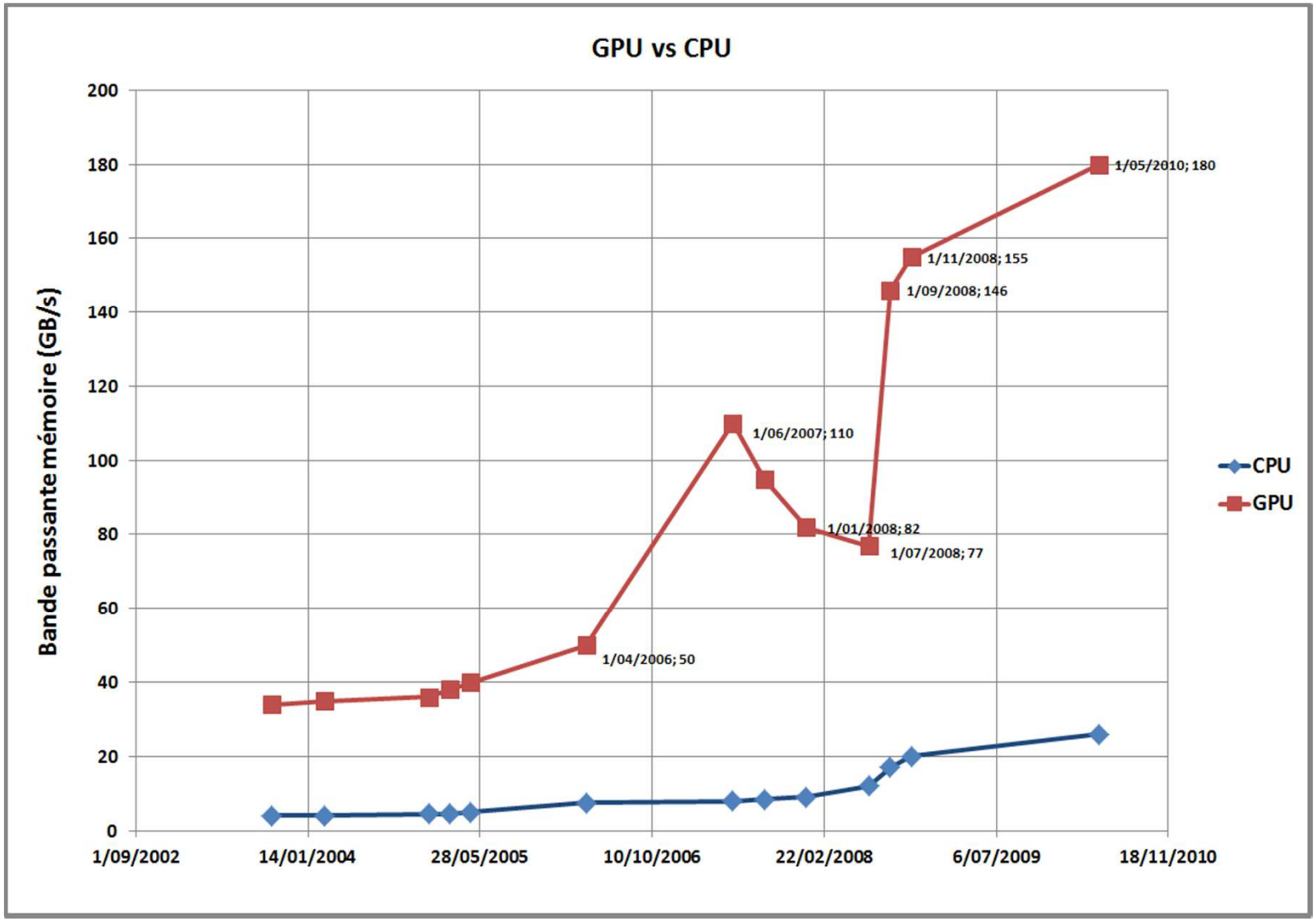


Présentation des GPU



Evolution de la puissance de pointe des CPU Intel et GPU Nvidia

Présentation des GPU



Evolution de la bande passante CPU vs. GPU

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

VI. Résultats Expérimentaux: cas d'utilisations du modèle

Conclusion

Programmation des GPUs

- **Brook GPU :** Existe depuis 2004
- **ATI Stream :** Pour les cartes ATI
- **DirectX 11, OpenGL :** Shaders GPGPU
- **OpenCL :** Compatible avec tous types de cartes

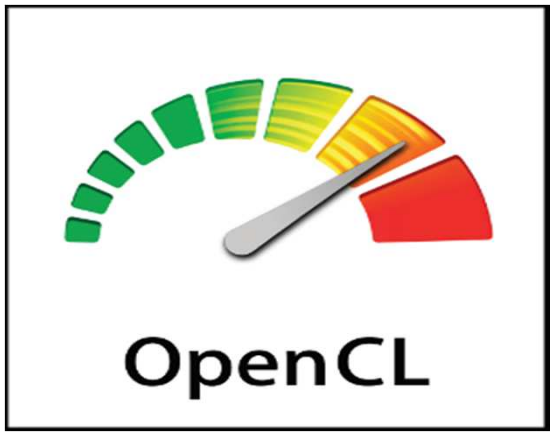
- **CUDA :** Pour les cartes NVIDIA

Ati Stream

- Créé par AMD/ATI en 2008.
- En concurrence avec NVIDIA
- Pour les cartes ATI uniquement
- Distribution des charges entre les cœurs CPU et GPU
- Inclut différentes librairies: FFT, LAPACK, BLAS

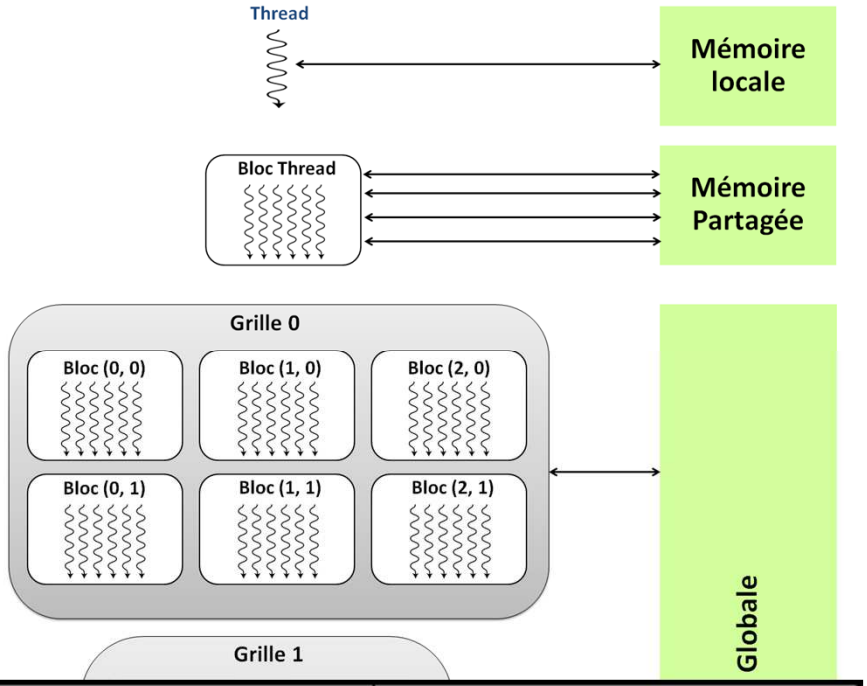
OpenCL

- Lancé par Apple afin d'unifier l'utilisation des multicoeurs et de GPU
- Rejoint par AMD/ATI et NVIDIA
- Première version en 2009
- Compatible avec les cartes NVIDIA et Ati
- Plus récent que CUDA : moins puissant !!
- Nouveau types de données (vecteur, image, etc.).



CUDA

- Compatible avec les GPU NVIDIA.
- Syntaxe proche du C.
- Grille: ensemble de blocs.
- Bloc: ensemble de threads.
- Threads synchronisés à travers la



Type of Memory	Utility	Size	Latency (horloge cycles)
Global	Principal Memory	1Go	400 to 600
Shared	Cooperation between threads	16 Ko	4
Registers	Local memory to each thread	100 byte per thread	1

CUDA

- Exemple d'addition de vecteurs :

$$\begin{pmatrix} A[1] \\ A[2] \\ \vdots \\ A[N] \end{pmatrix} + \begin{pmatrix} B[1] \\ B[2] \\ \vdots \\ B[N] \end{pmatrix} = \begin{pmatrix} C[1] \\ C[2] \\ \vdots \\ C[N] \end{pmatrix}$$

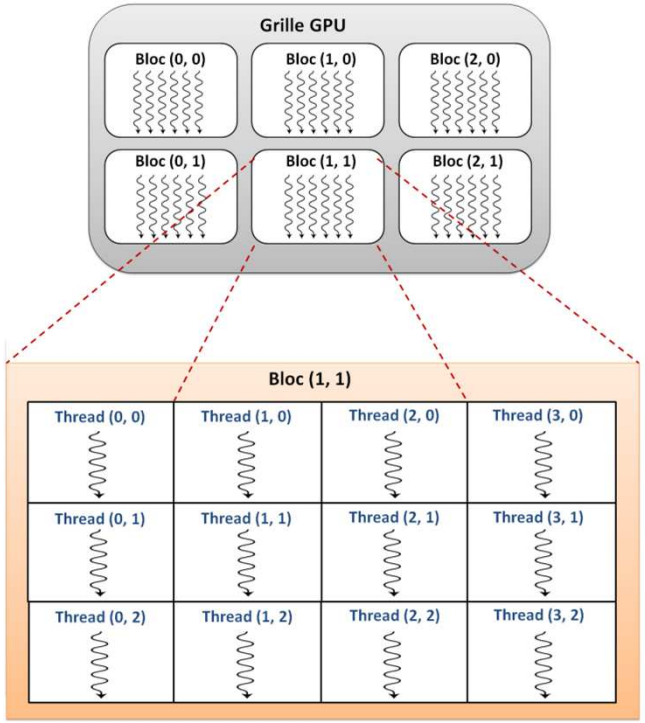
```
__global__ void vecAdd(float* A, float* B, float* C)
{
        int i = threadIdx.x;
        C[i] = A[i] + B[i];
}

int main()
{
        // Kernel invocation
        vecAdd<<<1, N>>>(A, B, C);
}
```

Kernel

Appel du Kernel

CUDA

$$\begin{pmatrix} A[1] \\ A[2] \\ \vdots \\ A[N] \end{pmatrix} + \begin{pmatrix} B[1] \\ B[2] \\ \vdots \\ B[N] \end{pmatrix} = \begin{pmatrix} C[1] \\ C[2] \\ \vdots \\ C[N] \end{pmatrix}$$




```
vecAdd<<<1, N>>>(A, B, C);
```
→ **1 Block de N threads**

```
vecAdd<<<2, N/2>>>(A, B, C);
```
→ **2 Blocks de N/2 threads**

```
vecAdd<<<100, N/100>>>(A, B, C);
```
→ **100 Blocks de N/100 threads**

CUDA

Un programme CUDA consiste en cinq étapes :

1. Allocation de mémoire sur GPU
2. Transfert des données depuis la mémoire CPU vers la mémoire GPU
3. Définition du nombre de threads
4. Lancement des fonctions CUDA en parallèle
5. Transfert des résultats depuis la mémoire GPU vers la mémoire CPU

CUDA

Exemple d'addition de matrices: allocation de mémoire

- Allocation de mémoire sur GPU

- La mémoire allouée recevra les données depuis la mémoire centrale

```
float *Ad, *Bd, *Cd;
const int size = N*N*sizeof(float);        // Taille de la matrice

cudaMalloc((void **)&A_d, size);           // Allouer la matrice A_d

cudaMalloc((void **)&B_d, size);           // Allouer la matrice B_d

cudaMalloc((void **)&C_d, size);           // Allouer la matrice C_d
```

CUDA

Exemple d'addition de matrices: transfert des données (CPU vers GPU)

- Transfert des données depuis la mémoire CPU vers la mémoire GPU
- Données initialisées sur la mémoire centrale
- Préciser le type de transfert : **HostToDevice**

```
cudaMemcpy(A_d, A, size, cudaMemcpyHostToDevice);
cudaMemcpy(B_d, B, size, cudaMemcpyHostToDevice);
   // A and B représentent les matrices provenant de la mémoire CPU
```

CUDA

Exemple d'addition de matrices : définir le nombre de threads

- Définir le nombre de threads sur GPU:
  - Définir la taille de la grille
  - Définir la taille des blocs
- Le nombre de threads dépend de la taille des données à traiter

```
dim3 dimBlock( blocksize, blocksize );          // Taille du bloc

dim3 dimGrid( N/dimBlock.x, N/dimBlock.y);      // Taille de la grille
```

CUDA

Exemple d'additin de matrices : lancement des kernels CUDA

- Définir la fonction CUDA (kernel)
- Appeler et exécuter la fonction CUDA
- Spécifier le nombre de threads déjà défini

```
__global__ void add matrix( float* a, float *b, float *c, int N )
{
   int i = blockIdx.x * blockDim.x + threadIdx.x;
   int j = blockIdx.y * blockDim.y + threadIdx.y;
   int index = i + j*N;
   if ( i < N && j < N )
      c[index] = a[index] + b[index];
}

add_matrix<<<dimGrid, dimBlock>>>( A_d, B_d, C_d, N );
```

CUDA

Exemple d'addition de matrices: transfert des résultats (GPU vers CPU)

- Résultats transférés vers la mémoire CPU une fois le traitement fini
- Préciser le type de transfert : **DeviceToHost**
- Libérer les espaces alloués sur la mémoire GPU

```
cudaMemcpy( C, C_d, size, cudaMemcpyDeviceToHost );

cudaFree( A_d );

cudaFree( B_d );

cudaFree( C_d );

// Les résultats seront sauvegardés sur la mémoire CPU « C ».
```

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

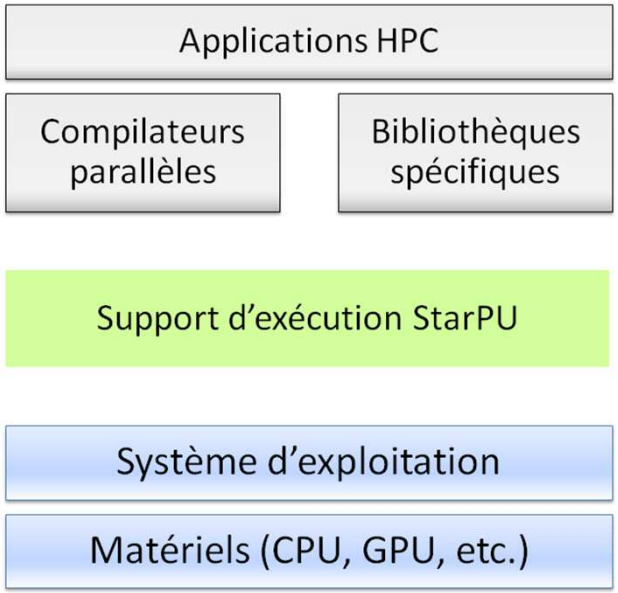
VI. Résultats Expérimentaux: cas d'utilisations du modèle

Conclusion

Exploitation des architectures hétérogènes

StarPU

- Développé aux laboratoires LABRI et INRIA Bordeaux SO
- Exploitation simultanée des cœurs CPUs et GPUs multiples
- Fonctions implémentées en C, CUDA ou OpenCL
- Stratégies d'ordonnancement efficaces



Aperçu de StarPU

Exploitation des architectures hétérogènes

StarSS

- Développé à l'université de Catalunia (Barcelone)
- Modèle de programmation flexible pour les multi- cœurs
- Basé essentiellement sur:

  - **CellSS:** programmation des processeurs Cell
  - **SPMs:** programmation des processeurs SPMs
  - **ClearSpeedS:** programmation des processeurs ClearSpeed
  - **GPUSS:** programmation Multi-GPU
  - **ClusterSs:** programmation des clusters
  - **GridSs:** programmes des ressources d'une grille

Exploitation des architectures hétérogènes

OpenACC

- API de haut niveau
- Supporté par CAPS entreprise, CRAY Inc, The Portland Group Inc (PGI) et nvidia
- Une collection de directives de compilation: boucle, etc.
- Prise en charge des tâches d'initialisation, lancement et arrêt d'accélérateurs
- Le lancement et l'arrêt d'accélérateurs est assuré par OpenACC
- Permet de fournir l'information aux compilateurs

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. **Application au traitement d'objets multimédias**

1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

VI. Résultats Expérimentaux: cas d'utilisations du modèle

Conclusion

Traitements intensifs

Gros volumes

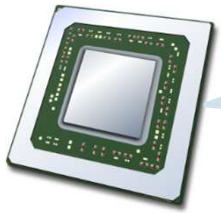
HD, Full HD, 4K, etc.

