UMONS
Université de Mons

INFORMATIQUE PARALLÈLE ET DISTRIBUÉE

CHAPTER 7 : MULTI-CPU/MULTI-GPU PROCESSING

APPLICATION FOR IMAGE AND VIDEO PROCESSING

Sidi Ahmed Mahmoudi

POLYTECH MONS

sidi.mahmoudi@umons.ac.be

13 December 2017

PLAN

PLAN

Introduction

Moore's Law : 1968

The number of transistors in an integrated circuit doubles approximately every two years

✓ Law well verified :the CPU power has doubled every 18 months

✗ CPU power capped to 4 GHZ : the law is no more verified

Introduction

- Multiplication of computing units in CPU : **Multi-CPU, Multi-core, GPU**
- GPU : initially used in 3D and video games.

CPU Intel Core i7 3770K (2012)

4 cores/8 threads

320 Euros

GPU GeForce GTX 580 (2011)

512 CUDA cores

350 Euros

Birth of GPGPU : General Purpose Graphic Processing Unit

PLAN

Introduction

I. GPU Presentation

II. GPU Programming

III. Exploitation of Multi-CPU-GPU Architectures

IV. Application for Multimedia processing

1. Multi-CPU/Multi-GPU based image processing
2. Multi-GPU based video processing in Real Time

V. Multi-CPU/Multi-GPU based framework for HD image and video processing

VI. Experimental Results

Conclusion

GPU Presentation

- Graphic processing units

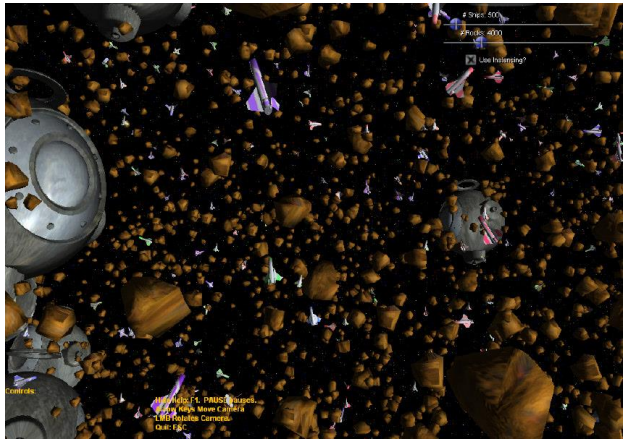
NVIDIA

ATI

- Integrated within graphic cards

GeForce GTX 580

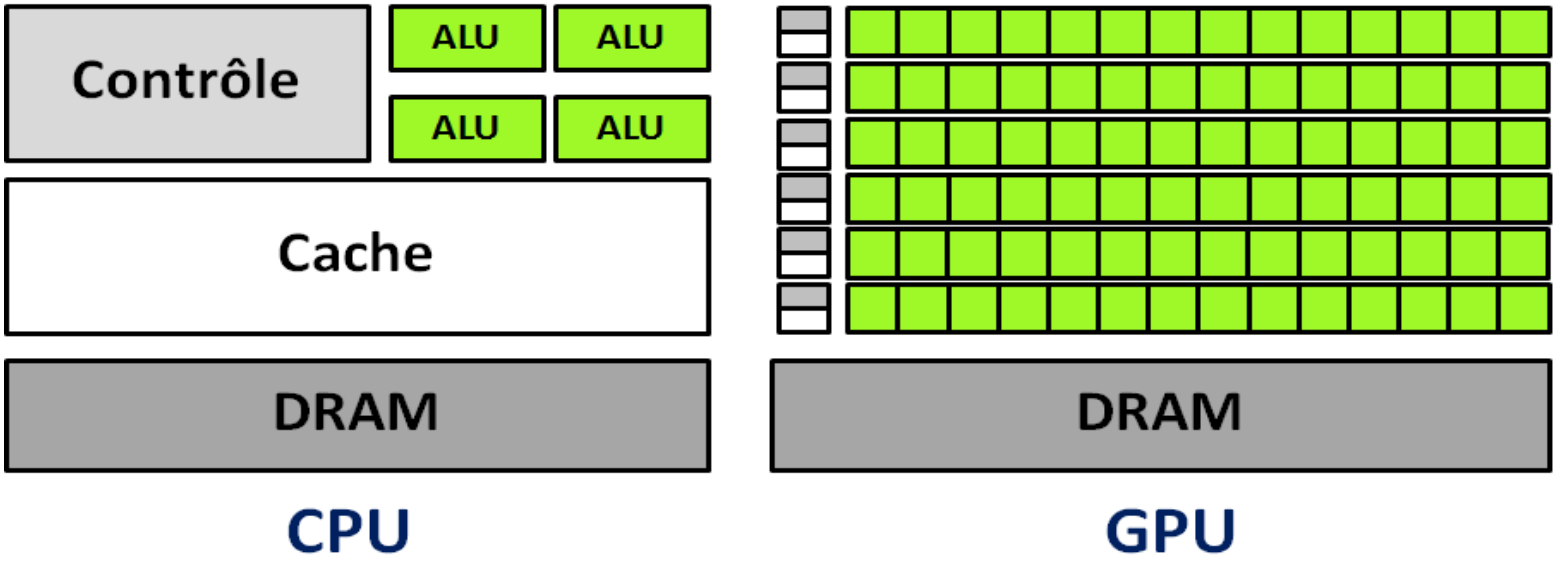
Radeon HD4870



- Initially used for 3D rendering and video games

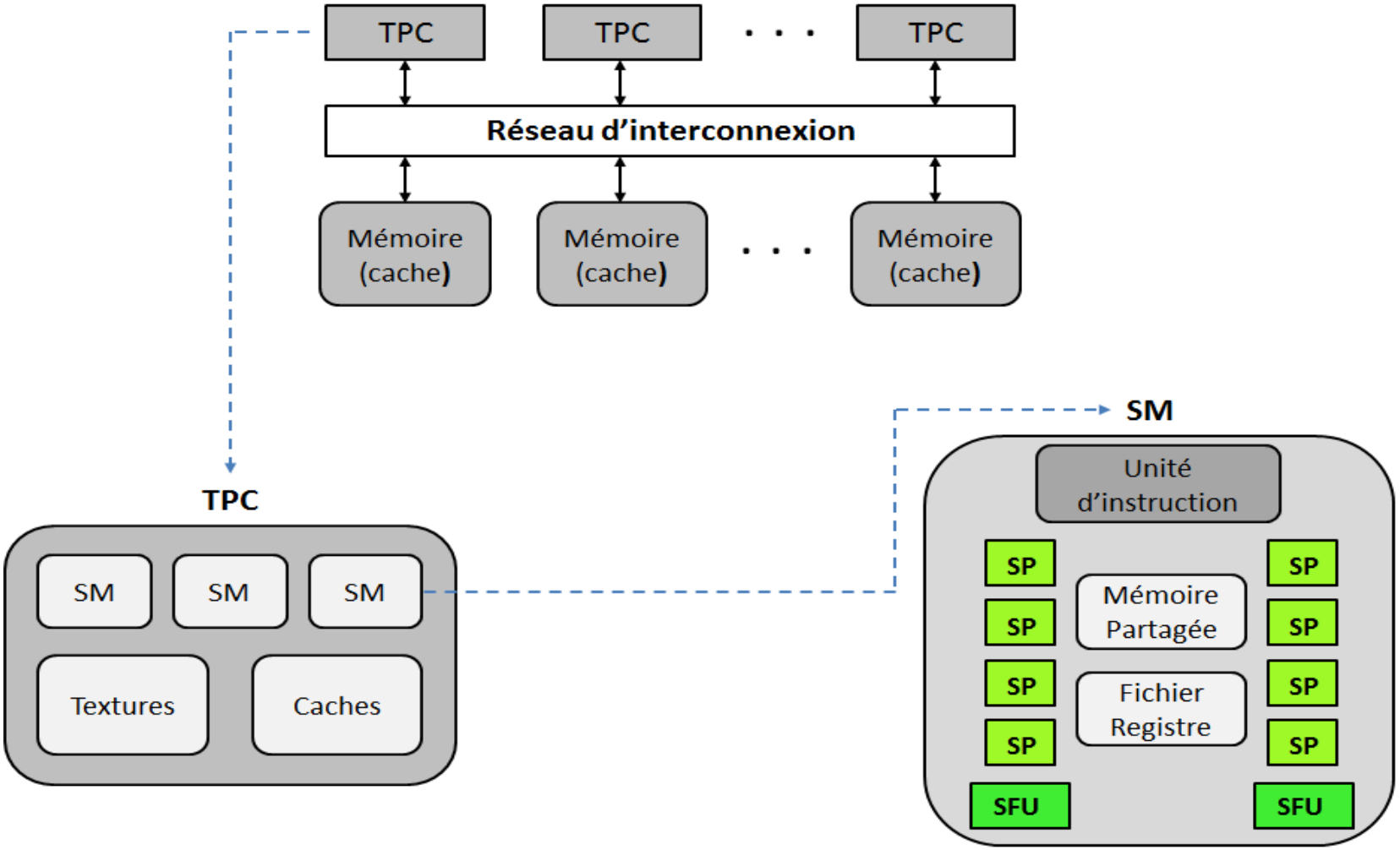
GPU Presentation

- GPU specialized for massively parallel treatments
- More transistors dedicated for computing
- Less number of control and cache units.