Traitement d'images

Traitement de vidéos

Imagerie médicale

Traitement d'objets multimédia



Multi-CPU

Temps de calcul ?

Temps réel ?

Coût ?

GPU ou Multi-GPU

Accélération

Calcul parallèle

Calcul hétérogène

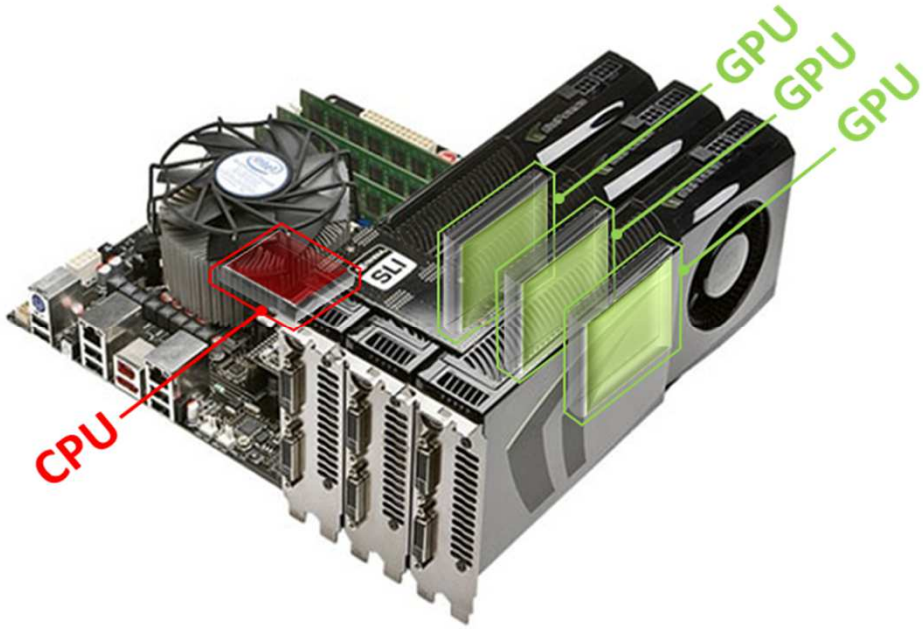
CPU ou/et GPU ?

Contexte et objectifs

Matériel

- Haute puissance de calcul des GPUs
- Architectures hétérogènes (Multi-CPU/Multi-GPU)

GPU



Contexte et objectifs

Matériel

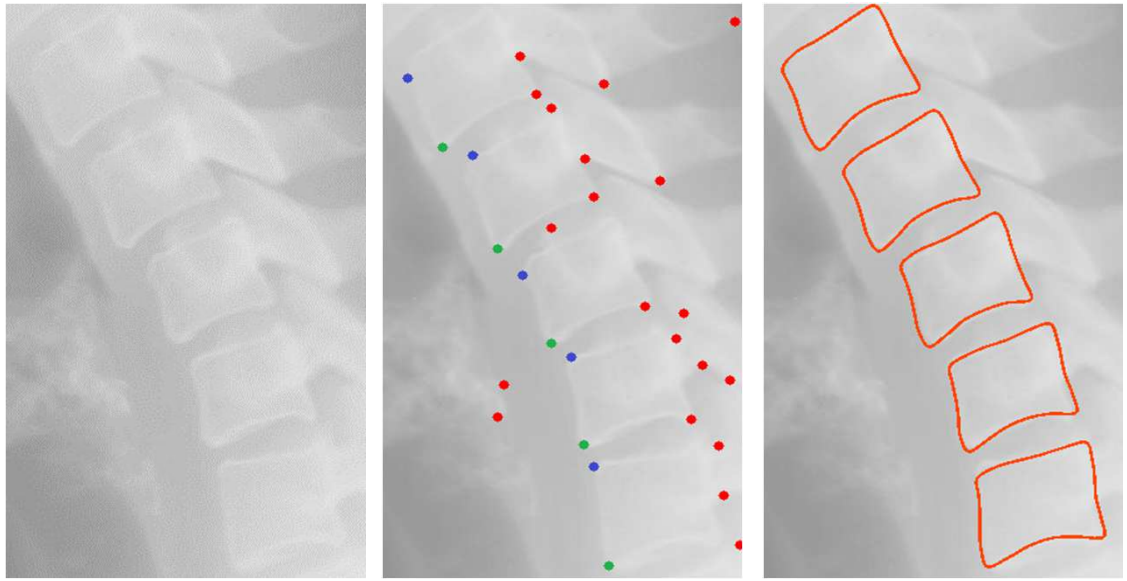
- Haute puissance de calcul des GPUs
- Architectures hétérogènes (Multi-CPU/Multi-GPU)

Applications

- Traitements intensifs d'objets multimédias (images, vidéos, etc.)
- Forte intensité : Volumes importants d'objets multimédias (HD, Full HD)

200 images

Environ 05 minutes



Application médicale [Lecron2011]

Contexte et objectifs

Matériel

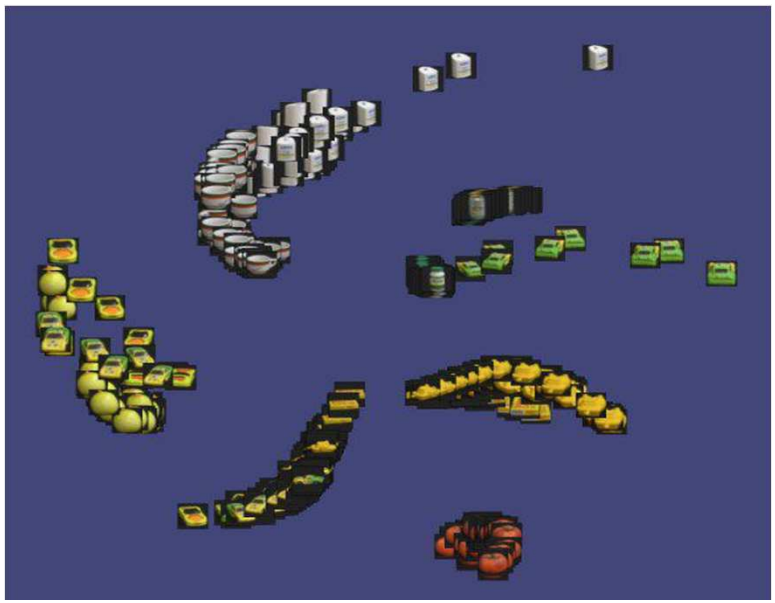
- Haute puissance de calcul des GPUs
- Architectures hétérogènes (Multi-CPU/Multi-GPU)

Applications

- Traitements intensifs d'objets multimédias (images, vidéos, etc.)
- Forte intensité : Volumes importants d'objets multimédias (HD, Full HD)

2000 images

Environ 10 minutes



Indexation d'images :MediaCycle [Siebert2009]

Contexte et objectifs

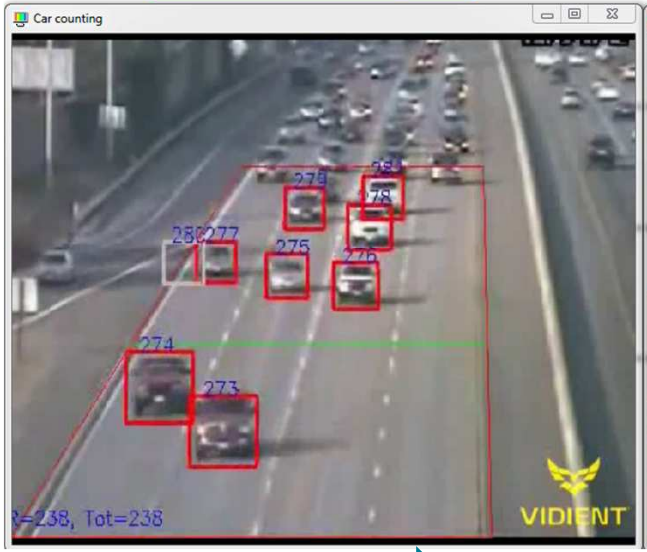
Matériel

- Haute puissance de calcul des GPUs
- Architectures hétérogènes (Multi-CPU/Multi-GPU)

Applications

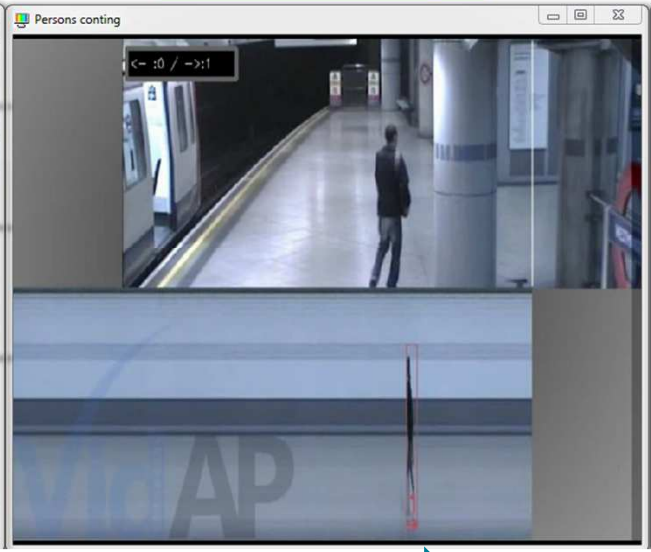
- Traitements intensifs d'objets multimédias (images, vidéos, etc.)
- Forte intensité : Volumes importants d'objets multimédias (HD, Full HD)

Car counting



Vidéo : 720x640 → 19 FPS

Persons conting



Vidéo : 640x376 → 20 FPS

High density crowd tracking

Vidéo : 720x640 → 8 FPS

Contexte et objectifs

Matériel

- Haute puissance de calcul des GPUs
- Architectures hétérogènes (Multi-CPU/Multi-GPU)

Applications

- Traitements intensifs d'objets multimédias (images, vidéos, etc.)
- Forte intensité : Volumes importants d'objets multimédias (HD, Full HD)

Contraintes

- Transferts coûteux entre mémoire CPU et mémoire GPU
- Choix adapté des ressources (CPU ou/et GPU)
- Ordonnancement efficace des tâches au sein des multiples CPU et GPU

Objectifs

- Gestion efficace des mémoires : réduction maximale des transferts
- Choix des ressources selon la nature des données et de méthodes
- Traitement rapide d'objets multimédias sur plateformes hétérogènes

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

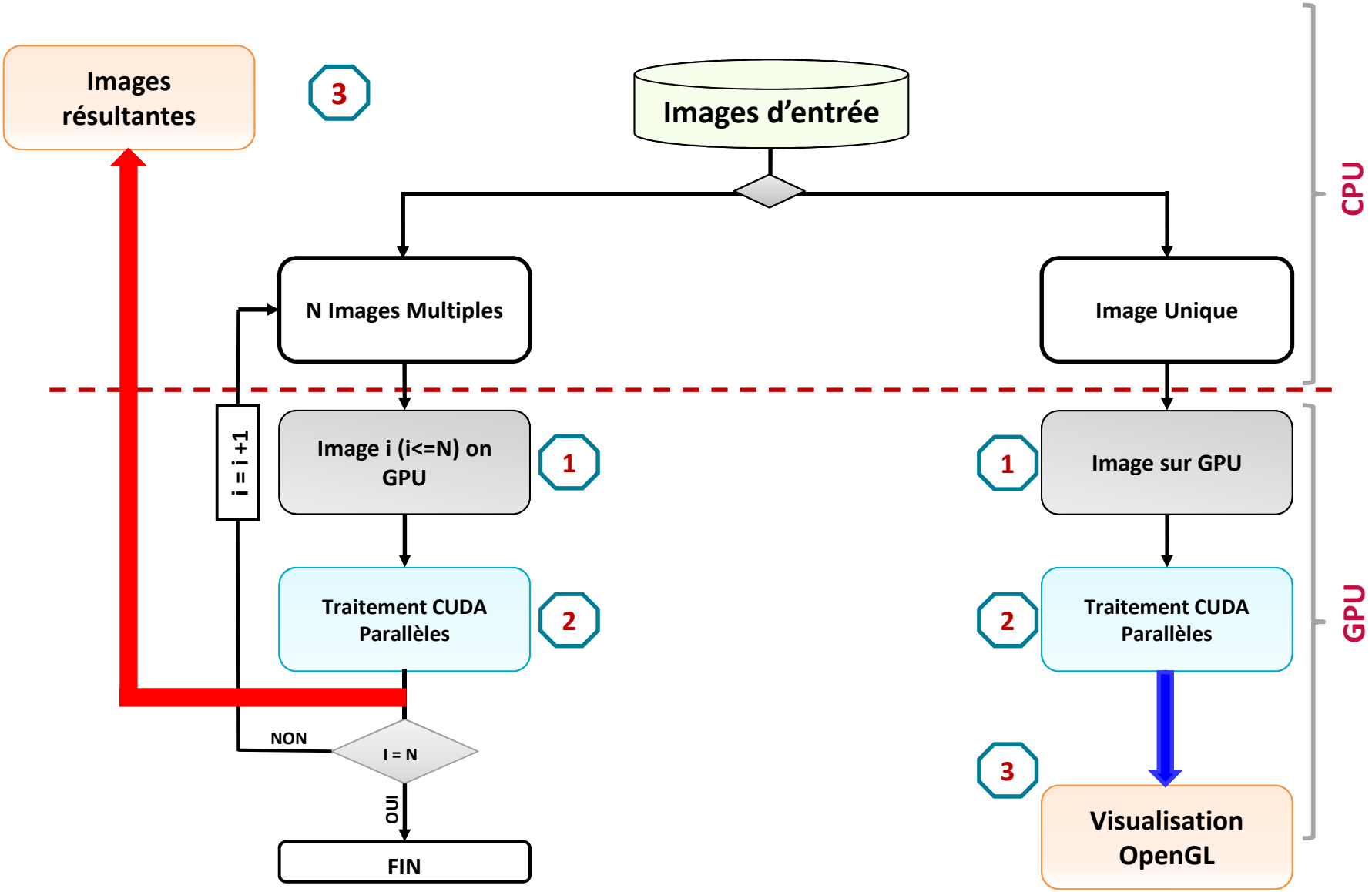
1. **Traitement d'images sur architectures Multi-CPU/Multi-GPU**
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

VI. Résultats Expérimentaux: cas d'utilisations du modèle

Conclusion

Schéma de développement proposé



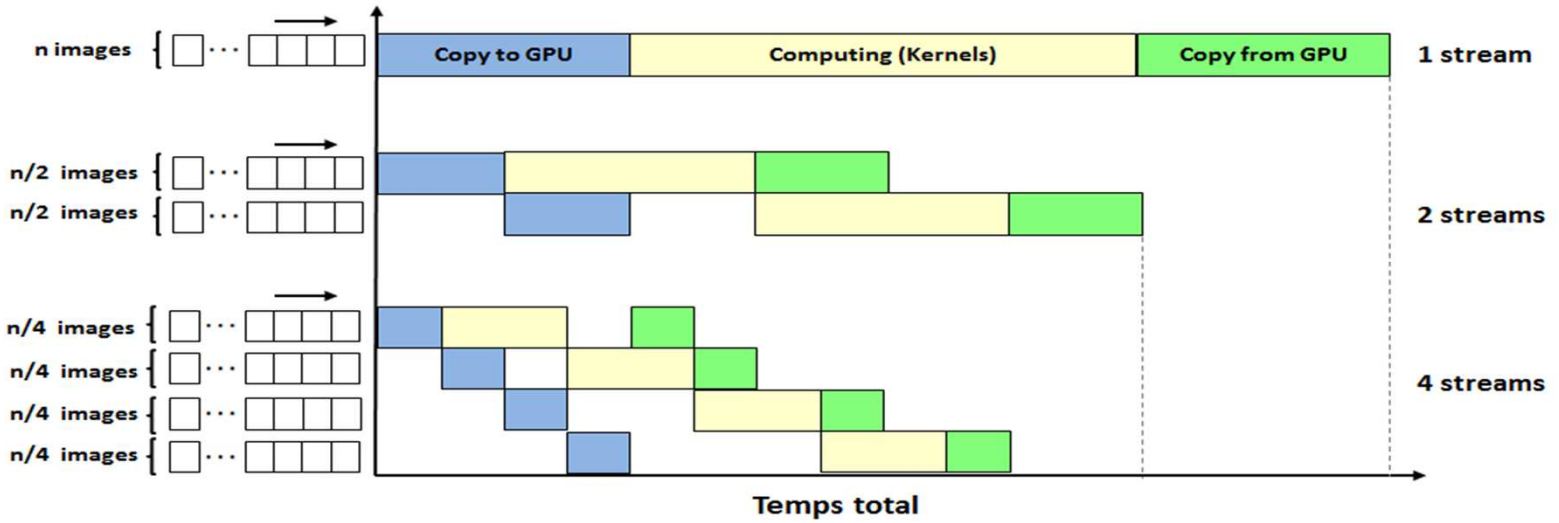
Optimisations GPU employées

Traitement d'image individuelle

- Mémoire texture : accès plus rapide au pixels
- Mémoire partagée : accès rapide au voisinage

Traitement d'images multiples

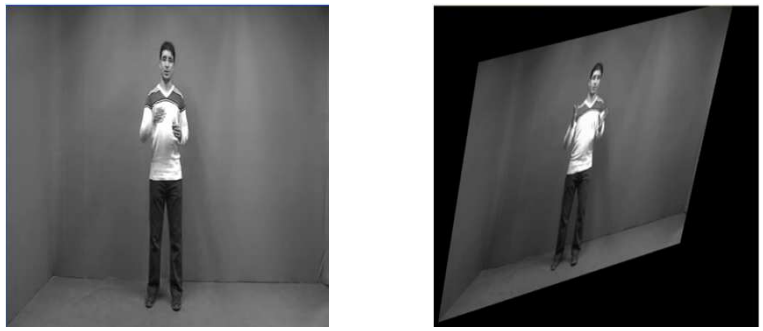
- Mémoires texture et partagée
- Recouvrement des transferts par les exécutions sur GPU : CUDA streams



Algorithmes implémentés sur GPU

Méthodes classiques de traitement d'images :

- Transformations géométriques :
  - rotation, translation, homographie
- Traitement parallèle entre pixels de l'image
- Accélération GPU: de **10x** à **100x**



Homographie sur GPU

Extraction de points d'intérêts sur GPU:

- Etape préliminaire à de nombreux processus de vision par ordinateur
- Implémentation GPU basée sur le principe de Harris
- Efficacité: invariance à la rotation, à la luminosité, à l'échelle, etc.

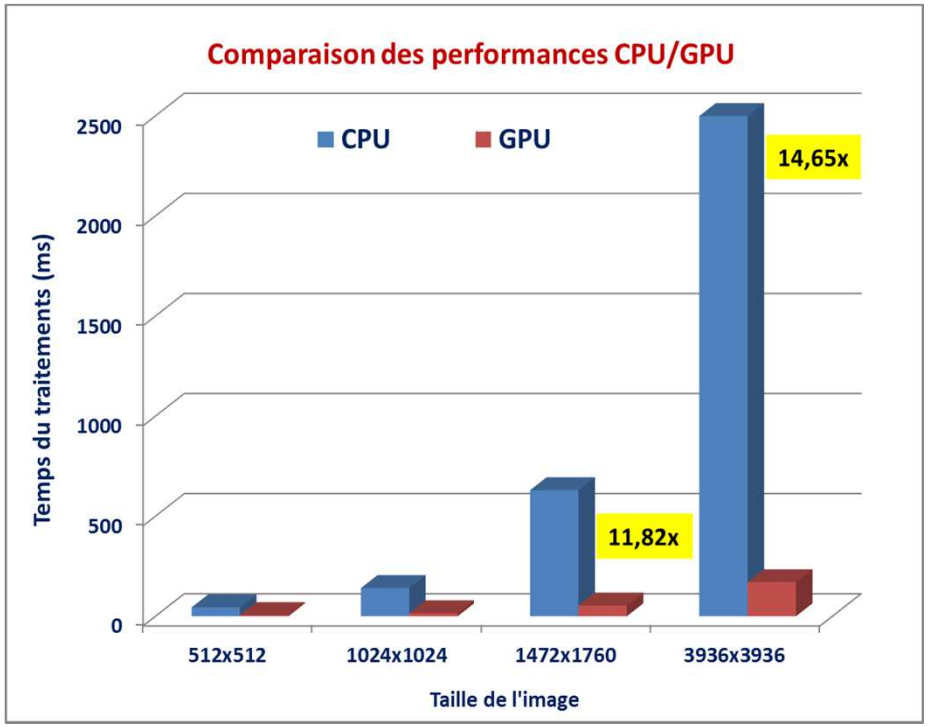
Détection des coins sur GPU

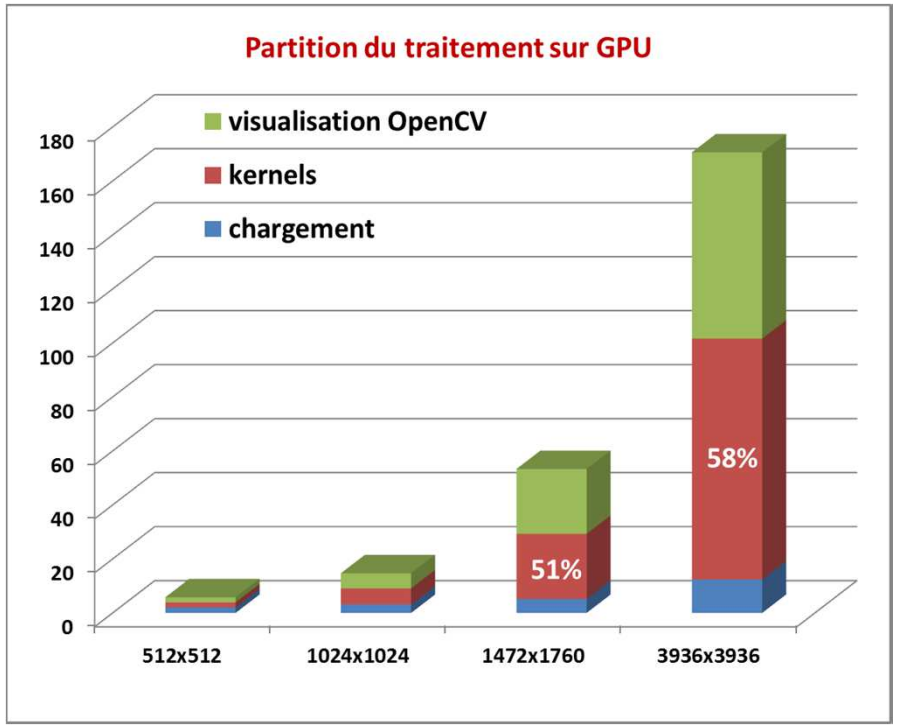
Détection des contours sur GPU:

- Implémentation GPU basée sur le principe de Deriche-Canny
- Efficacité: robustesse aux bruits, nombre réduit d'opérations
- Bonne qualité des contours extraits

Détection des contours sur GPU

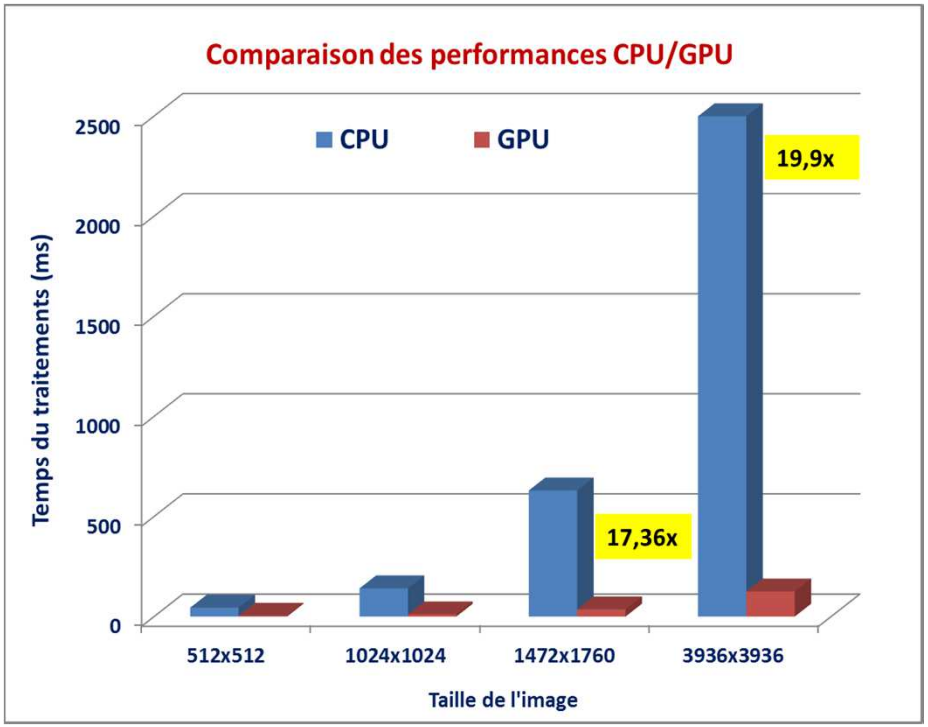
Résultats : traitement GPU d'image unique

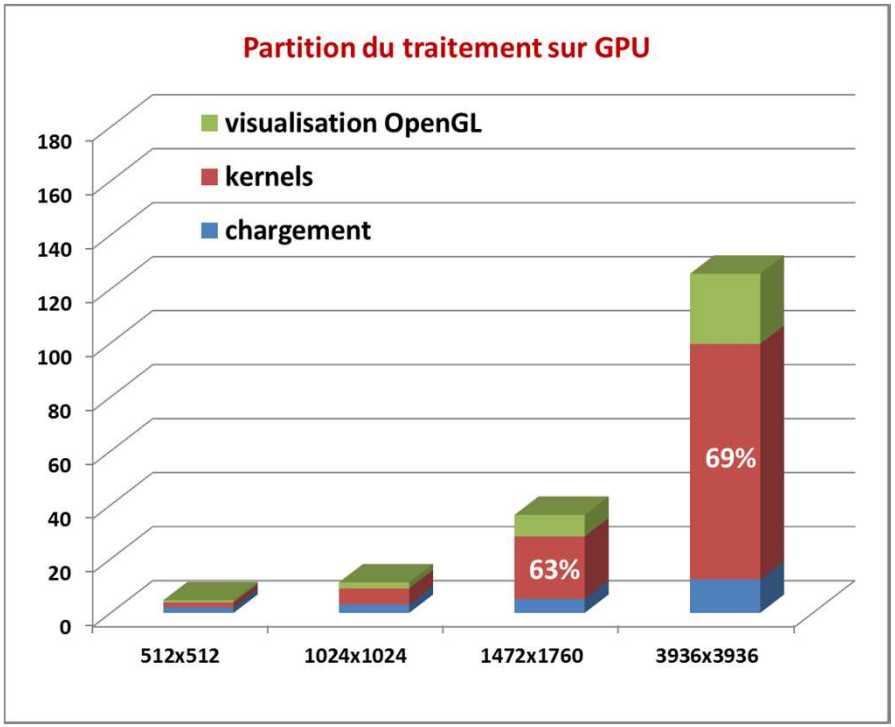




Détection des coins et de contours sur GPU : Mémoire Globale

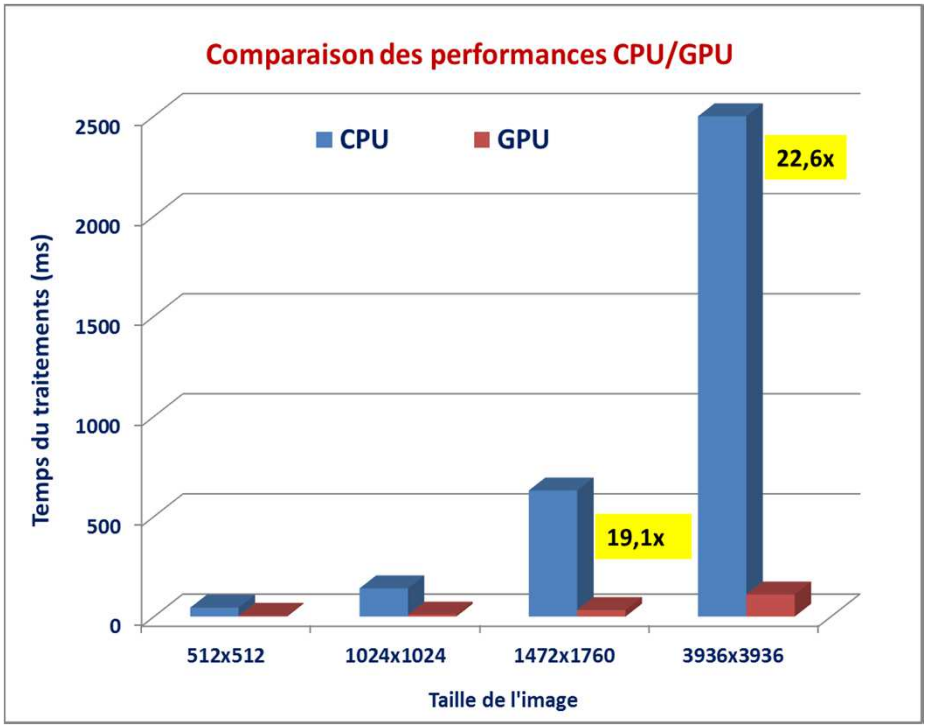
Résultats : traitement GPU d'image unique