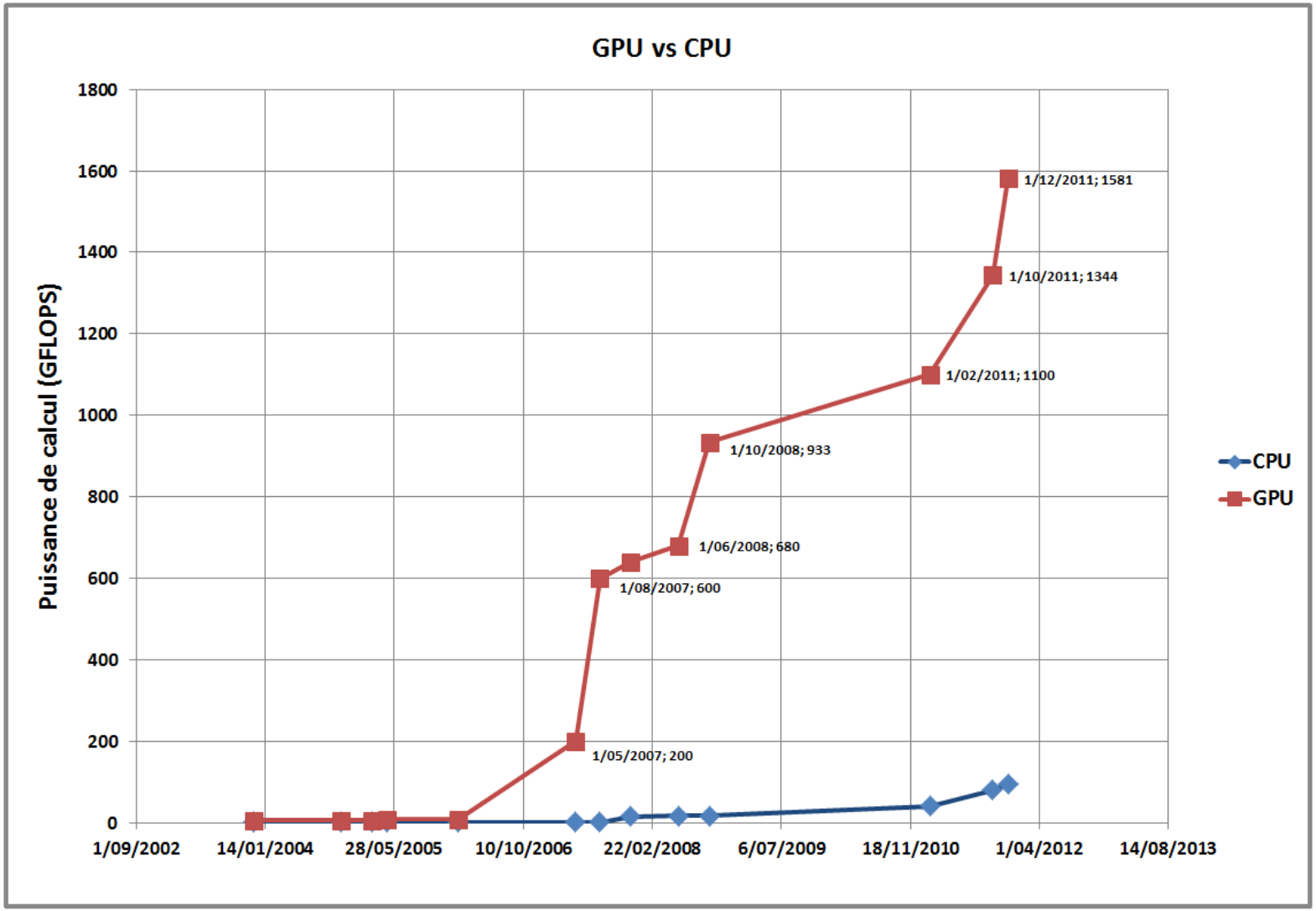


- **GPU:** Multiplication of computing units

GPU Presentation

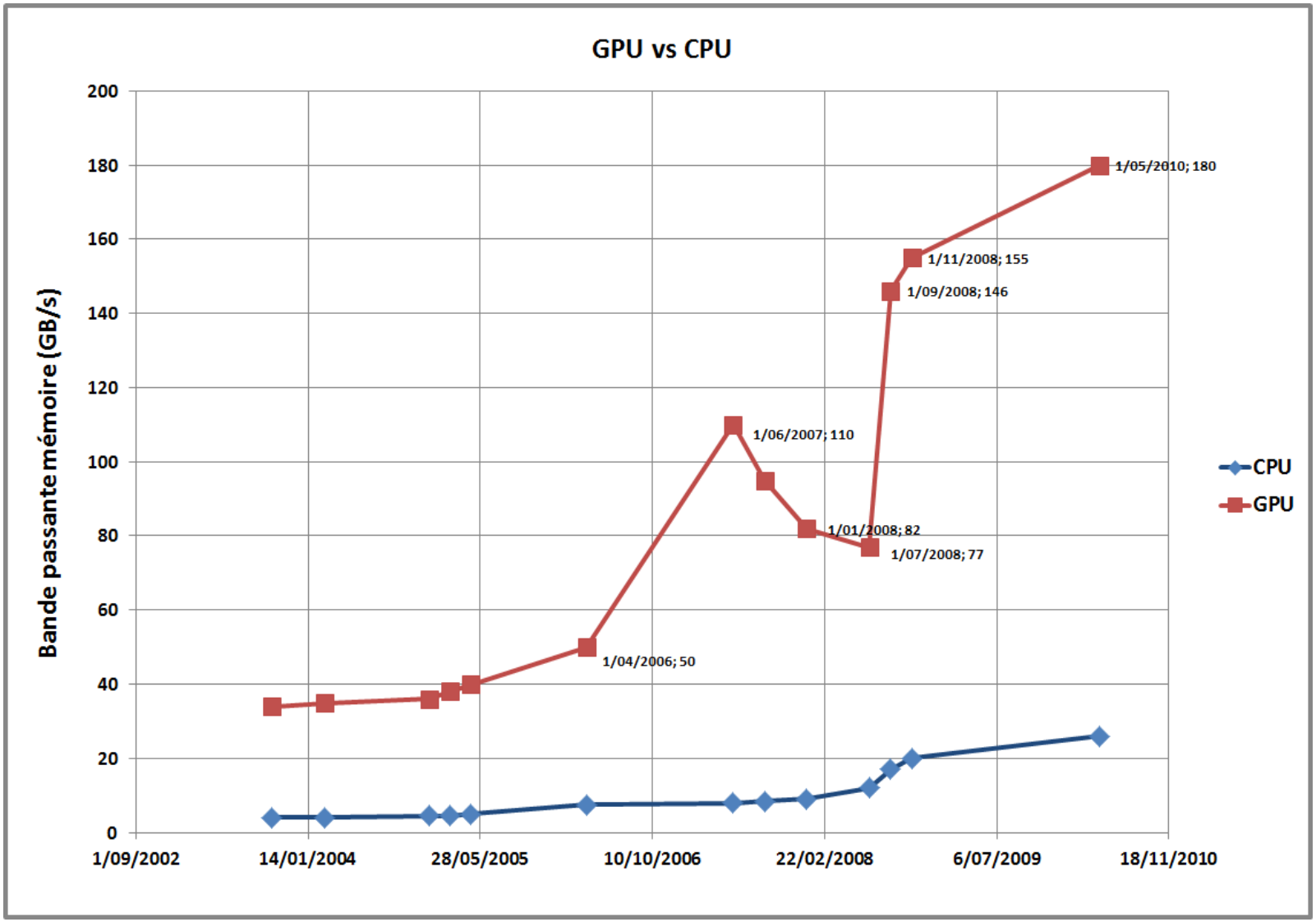


GPU Presentation



Power evolution of CPU Intel vs. GPU Nvidia

GPU Presentation



Bandwidth evolution of CPU Intel vs. GPU Nvidia

GPU Presentation

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325

TOP 500. Current List, June 2014. http://www.top500.org

PLAN

Introduction

I. GPU Presentation

II. GPU Programming

III. Exploitation of Multi-CPU-GPU Architectures

IV. Application for Multimedia processing

1. Multi-CPU/Multi-GPU based image processing
2. Multi-GPU based video processing in Real Time

V. Multi-CPU/Multi-GPU based framework for HD image and video processing

VI. Experimental Results

Conclusion

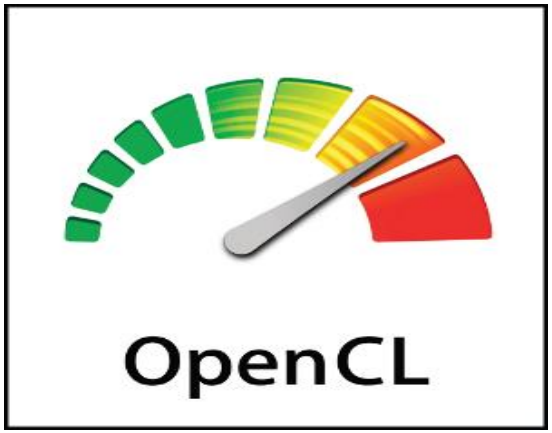
GPU Programming

- **Brook GPU :** since 2004
- **ATI Stream :** for ATI graphic cards
- **DirectX 11, OpenGL :** GPGPU shaders
- **OpenCL :** compatible with both NVIDIA and ATI