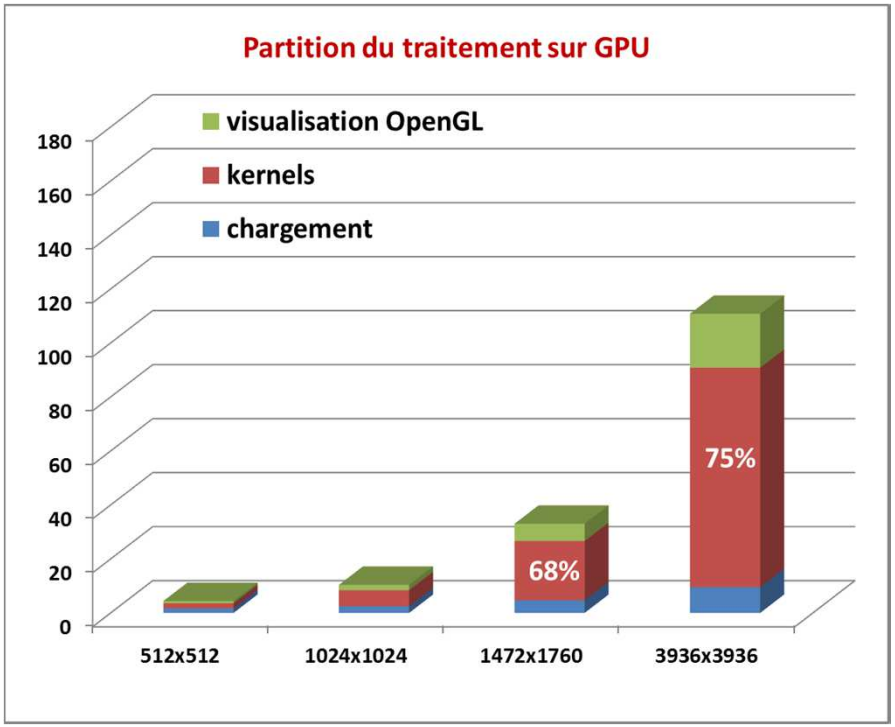




Détection des coins et de contours sur GPU : Visualisation OpenGL

Résultats : traitement GPU d'image unique

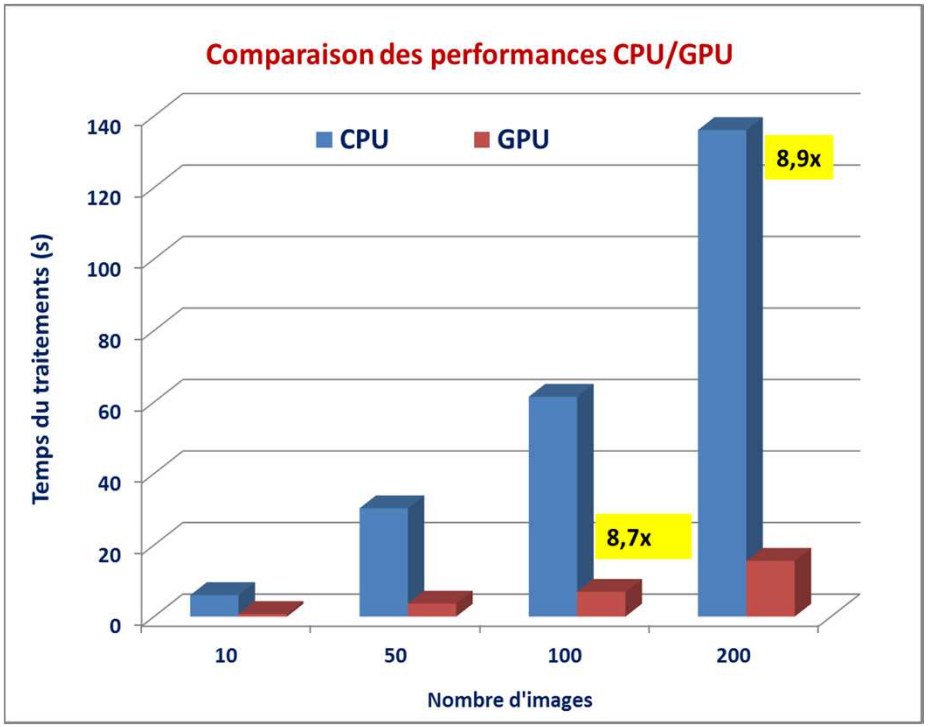


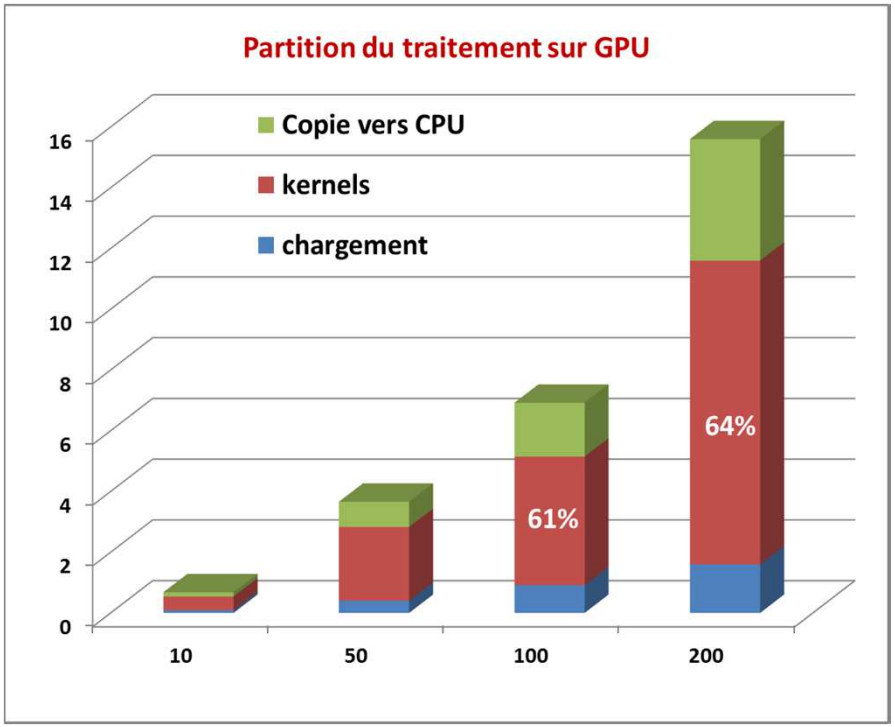


Détection des coins et de contours sur GPU : Mémoires « texture et partagée »

Résultats : traitement GPU d'images multiples

Résolution d'images : 2048x2048

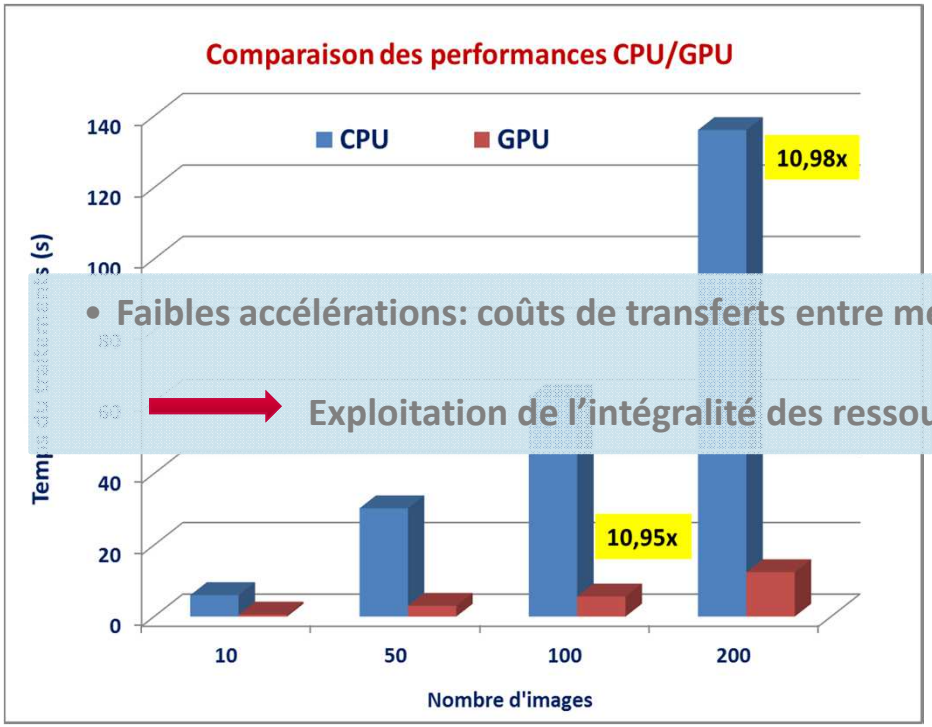


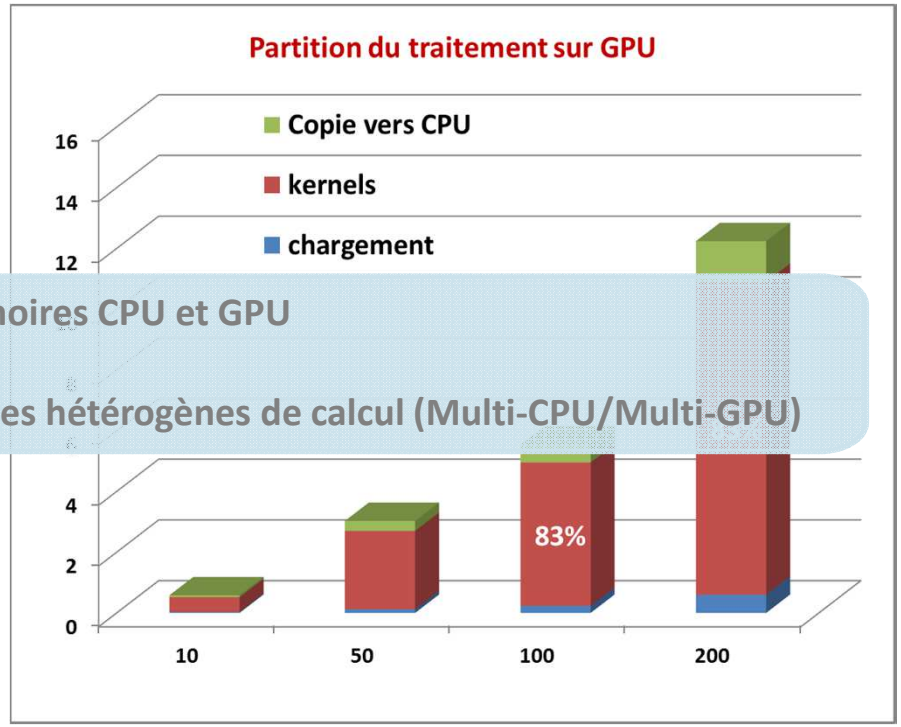


Détection des coins et de contours sur GPU : Mémoires « texture et partagée »

Résultats : traitement GPU d'images multiples

Résolution d'images : 2048x2048



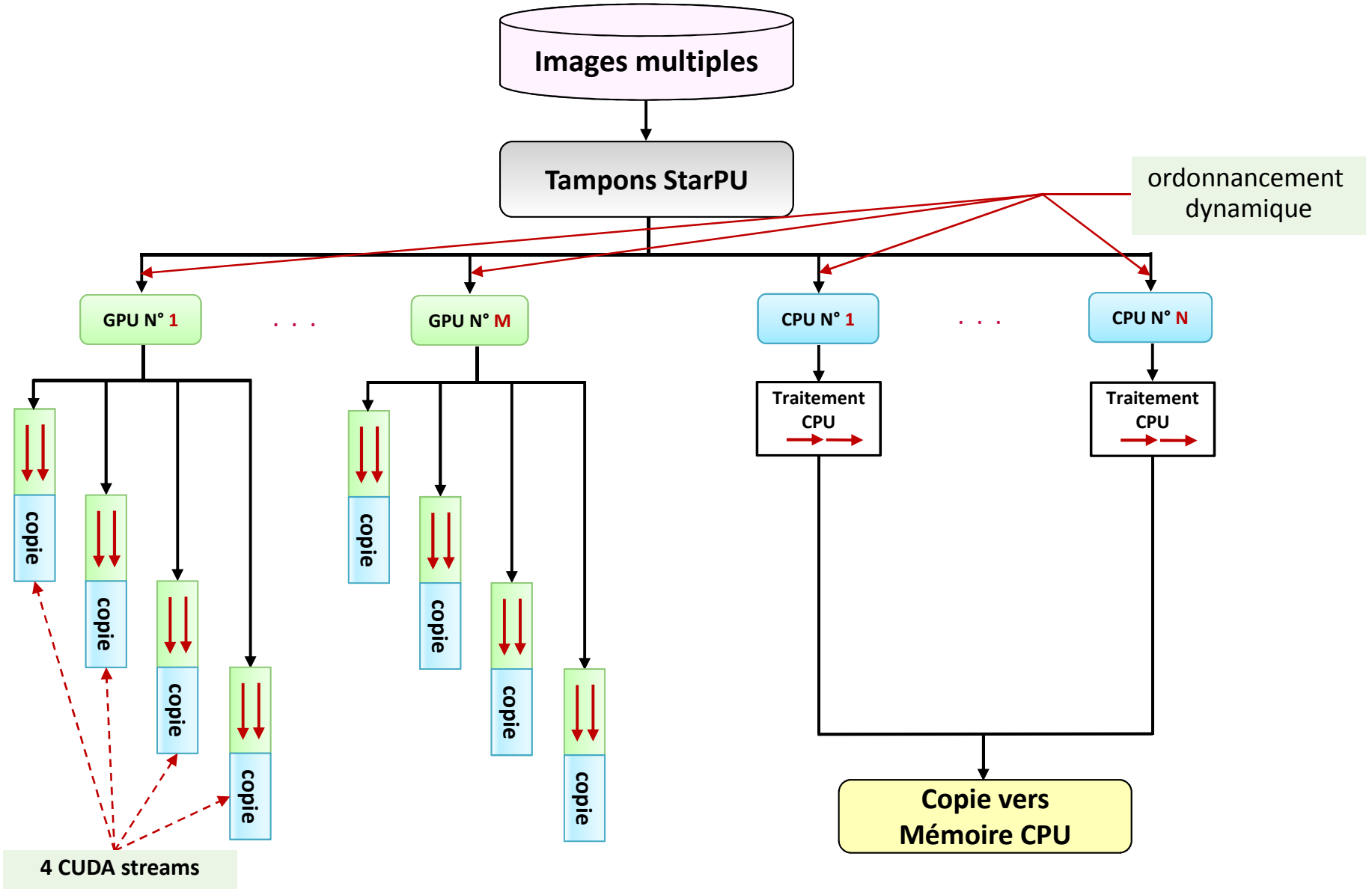


- Faibles accélérations: coûts de transferts entre mémoires CPU et GPU

→ Exploitation de l'intégralité des ressources hétérogènes de calcul (Multi-CPU/Multi-GPU)

Détection des coins et de contours sur GPU : CUDA streaming

Schéma 2 : traitement hétérogène d'images multiples



Traitement hétérogène: Optimisations employées

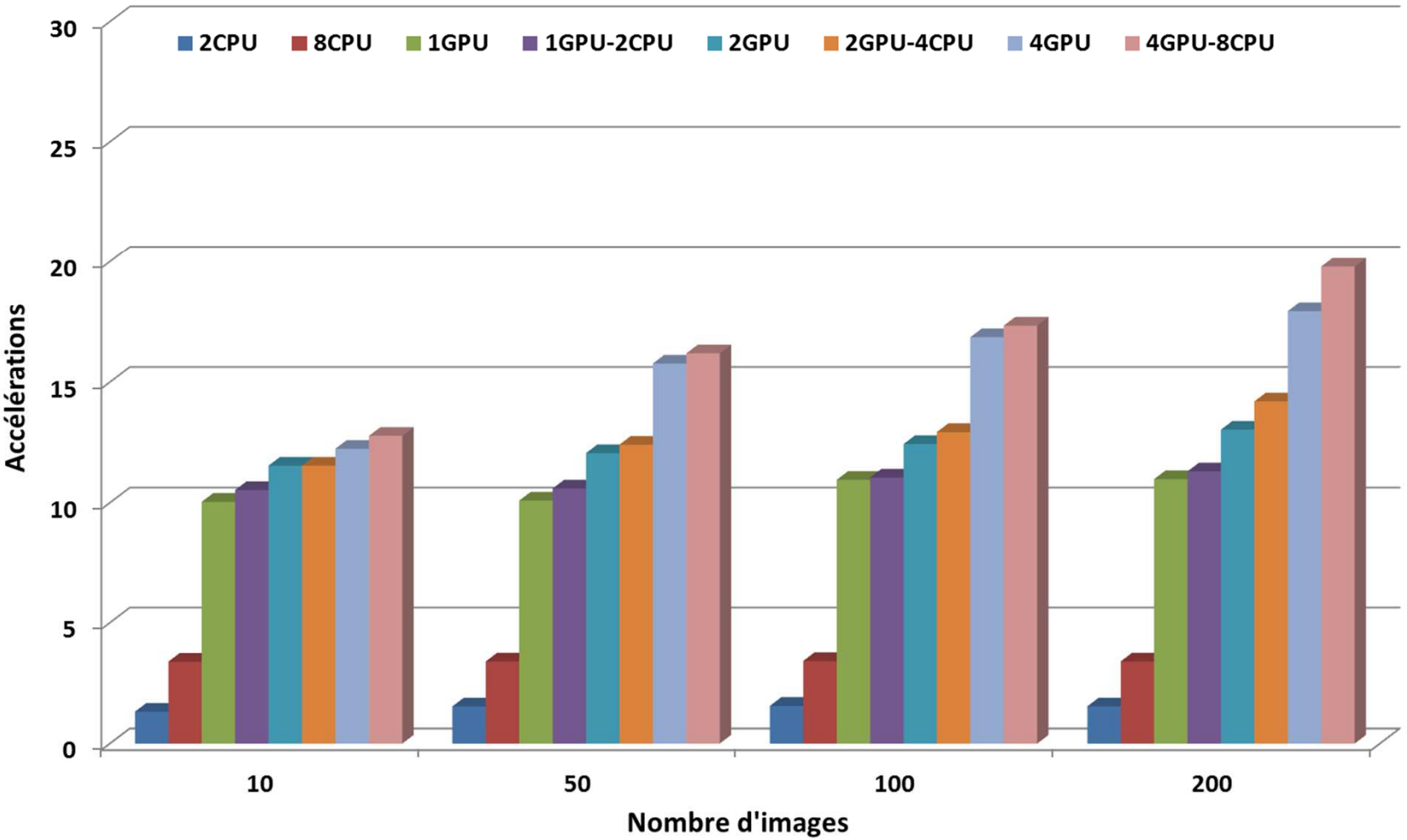
Ordonnancement des tâches:

- Appliqué au sein des cœurs hétérogènes (Multi-CPU/Multi-GPU)
- Estimation préliminaire de durée de tâches sur base de l'historique (tâches précédentes)
- Ordonnancement prenant en compte:
  - ✓ durées estimées des tâches
  - ✓ estimation du temps de transfert de données

CUDA streaming

- Appliqué au sein des GPUs multiples
- recouvrement des transferts par les exécutions
- Chaque GPU traite un sous-ensemble d'images utilisant 4 streams CUDA
- Le choix de quatre streams CUDA offre de meilleures performances

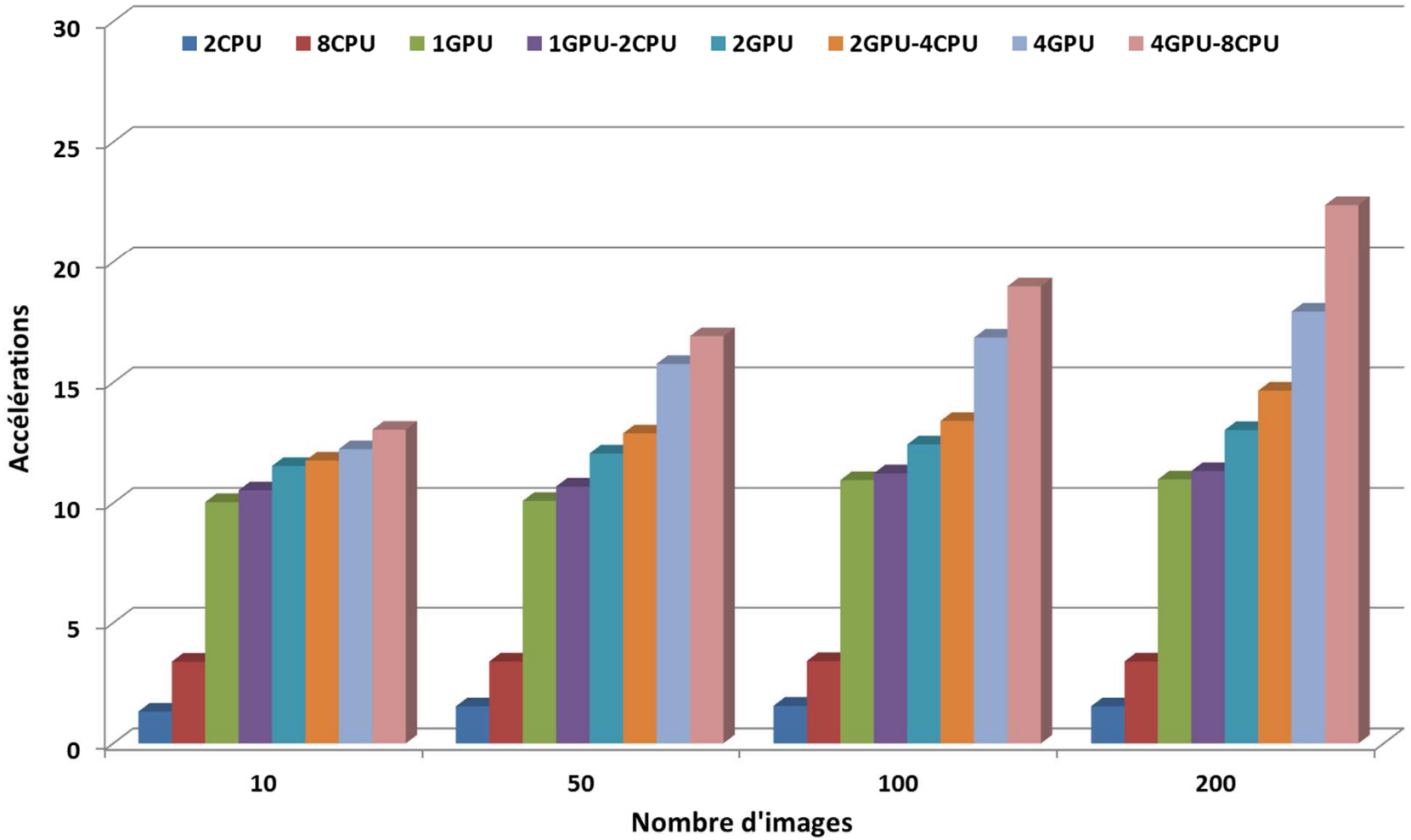
Traitement hétérogène : résultats



Résolution d'images : 512x512, 1024x1024, 2048x2048, 3936x3936

Détection hétérogène des coins et contours d'images multiples

Traitement hétérogène : résultats

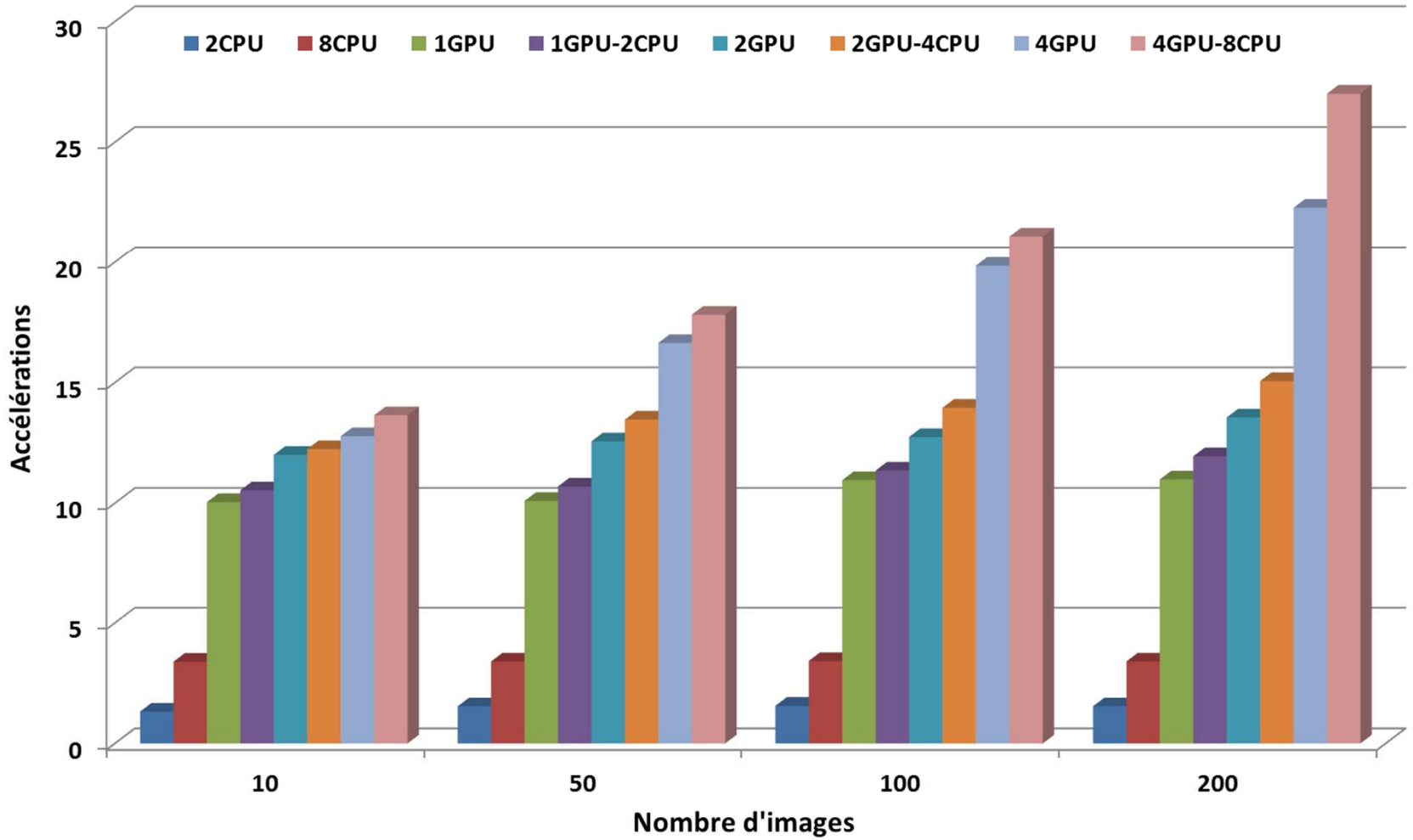


Résolution d'images : 512x512, 1024x1024, 2048x2048, 3936x3936

Détection hétérogène des coins et contours d'images multiples

Ordonnancement dynamique

Traitement hétérogène : résultats



Résolution d'images : 512x512, 1024x1024, 2048x2048, 3936x3936

Détection hétérogène des coins et contours d'images multiples

Ordonnancement dynamique + CUDA streaming

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

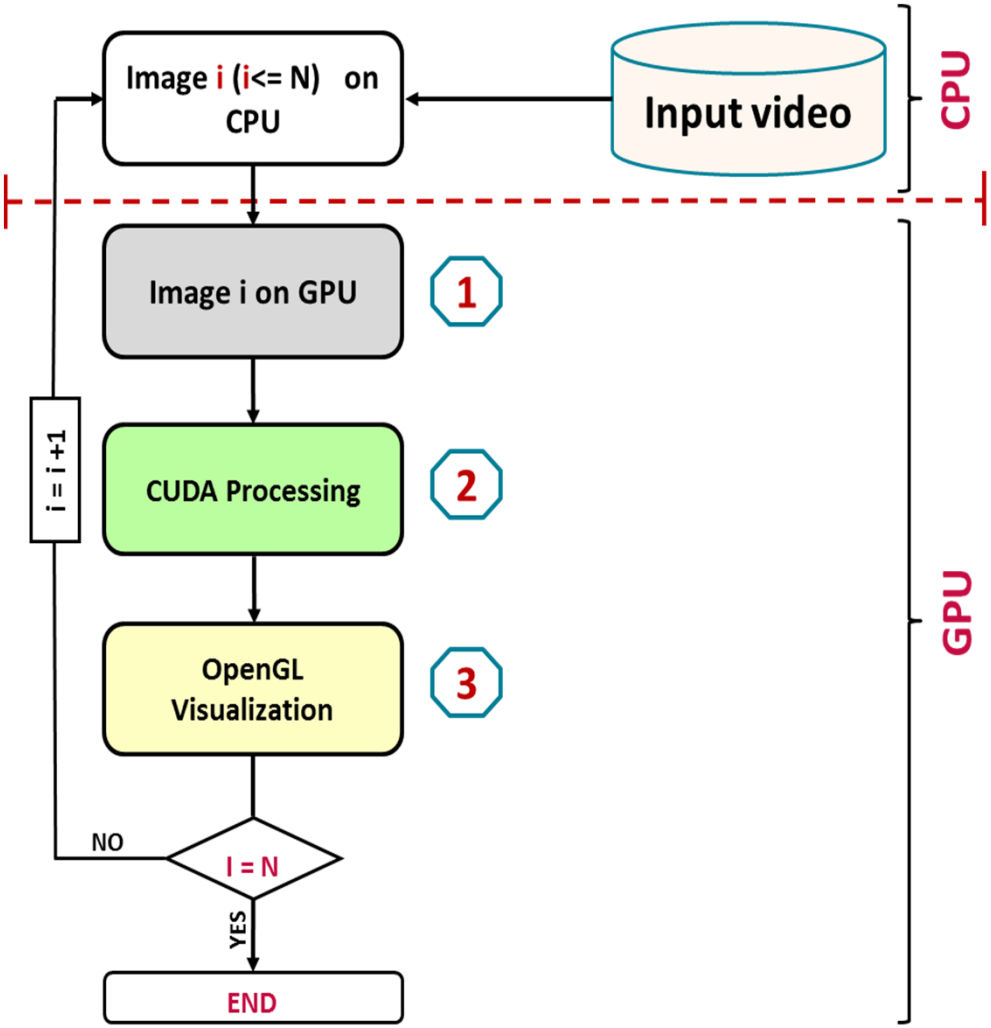
1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. **Traitement Multi-GPU de vidéos HD/Full HD en temps réel**

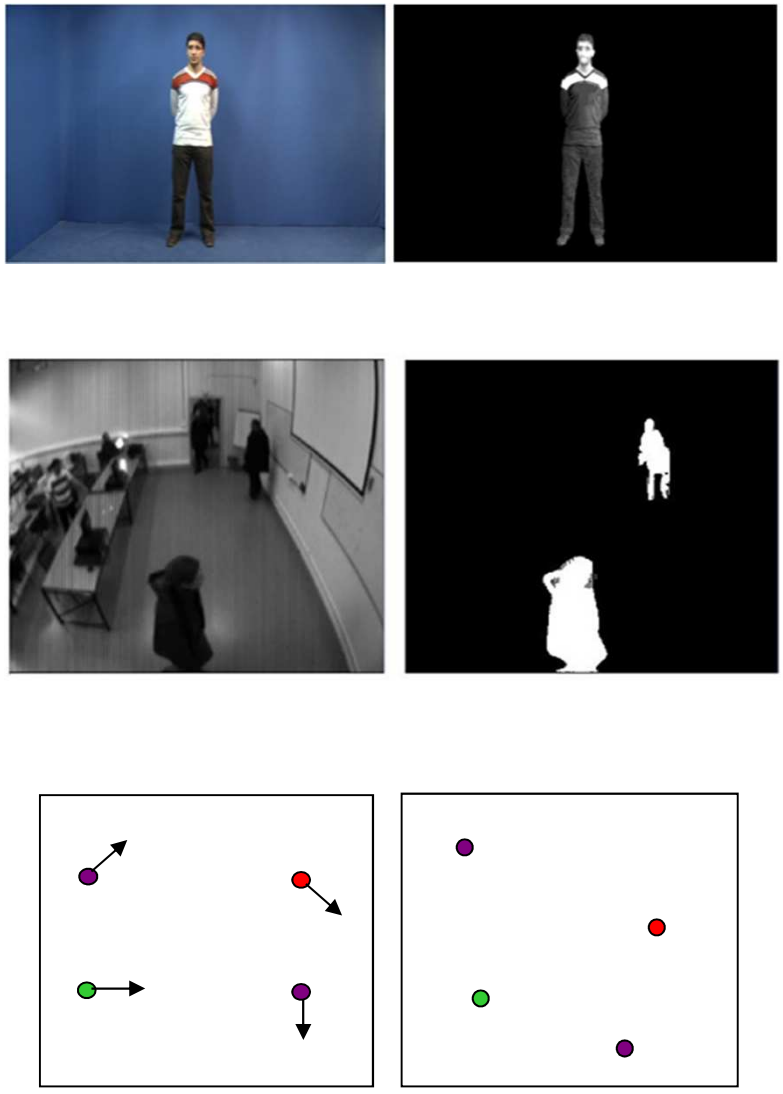
V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