- **CUDA :** for NVIDIA cards only

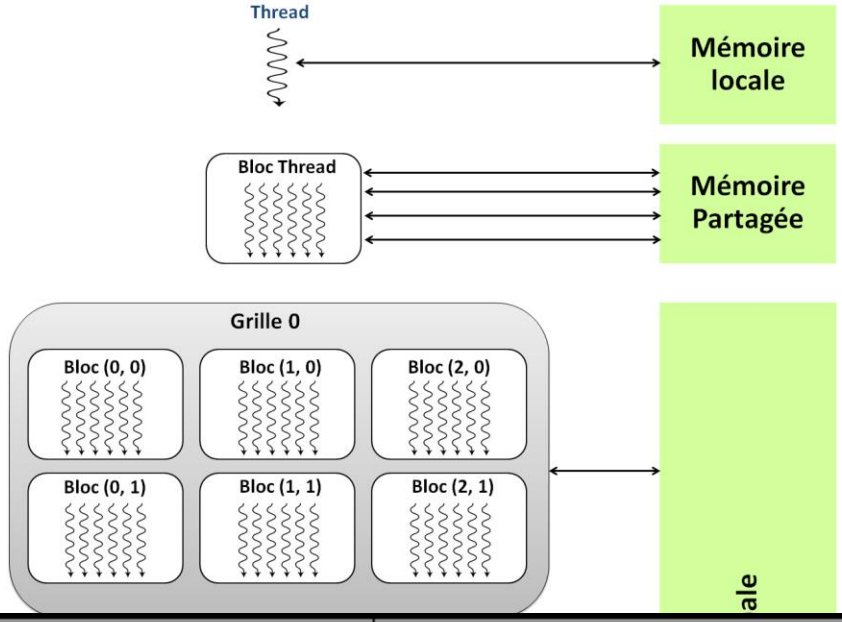
OpenCL

- Launched by Apple in order to unify the use of multi-cores and GPUs
- Joint by AMD/ATI and NVIDIA
- First version in 2009
- Compatible with both NVIDIA and Ati cards
- Newer than CUDA : less powerful !!
- New GPU data types (vector, image, etc.).



CUDA

- Compatible with GPU NVIDIA.
- Syntax similar to C.
- Grid: set of blocs.
- Bloc: set of threads.



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

- Example of vectors addition :

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

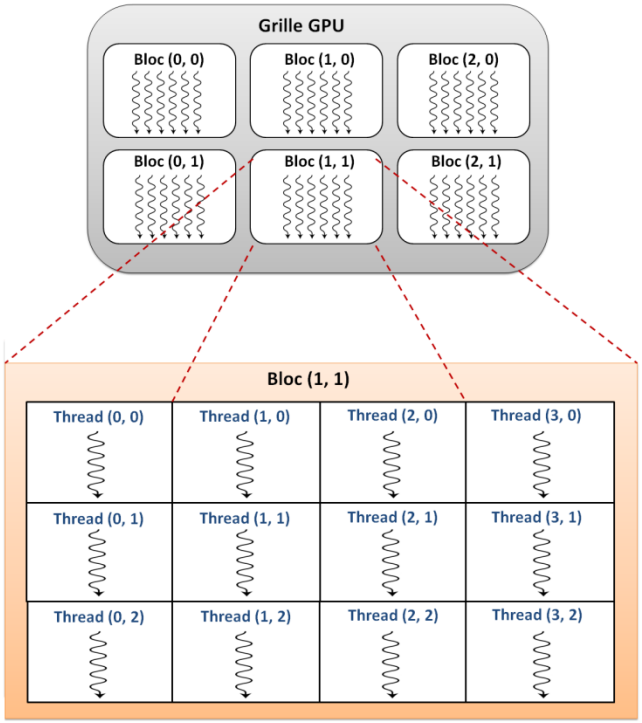
```
__global__ void vecAdd(float* A, float* B, float* C)
{
        int i = threadIdx.x;
        C[i] = A[i] + B[i];
}

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

Kernel

Call of 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 Bloc of N threads**

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

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

CUDA

A CUDA program consists of five steps :

1. Memory allocation on GPU
2. Data transfer from CPU to GPU memory
3. Definition of the number of CUDA threads
4. Launching of CUDA functions in parallel
5. Data (result) transfer from GPU to CPU memory

CUDA

Example of matrix addition: Memory allocation

- Memory allocation on GPU
- The allocated memory will receive data from central (CPU) memory.

```
float *Ad, *Bd, *Cd;
const int size = N*N*sizeof(float);          // matrix size

cudaMalloc((void **)&A_d, size);             // Allocate matrix A_d

cudaMalloc((void **)&B_d, size);             // Allocate matrix B_d

cudaMalloc((void **)&C_d, size);             // Allocate matrix C_d
```

CUDA

Example of matrix addition : data transfer (CPU Mem to GPU Mem)

- Data transfer from CPU memory to GPU memory
- Matrix initialized on central memory
- Precise the type of transfer : **HostToDevice**

```
cudaMemcpy(A_d, A, size, cudaMemcpyHostToDevice);
cudaMemcpy(B_d, B, size, cudaMemcpyHostToDevice);
   // A and B represent the matrix initialized on CPU memory
```

CUDA

Example of matrix addition : define the number of threads

- Define the number of GPU threads :
  - Define the grid size
  - Define the bloc size
- The threads number depends from the size of data to treat

```
dim3 dimBlock( blocksize, blocksize );          //  Bloc size

dim3 dimGrid( N/dimBlock.x, N/dimBlock.y);      //  Grid size
```

CUDA

Example of matrix addition : launching of CUDA kernels

- Define the CUDA function (kernel)
- Call and execute the CUDA kernel
- Specify the threads number (already defined)

```
__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

Example of matrix addition: Results transfer (GPU Mem to CPU Mem)

- Transfer of results from GPU to CPU memory
- Precise the type of transfer : **DeviceToHost**
- Release the allocated memory on GPU

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

cudaFree( A_d );

cudaFree( B_d );

cudaFree( C_d );