VI. Résultats Expérimentaux: cas d'utilisations du modèle

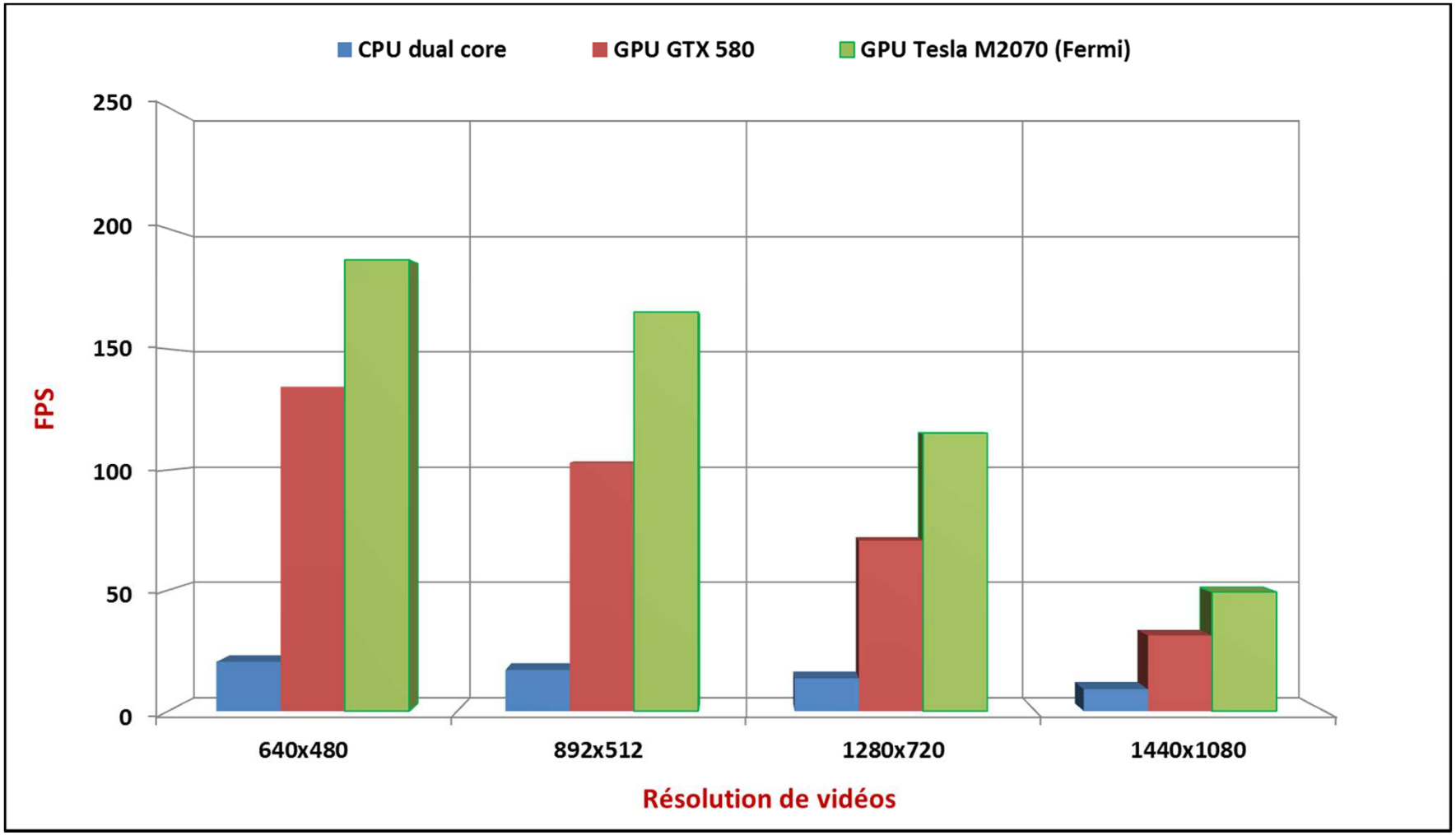
Conclusion

Traitement de vidéos HD sur GPU



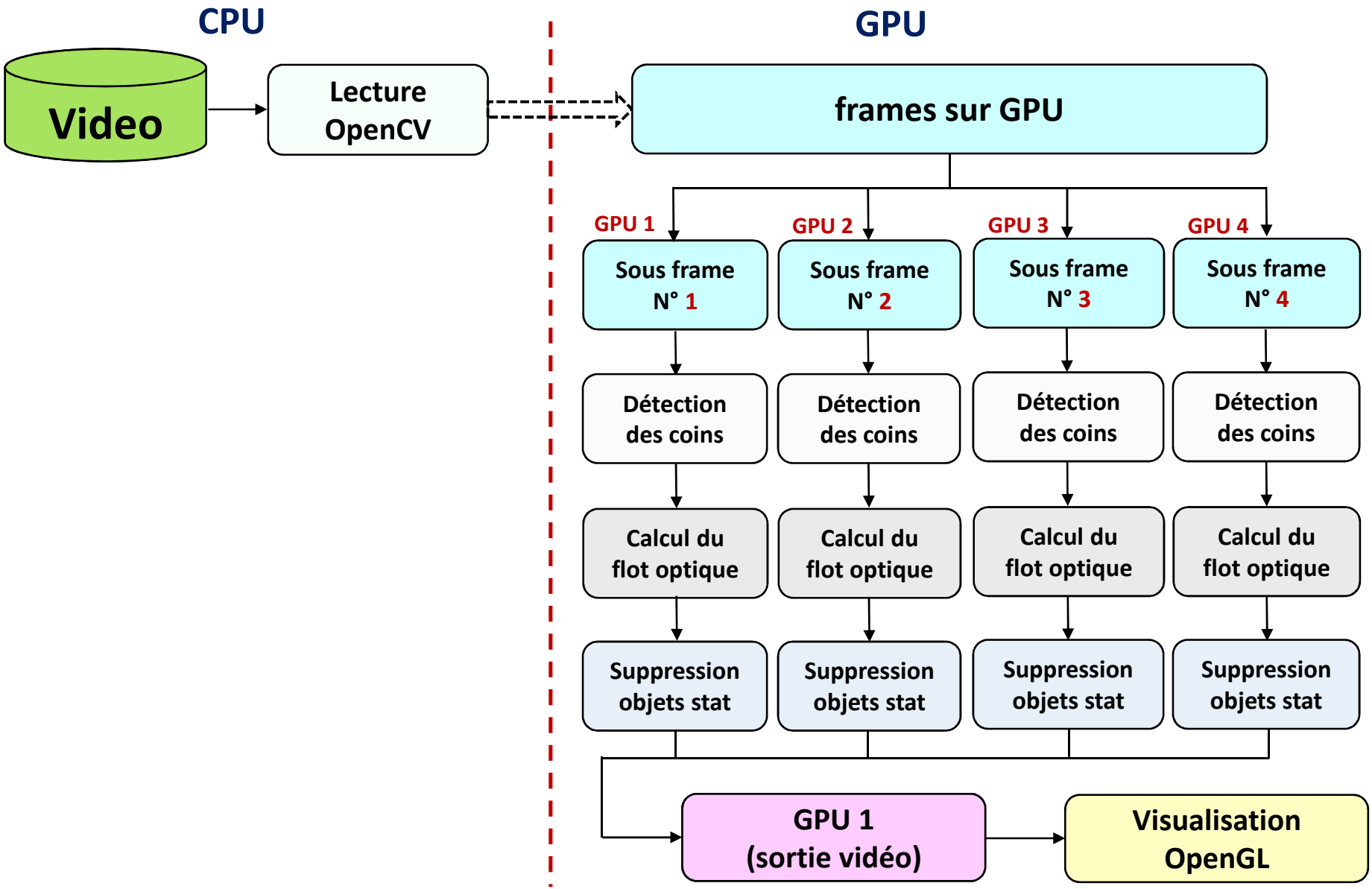


Extraction de silhouettes sur GPU

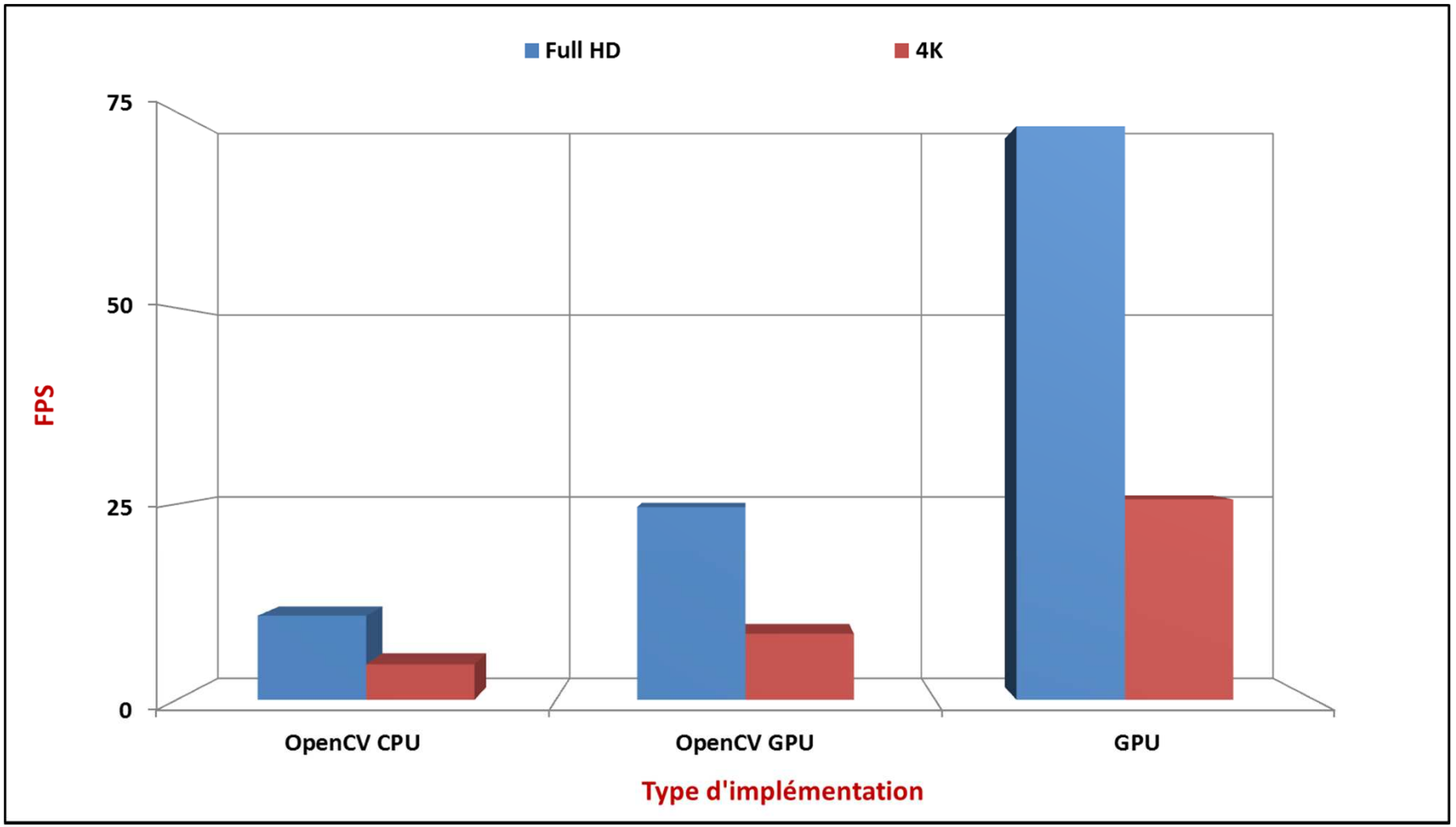


Performances d'extraction de silhouette sur GPU

Suivi de mouvements sur GPU multiples



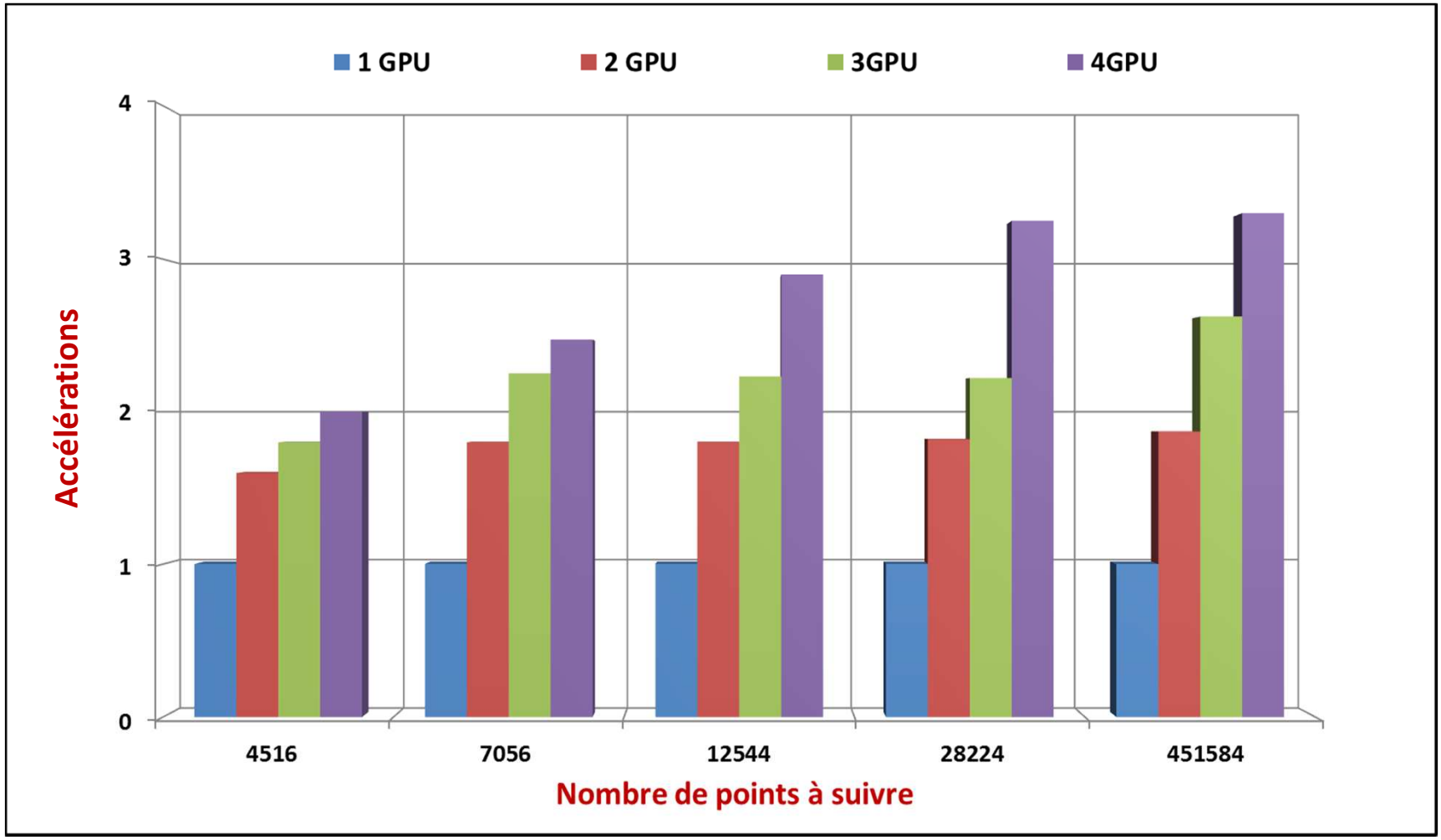
Suivi de mouvements sur un GPU



Performances de suivi de mouvements sur un GPU

HD : 1280x720 **Full HD : 1920x1080** **4K : 3840x2160**

Suivi de mouvements sur GPU multiples



Performances de suivi de mouvements sur GPU multiples

Suivi de mouvements sur GPU multiples

Suivi de mouvements dans une vidéo Full HD (1920x1080)

Suivi de mouvements sur GPU multiples

Suivi de mouvements dans une vidéo Full HD (1920x1080)

Suivi de mouvements sur GPU multiples

Suivi de mouvements dans une vidéo Full HD (1920x1080)

PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

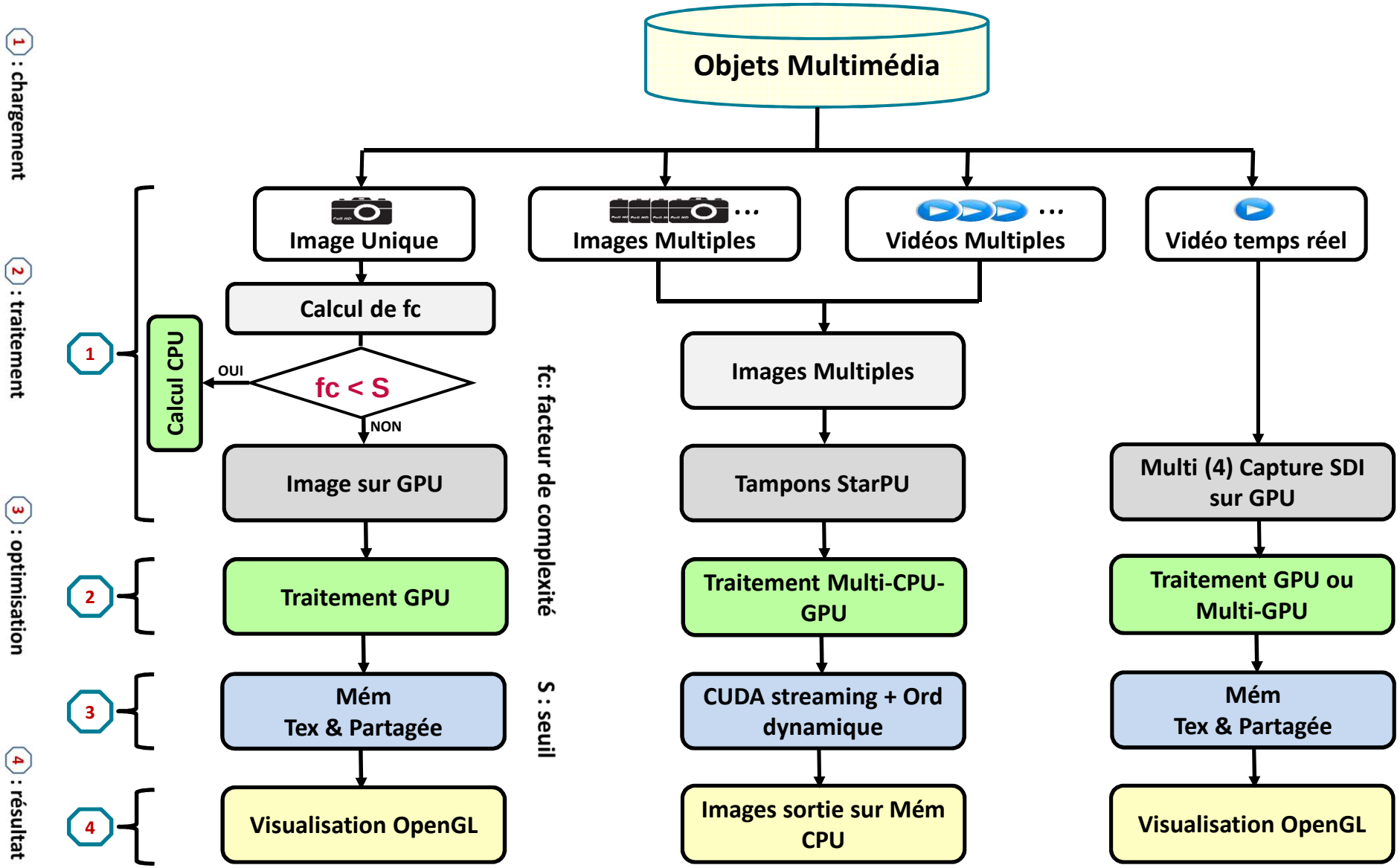
1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. **Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes**

VI. Résultats Expérimentaux: cas d'utilisations du modèle

Conclusion

Modèle de traitement d'objet multimédias



PLAN

Introduction

I. Présentation des GPU

II. Programmation des GPU

III. Exploitation des architectures hétérogènes Multi-CPU/Multi-GPU

IV. Application au traitement d'objets multimédias

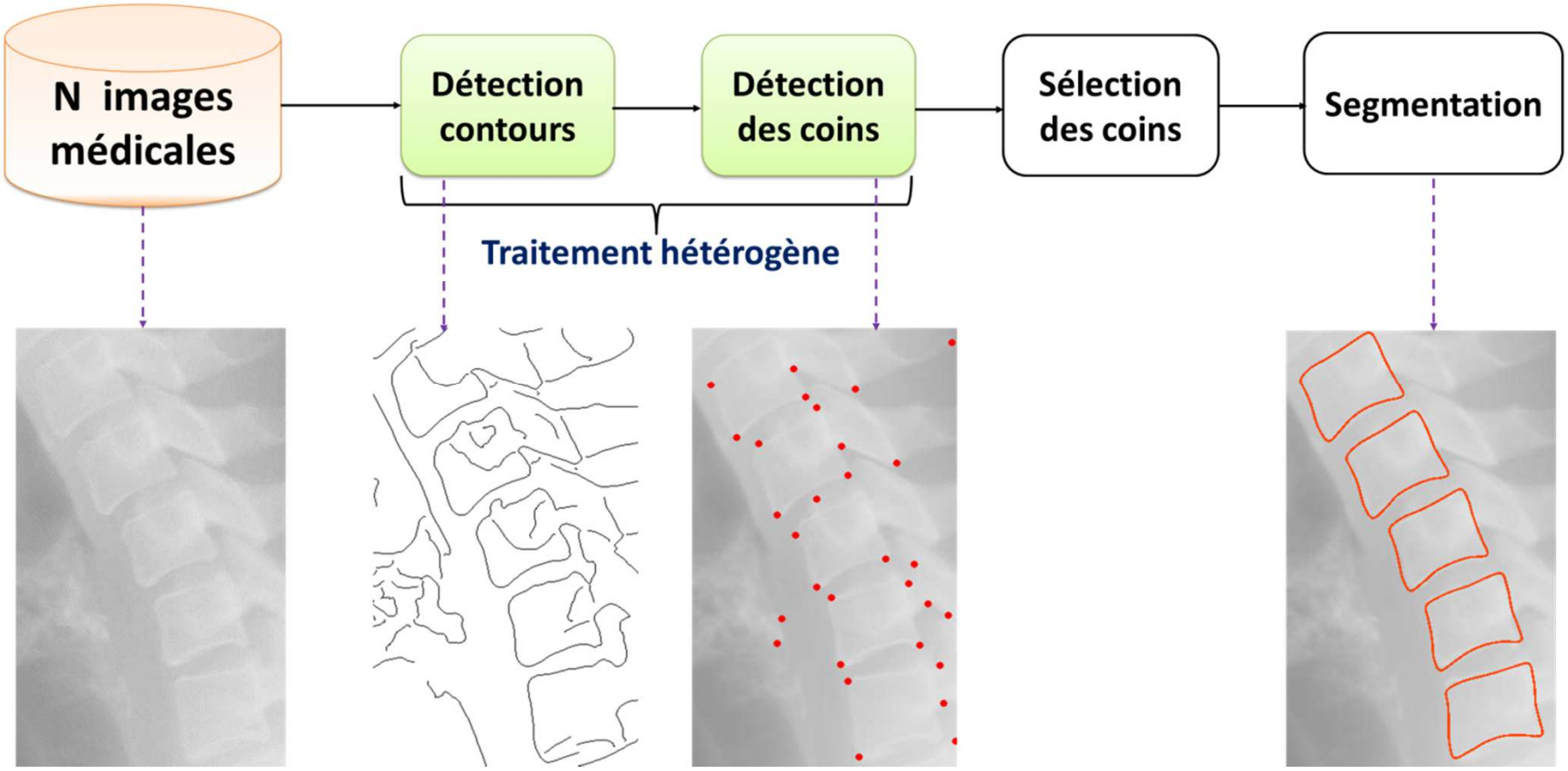
1. Traitement d'images sur architectures Multi-CPU/Multi-GPU
2. Traitement Multi-GPU de vidéos HD/Full HD en temps réel

V. Modèle de traitement d'images et de vidéos sur architectures parallèles et hétérogènes

VI. **Résultats Expérimentaux: cas d'utilisations du modèle**

Conclusion

1er cas : segmentation des vertèbres



Traitement hétérogènes d'images médicales : segmentation des vertèbres

1er cas : segmentation des vertèbres

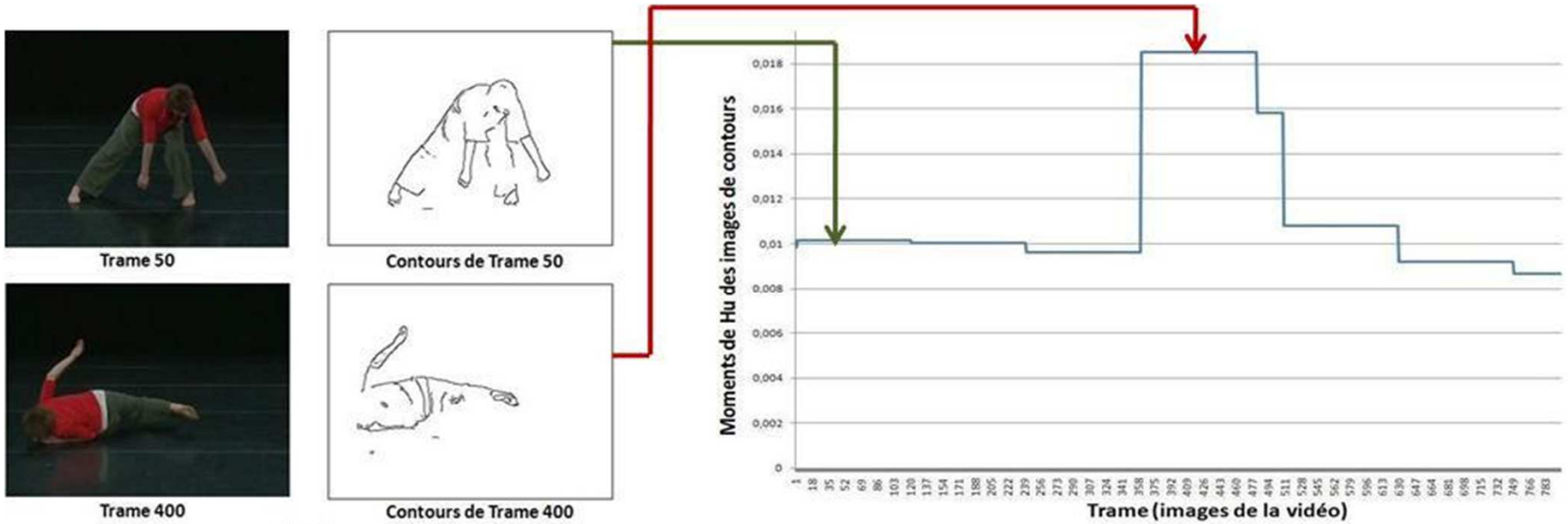
Etapes	1CPU	8CPU		1GPU/8CPU				4GPU/8CPU			
				1GPU		8CPU		8CPU		4GPU/8CPU	
	Temps(T)	T	Acc	T	Acc	T	Acc	T	Acc	T	Acc
Egalisation histogramme	62.10 *s*	15.44 *s*	4.02×	/		15.44 *s*	4.02×	15.44 *s*	4.02×	/	
Détection contours	135.8 *s*	39.06 *s*	3.48×	15.80 *s*	08.60×	/		/		4.84 *s*	28.06×
Détection coins (poly)	46.12 *s*	11.51 *s*	4.01×	/		11.51 *s*	4.01×	11.51 *s*	4.01×	/	
Temps total	T	T	Acc	T		Acc		T		Acc	
	244.02 *s*	66.01 *s*	3.70 x	42.75 *s*		5.71 x		**31.79 *s***		**7.68 x**	

Performances de détection hétérogène de vertèbres de 200 images (1476x1680)