// Results wil be savec on CPU memory « C ».
```

PLAN

Introduction

I. GPU Presentation

II. GPU Programming

III. **Exploitation of Multi-CPU-GPU Architectures**

IV. Application for Multimedia processing

1. Multi-CPU/Multi-GPU based image processing
2. Multi-GPU based video processing in Real Time

V. Multi-CPU/Multi-GPU based framework for HD image and video processing

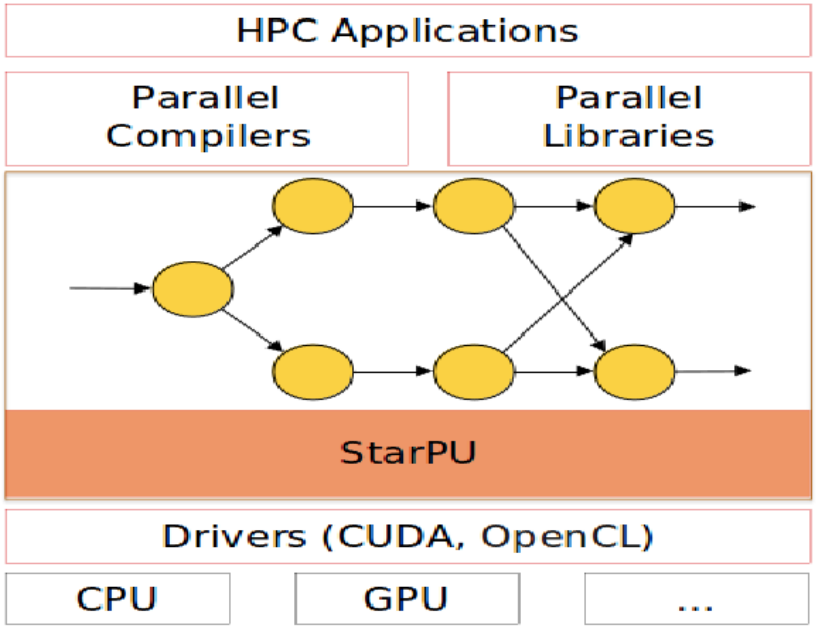
VI. Experimental Results

Conclusion

Exploitation of Multi-CPU/Multi-GPU Architectures

StarPU

- Developed in LABRI laboratory. Bordeaux, France
- Simultaneous exploitation of multiple CPUs and GPUs
- Exploit functions implemented with C, CUDA or OpenCL
- Efficient scheduling strategies



Survey of StarPU [1]

Exploitation of Multi-CPU/Multi-GPU Architectures

StarSS

- Developed at the University of Catalonia (Barcelona)
- Flexible programming model for multicore
- Based essentially on :

  - **CellSs:** programming Cell processors
  - **SPMSs:** programming SMPs processors
  - **ClearSpeedSs:** programming Clear Speed processors
  - **GPUSs:** programming Multi-GPUs
  - **ClusterSs:** programming of clusters
  - **GridSs:** programming resources in grids

Exploitation des architectures hétérogènes

OpenACC

- API for high level programming
- Supported by CAPS enterprise, CRAY Inc, The Portland Group Inc (PGI) and NVIDIA
- A collection of compilation directives
- Takes into account the tasks of initialization, launching and stopping of accelerators
- Provides information for compilers
- Equivalent to Open MP for CPU programming

PLAN

Introduction

I. GPU Presentation

II. GPU Programming

III. Exploitation of Multi-CPU-GPU Architectures

IV. **Application for Multimedia processing**

1. Multi-CPU/Multi-GPU based image processing
2. Multi-GPU based video processing in Real Time

V. Multi-CPU/Multi-GPU based framework for HD image and video processing

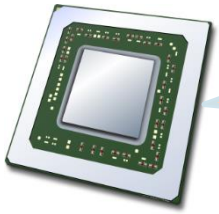
VI. Experimental Results

Conclusion

Intensive treatments

Large volumes

HD, Full HD, 4K, etc.



Multi-CPU

Computing time ?

Real time ?

Cost ?

Image processing

Video processing

Medical imaging

Multimedia processing

GPU or Multi-GPU

Acceleration

Parallel computing

Hybrid computing

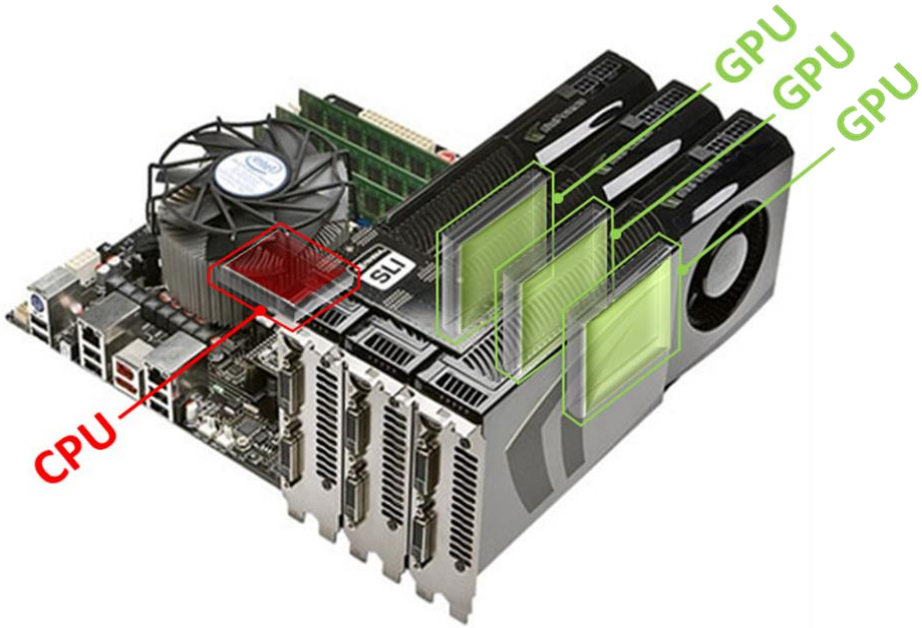
CPU or/and GPU ?

Application for Multimedia processing

Hardware

- High computing power of GPUs.
- Heterogeneous architectures (Multi-CPU/Multi-GPU)

GPU



Application for Multimedia processing

Hardware

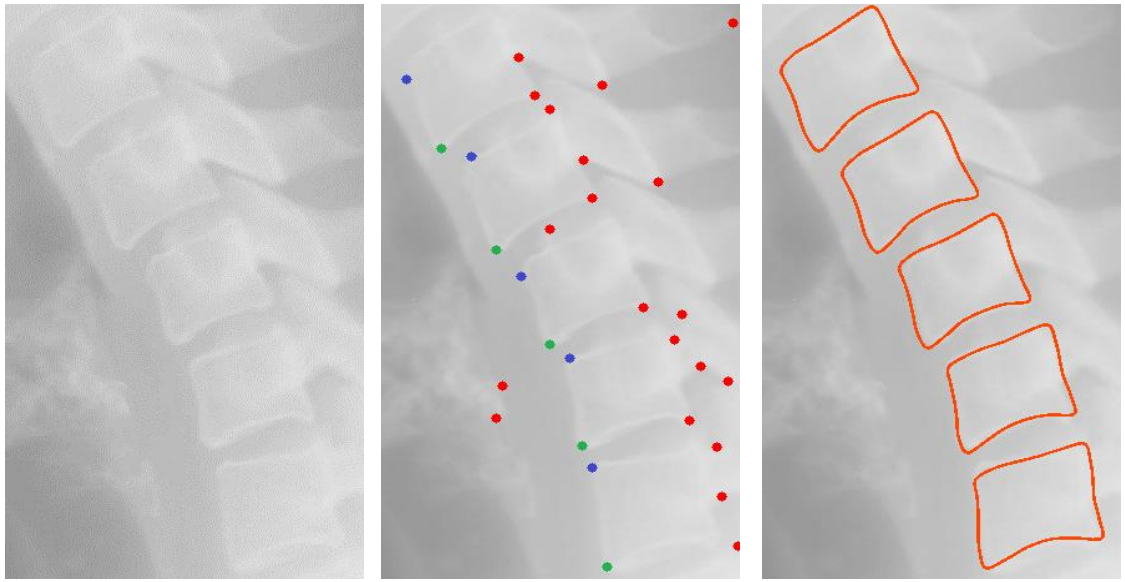
- High computing power of GPUs.
- Heterogeneous architectures (Multi-CPU/Multi-GPU)

Applications

- Intensive processing of images and videos
- intensity : HD, Full HD or 4K of images videos (real time)