2ème cas : indexation de vidéos

VideoCycle: Indexation de séquences vidéo à partir de caractéristiques d'images

- **Silhouette**
- **Zones de mouvements**
- **Contours**
- **Moments de Hu**

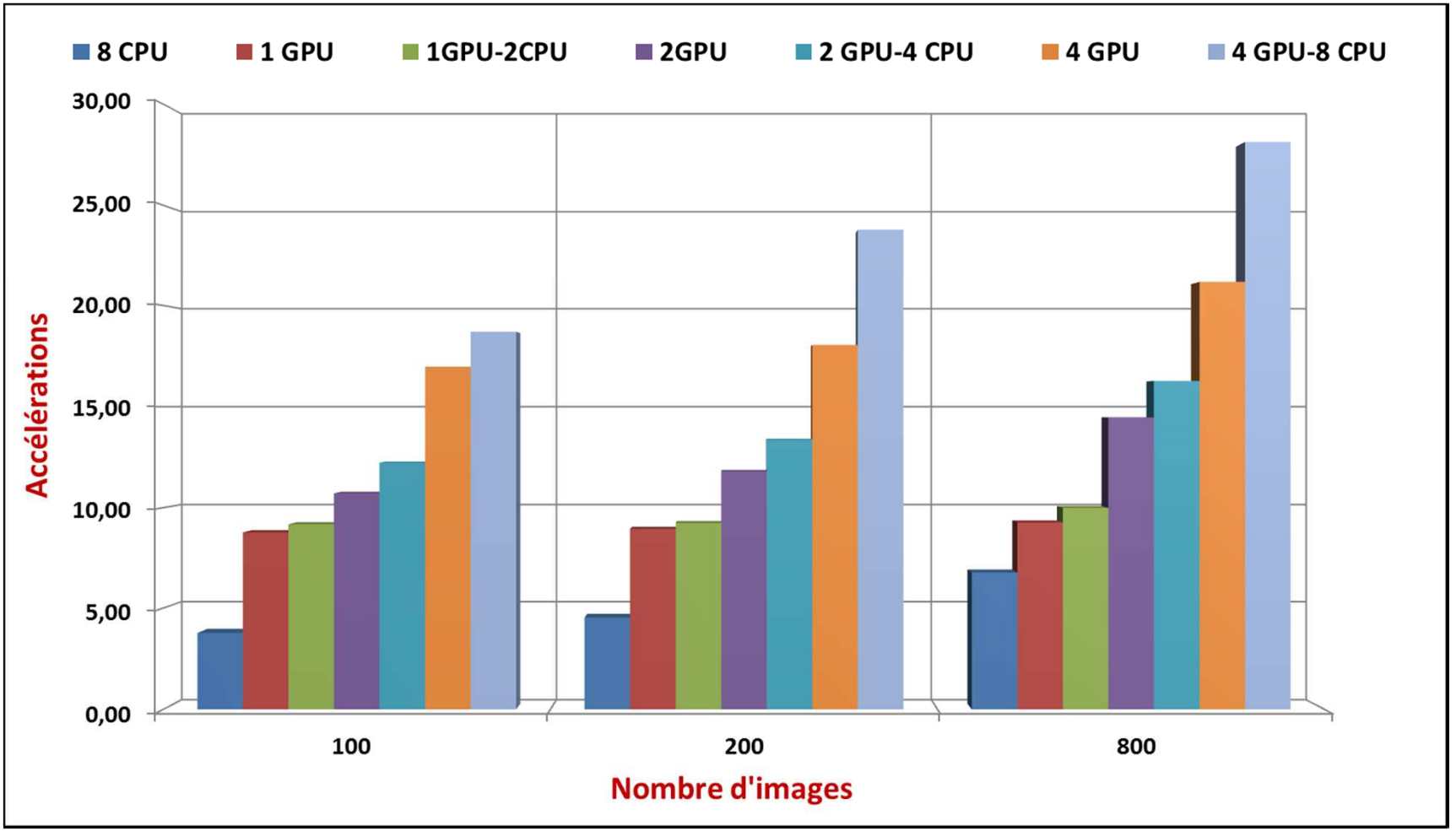


(1) **Détection hétérogène des contours**

(2) **Moments de Hu extraits à partir des contours**

Traitement hétérogène de vidéos multiples : indexation de vidéos

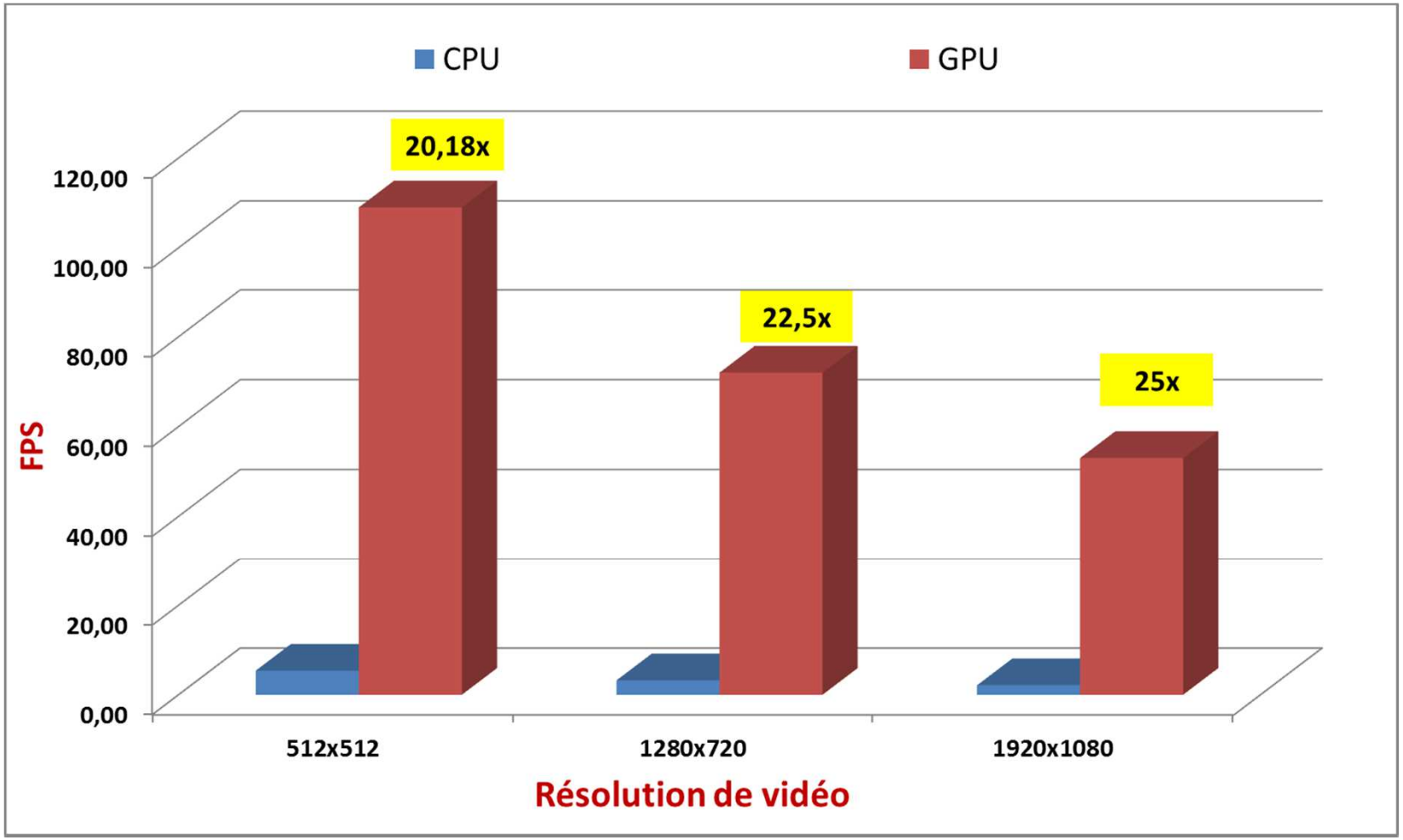
2ème cas : indexation de vidéos



Performances du calcul hétérogène d'extraction de caractéristiques

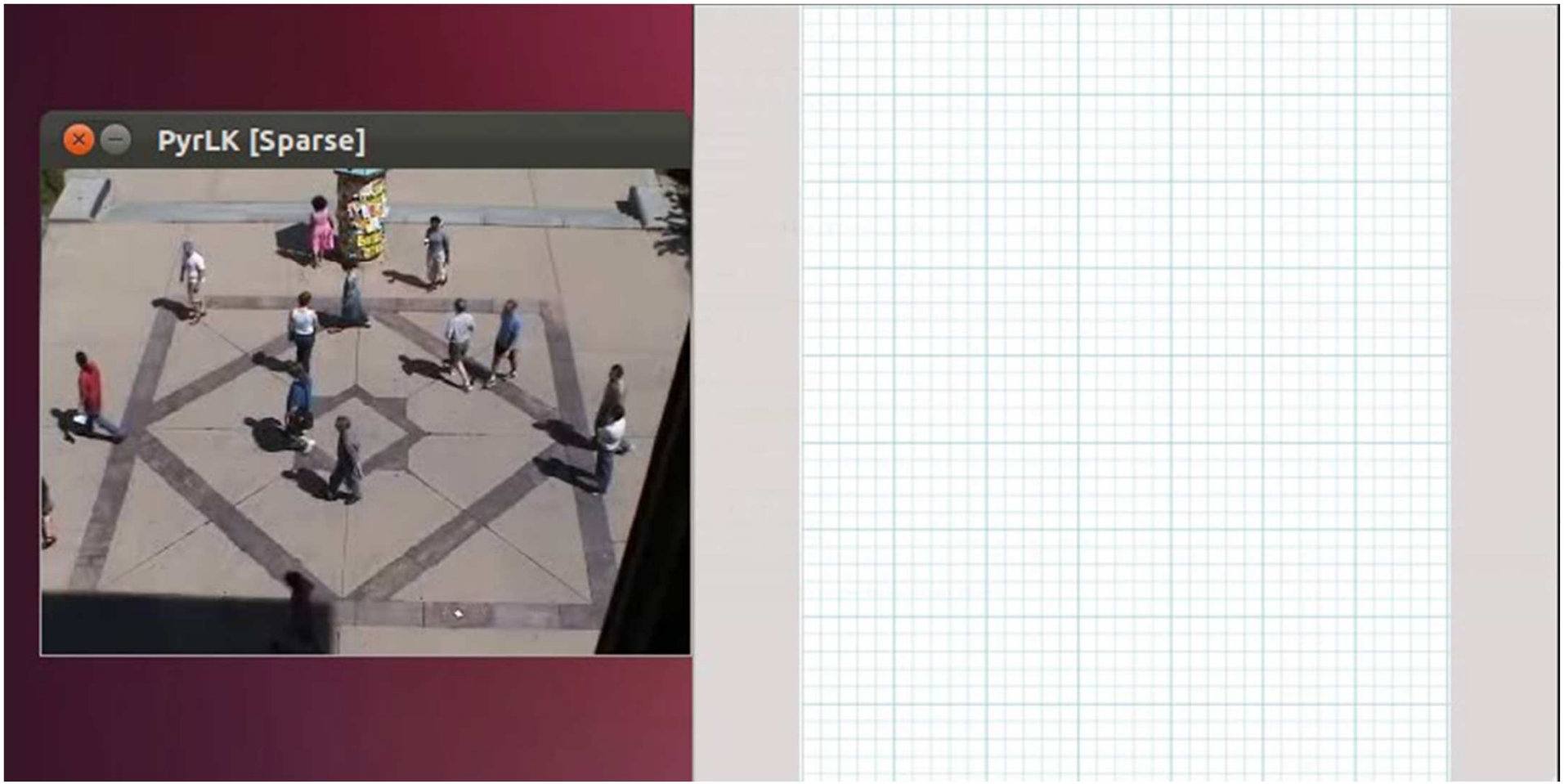
3ème cas: Détection de mouvements avec caméra mobile

3ème cas: Détection de mouvements avec caméra mobile



Performances du traitement GPU appliqué à la détection de mouvements

4ème cas: Détection d'évènement anormaux en temps réel



Détection d'évènement anormaux dans la foule (en temps réel)

Conclusion

- **Traitement parallèle intra et inter-images**
  - Traitement parallèle des pixels sous CUDA et visualisation rapide via OpenGL
  - Exploitation des cœurs CPUs et GPUs multiples
  - Ordonnancement de tâches basé sur l'historique

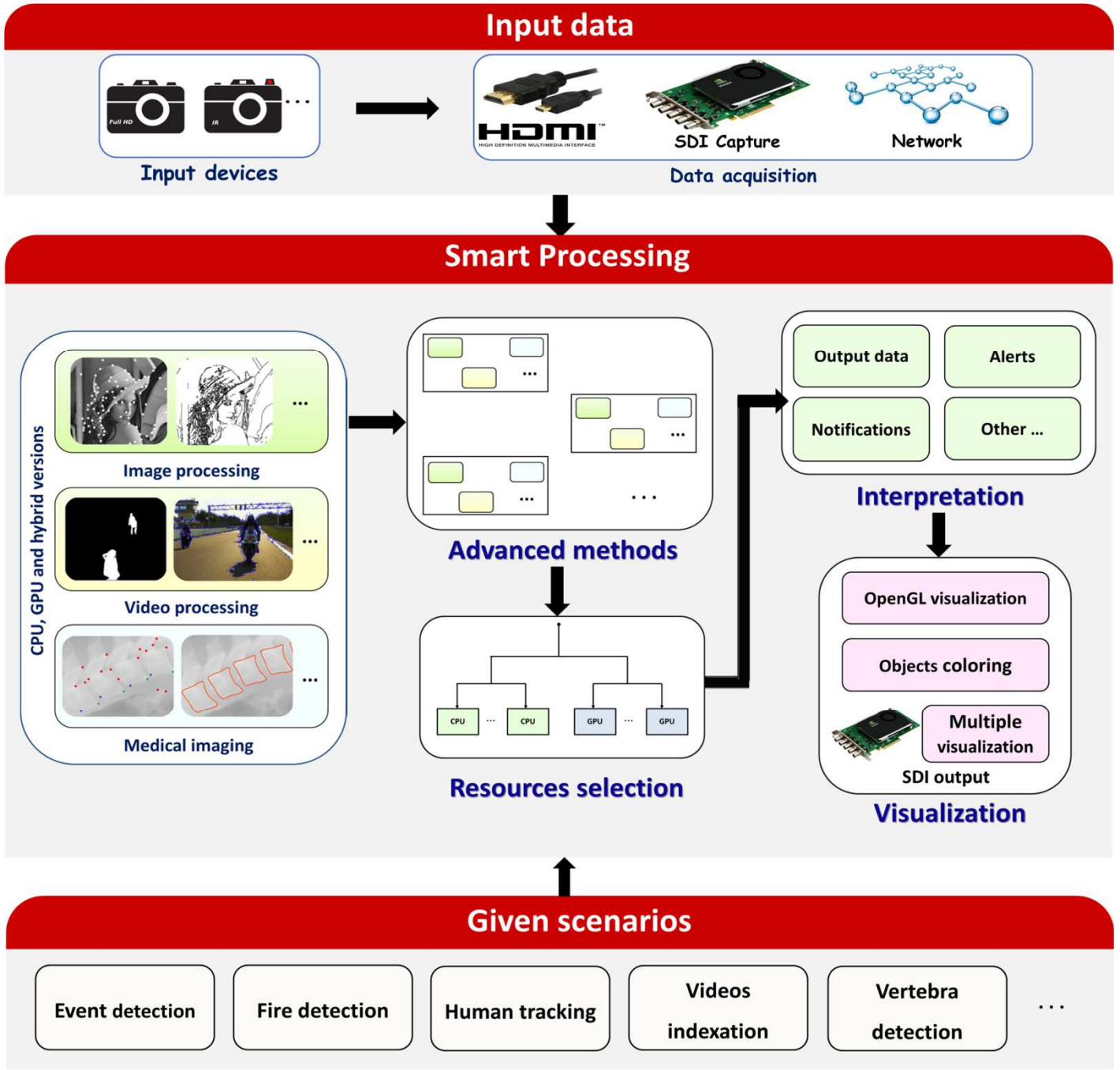
- **Gestion efficace des mémoires sur GPU multiples**
  - Exploitation adaptée des mémoires partagée et de textures
  - Recouvrement des transferts par les kernels d'exécutions: CUDA streams

- **Traitement GPU/Multi-GPU de vidéos Full HD/4K en temps réel**
  - Méthode de détection, analyse et suivi de mouvements sous CUDA
  - Traitements appliqués en temps réel sur des vidéos HD, Full HD et 4K

- **Choix efficace des ressources**
  - Estimation de complexité (facteur de parallélisation, nombre d'opérations, etc.)
  - Choix de ressource (CPU ou GPU) à partir de ce facteur

Perspectives

- Estimation de complexité améliorée: ratio calcul/accès mémoire, graphe de dépendances, etc.
- Ordonnancement de tâches prenant en compte plus de paramètres
- Choix automatique du nombre de ressources à exploiter
- Analyse de consommation d'énergie
- Système intelligent de suivi de mouvements dans différents scénarios


Input data
Full HD
IR
Input devices
HDMI
HIGH DEFINITION MULTIMEDIA INTERFACE
SDI Capture
Network
Data acquisition
Smart Processing
CPU, GPU and hybrid versions
Image processing
Video processing
Medical imaging
Advanced methods
CPU
CPU
GPU
GPU
Resources selection
Output data
Alerts
Notifications
Other ...
Interpretation
OpenGL visualization
Objects coloring
Multiple visualization
SDI output
Visualization
Given scenarios
Event detection
Fire detection
Human tracking
Videos indexation
Vertebra detection
...

Revues internationales

- F. Lecron, **S. A. Mahmoudi**, M. Benjelloun, S. Mahmoudi and P. Manneback **"Heerogenous Computing for Vertebra Detection and Segmentation in X-Ray Images**", *International Journal of Biomedical Imaging : Parallel Computation in Medical Imaging Applications*. Juin 2011.

- **S. A. Mahmoudi**, P. Manneback, C. Augonnet, S. Thibault « Traitement d'Images sur Architectures Parallèles et Hétérogènes »*, Revue des sciences et technologies de l'information.* Article vol :31/8-10 – 2012, pp.1183-1203, doi:10.3166/tsi.31.1183-1203, Octobre 2012

Conférences internationales

- P. D. Possa, **S. A. Mahmoudi**, N. Harb, C. Valderrama " A New Self-Adapting Architecture for Feature Detection ", FPL 2012 : 22nd International Conference on Field Programmable Logic and Applications, Oslo, Novège. Août 2012.

- **S. A. Mahmoudi**, P. Manneback, C. Augonnet, S. Thibault "**Détection optimale des coins et contours dans des bases d'images volumineuses sur architectures multi-cœurs hétérogènes***", 20eme Rencontres Francophones du Parallélisme, RenPar'20*, Saint-Malo, France. Mai 2011.

- **S. A. Mahmoudi**, S. Frémal, M. Bagein, P. Manneback, "**Calcul intensif sur GPU : exemples en traitement d'images, en bio-informatique et en télécommunication***", CIAE 2011 : Colloque d'Informatique, Automatique et Electronique*, Casablanca, Maroc. Mars 2011.

- **S. A. Mahmoudi**, F. Lecron, P. Manneback, M. Benjelloun, S. Mahmoudi, "**GPU-Based Segmentation of Cervical Vertebra in X-Ray Images**", *Workshop HPCCE. IEEE International Conference on Cluster Computing*, Crete, Greece. Septembre 2010.

- **S. A. Mahmoudi**, P. Manneback, "**Parallel Image Processing with CUDA and OpenGL***", Network for High-Performance Computing on Complex Environments.* Lisbon, Portugal. COST ACTION IC 805, WG Meeting. October 2009.

- **S. A. Mahmoudi**, P. Manneback, "**Traitements d'images sur GPU sous CUDA et OpenGL : application a l'imagerie médicale***", Journées CIGIL : Calcul Intensif et Grilles Informatiques a Lille*. Lille, France. December 2009.

- **S. A. Mahmoudi**, Pierre Manneback, « Traitement d'objets multimédias sur gpu", Seconde journée scientifique du pôle hainuyer. Belgique, Mai 2010.

Rapports techniques

- S. Dupont, C. Frisson, **S. A. Mahmoudi**, X. Siebert, J. Urbain, T. Ravet, "**MediaBlender : Interactive Multimedia Segmentation and Annotation**", *QPSR of the numediart research program*, volume 3, December 2010.

- M. Mancas, R. B. Madkhour, **S. A. Mahmoudi**, T. Ravet, "**VirTrack: Tracking for Virtual Studios***", QPSR of the numediart research program*, volume 3, N° 2010

- M. Mancas, J. Tilmanne, R. Chessini, S. Hidot, C. Machy, **S. A. Mahmoudi**, T. Ravet, "**MATRIX : Natural Interaction Between Real and Virtual Worlds**", *QPRS of the numediart research program*, vol. 1, N° 5, January 2009.

- M. Mancas, M. Bagein, N. Guichard, S. Hidot, C. Machy, **S. A. Mahmoudi**, X. Siebert, "**AVS : Augmented Virtual Studio**", *QPSR of the numediart,* 2008

MERCI