200 images

05 minutes



Vertebra detection [Lecron2011]

Application for Multimedia processing

Hardware

- High computing power of GPUs.
- Heterogeneous architectures (Multi-CPU/Multi-GPU)

Applications

- Intensive processing of videos: motion tracking and recognition.
- intensity : HD, Full HD or 4K videos to treat in real time

2000 images

10 minutes

Images indexation :MediaCycle [Siebert2009]

Application for Multimedia processing

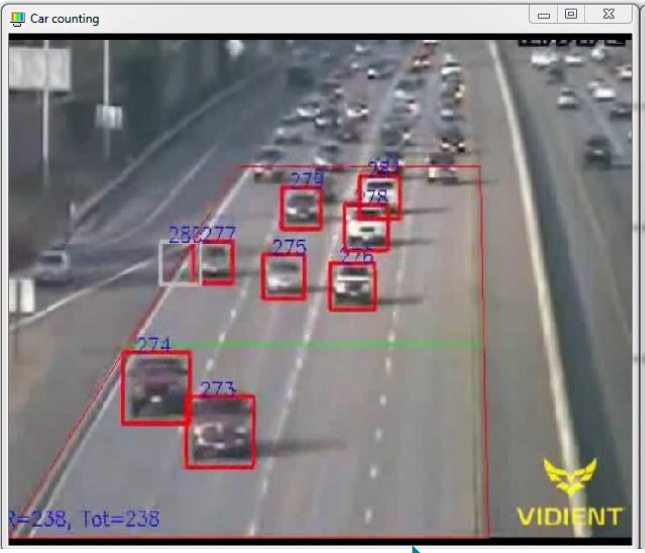
Hardware

- High computing power of GPUs.
- Heterogeneous architectures (Multi-CPU/Multi-GPU)

Applications

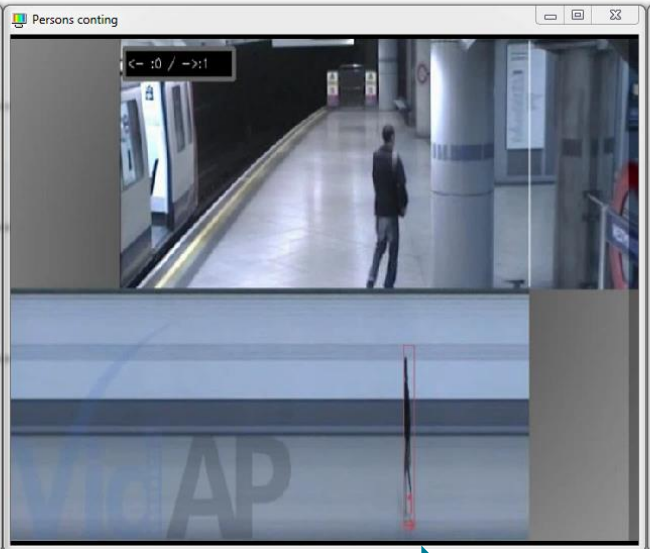
- Intensive processing of videos: motion tracking and recognition.
- intensity : HD, Full HD or 4K videos to treat in real time

Car counting



Vidéo : 720x640 ➡ 19 FPS

Persons conting



Vidéo : 640x376 ➡ 20 FPS

High density crowd tracking

Vidéo : 720x640 ➡ 8 FPS

Application for Multimedia processing

Hardware

- High computing power of GPUs.
- Heterogeneous architectures (Multi-CPU/Multi-GPU)

Applications

- Intensive processing of videos: motion tracking and recognition.
- intensity : HD, Full HD or 4K videos to treat in real time

Constraints

- Transfer times between CPU and GPU memories.
- Efficient distribution of tasks between multiple GPUs

Objectives

- Efficient management of memories: significant reduction of transfer times
- Efficient image and video processing on GPU and Multi-GPU platforms

PLAN

Introduction

I. GPU Presentation

II. GPU Programming

III. Exploitation of Multi-CPU-GPU Architectures

IV. **Application for Multimedia processing**

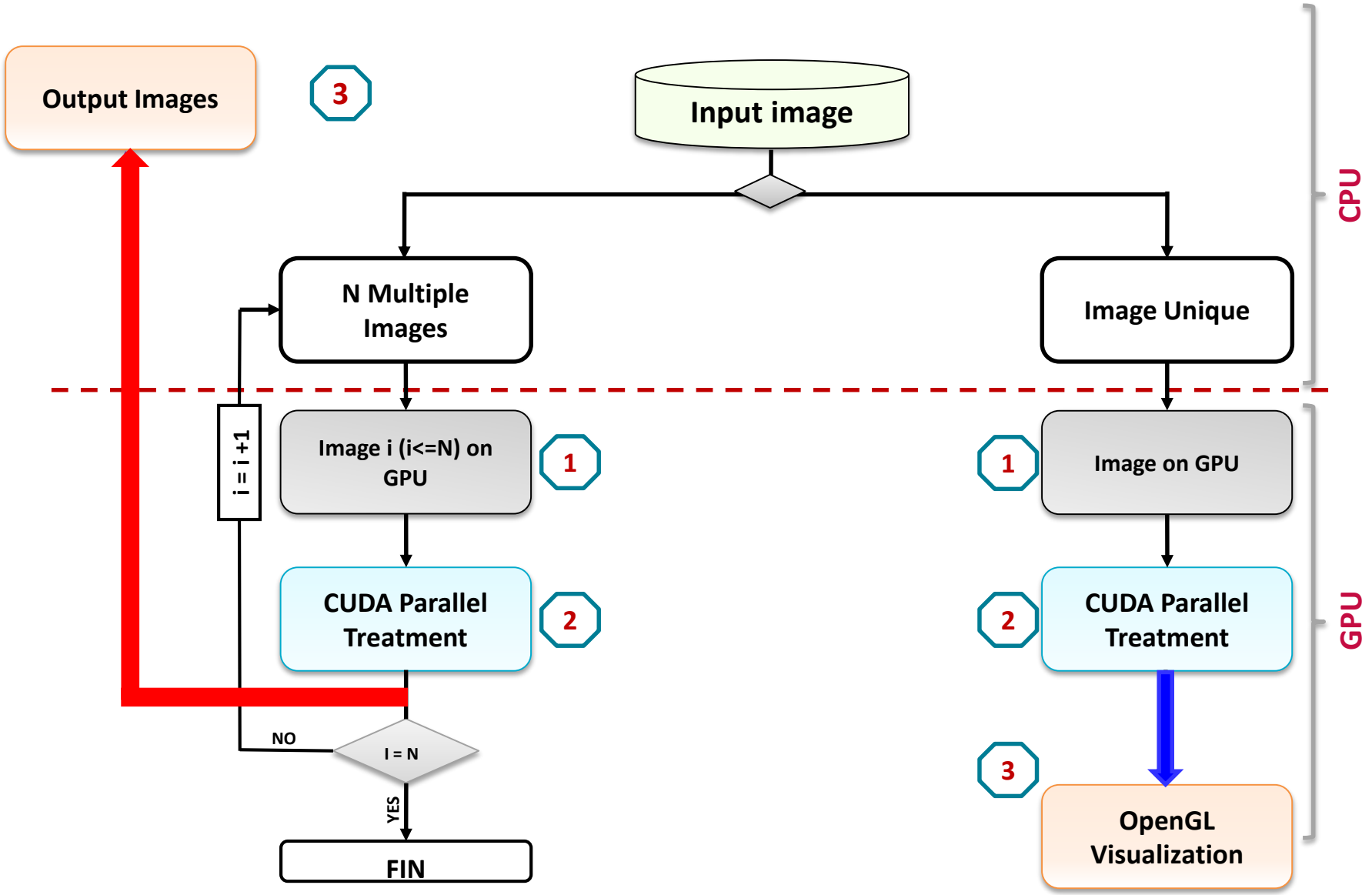
1. **Multi-CPU/Multi-GPU based image processing**
2. Multi-GPU based video processing in Real Time

V. Multi-CPU/Multi-GPU based framework for HD image and video processing

VI. Experimental Results

Conclusion

Scheme development : image processing



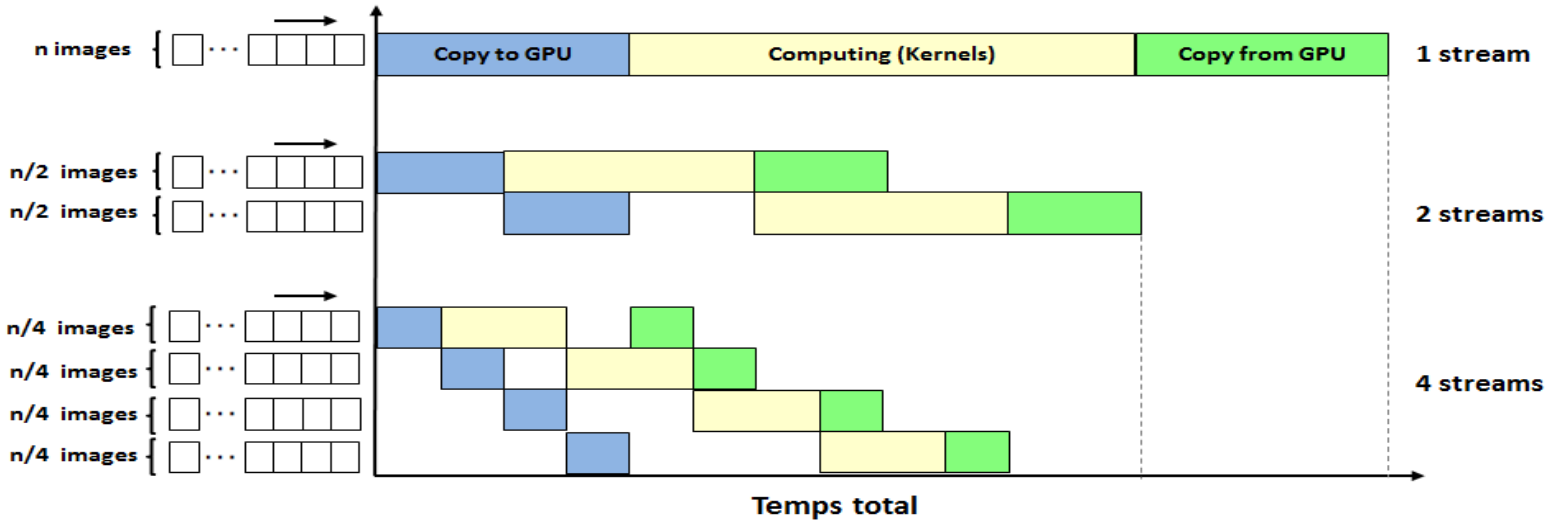
GPU optimizations

Single image processing

- Texture Memory: fast access to pixel values
- Shared Memory : fast access to neighbors' values

Multiple image processing

- Texture et shared memories
- Overlapping of data transfers by kernels executions on GPU : CUDA streams



Implemented algorithms on GPU

Classic image processing algorithms :

- Geometrical transformations:
  - rotation, translation, homography
- Parallel treatments between image pixels
- GPU acceleration : from **10x** to **100x**

Homography on GPU

Points of interest detection on GPU:

- Preliminary step of several computer vision methods
- GPU implementation based on Harris technic
- Efficiency: invariant to rotation, brightness, etc.

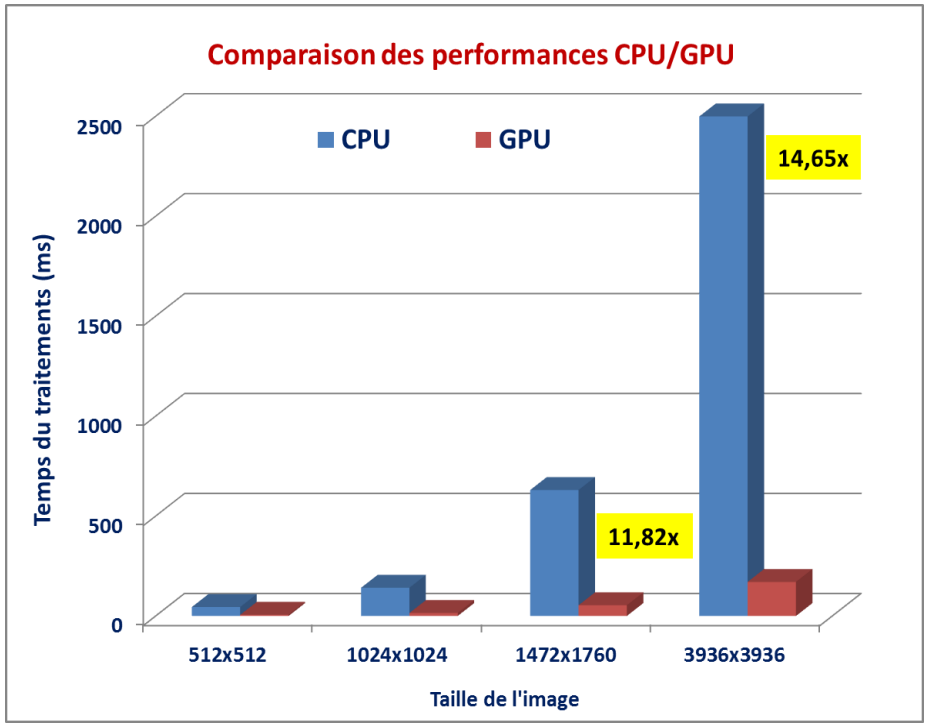
Corners detection on GPU

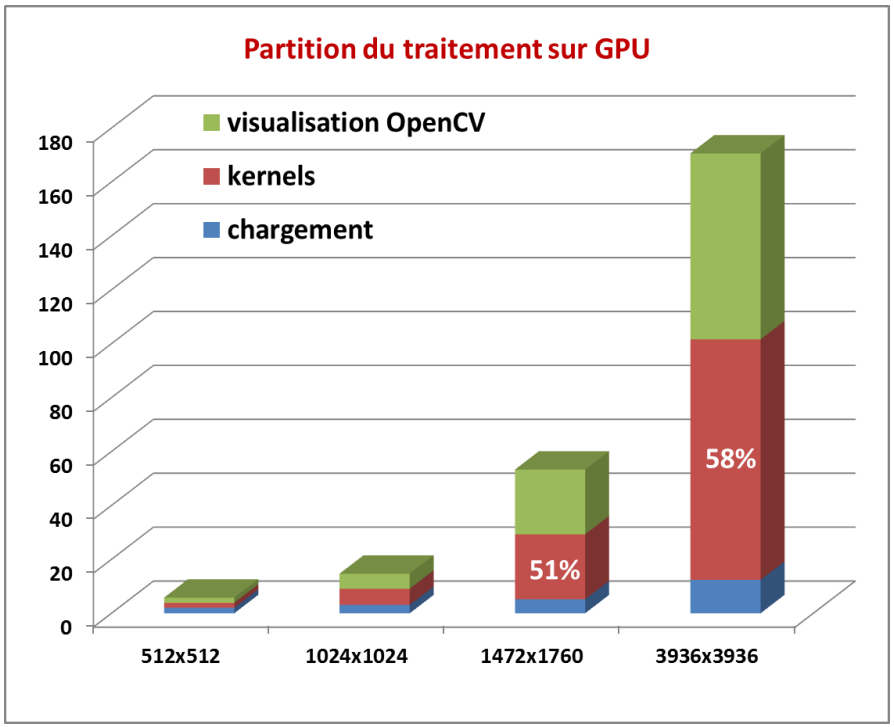
Contours detection on GPU:

- GPU implementation based on Deriche-Canny approach
- Efficiency: noise robustness, reduced number of operations
- Good quality of detected contours

Contours detection on GPU

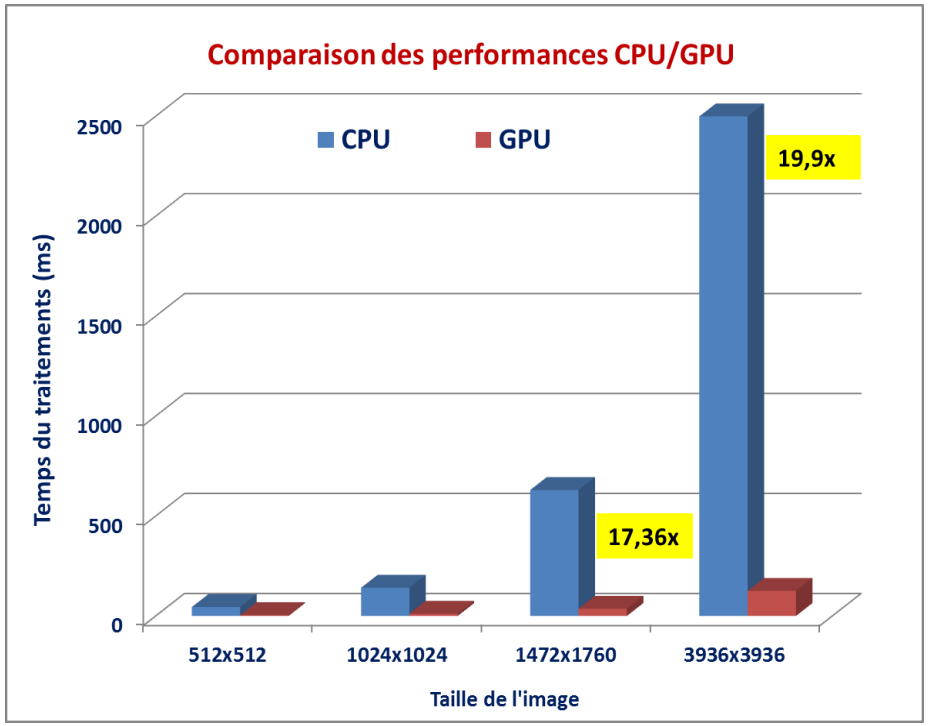
Results : Single image processing on GPU

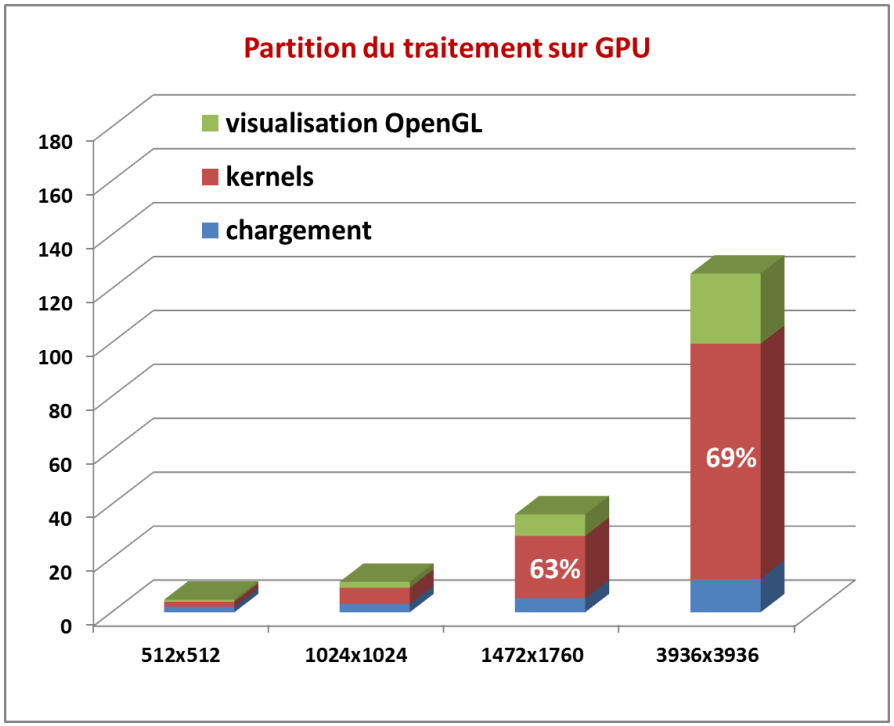




Edges and corners detection on GPU : **Global Memory**

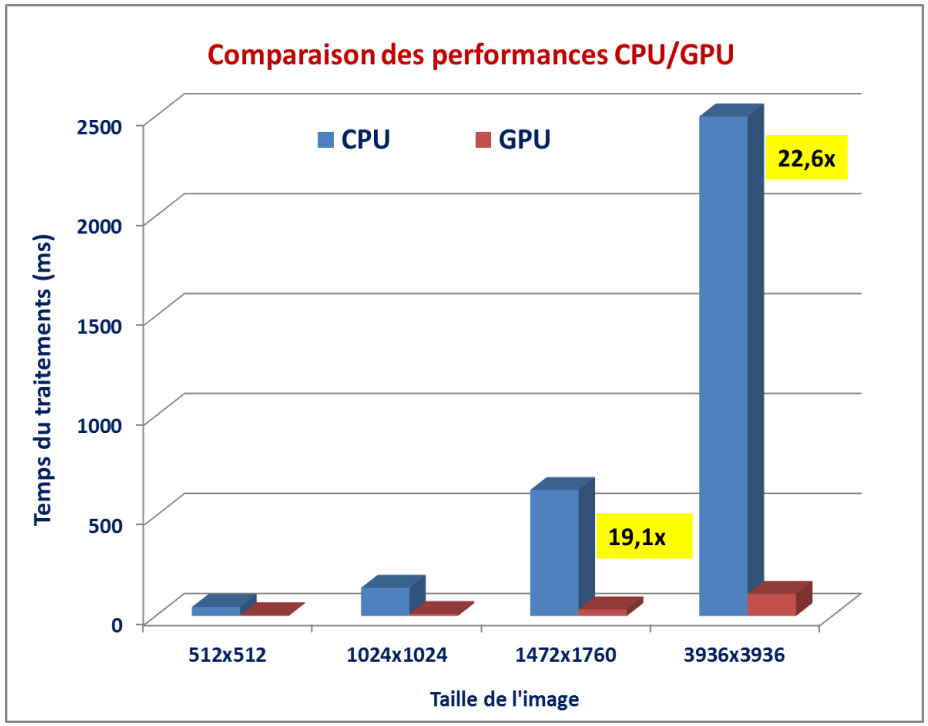
Results : Single image processing on GPU

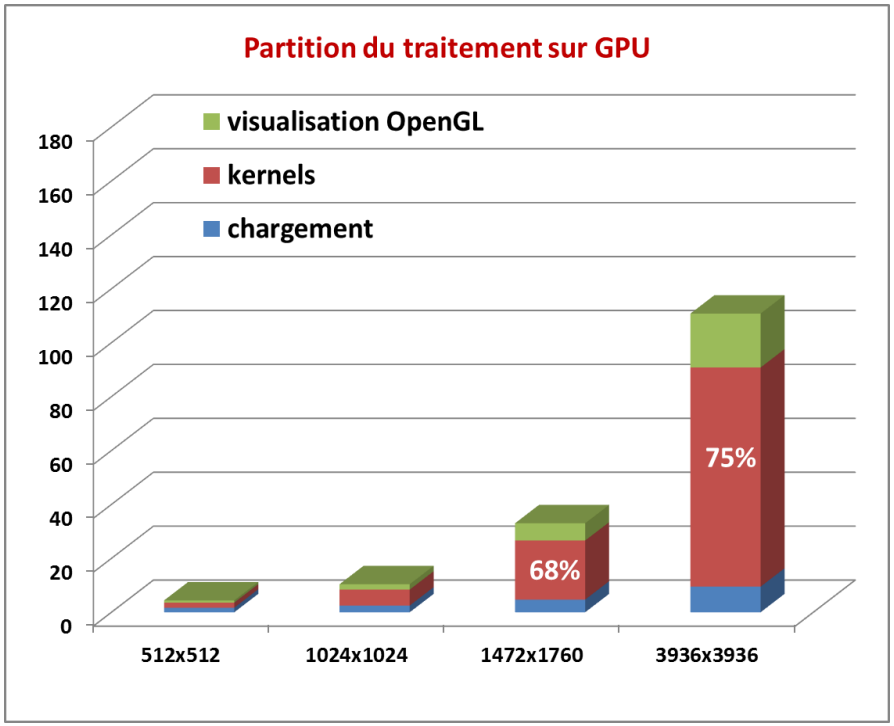




Edges and corners detection on GPU : **OpenGL Visualization**

Results : Single image processing on GPU

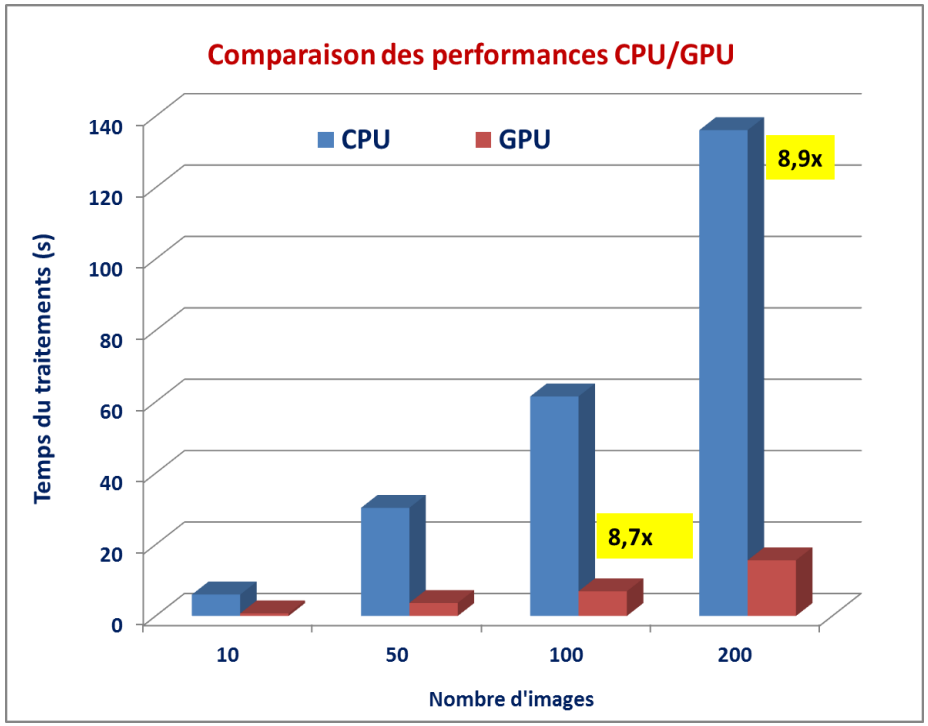


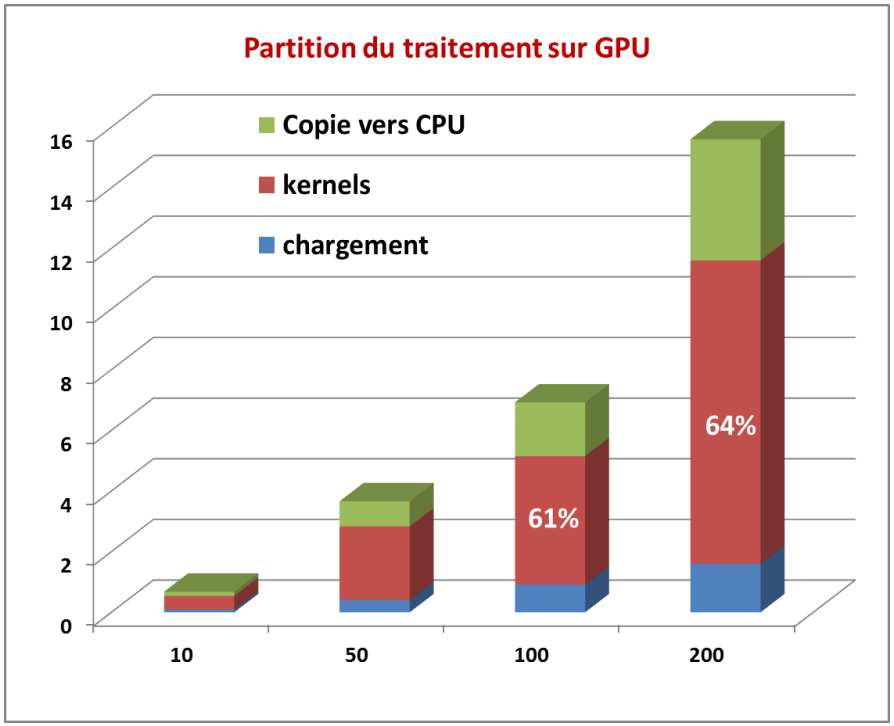


Edges and corners detection on GPU : **Texture & Shared memories**

Results : Multiple images processing on GPU

Images resolution : 2048x2048

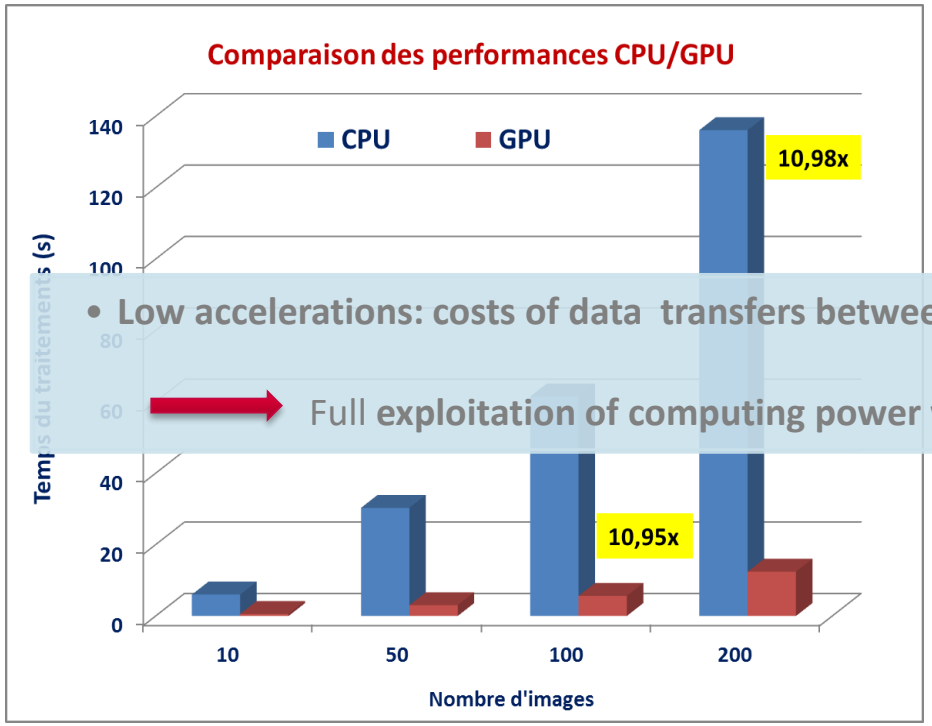


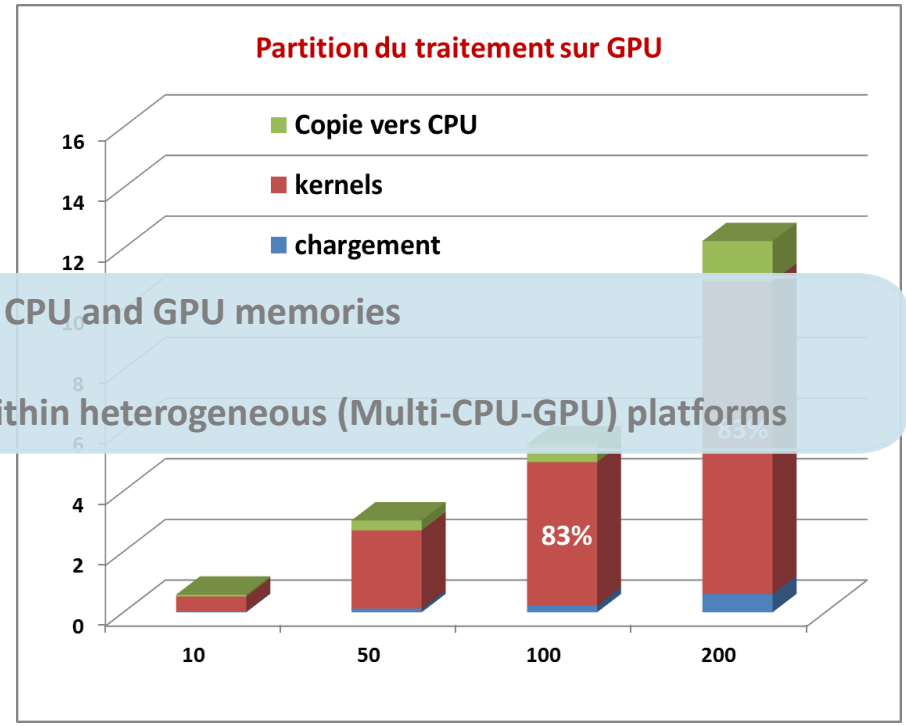


Edges and corners detection on GPU : **Texture & Shared memories**

Results : Multiple images processing on GPU

Images resolution : 2048x2048





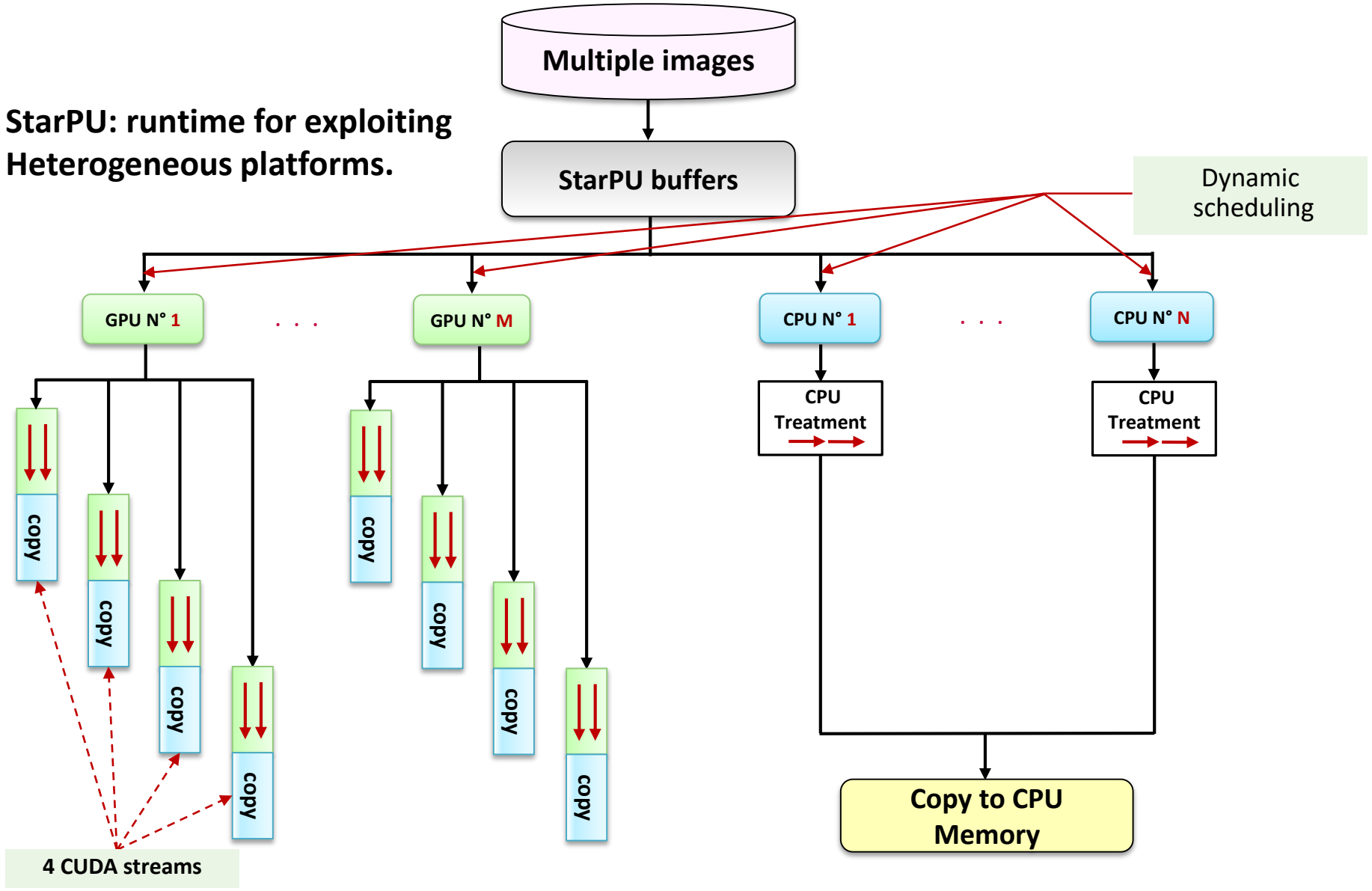
- Low accelerations: costs of data transfers between CPU and GPU memories

→ Full exploitation of computing power within heterogeneous (Multi-CPU-GPU) platforms

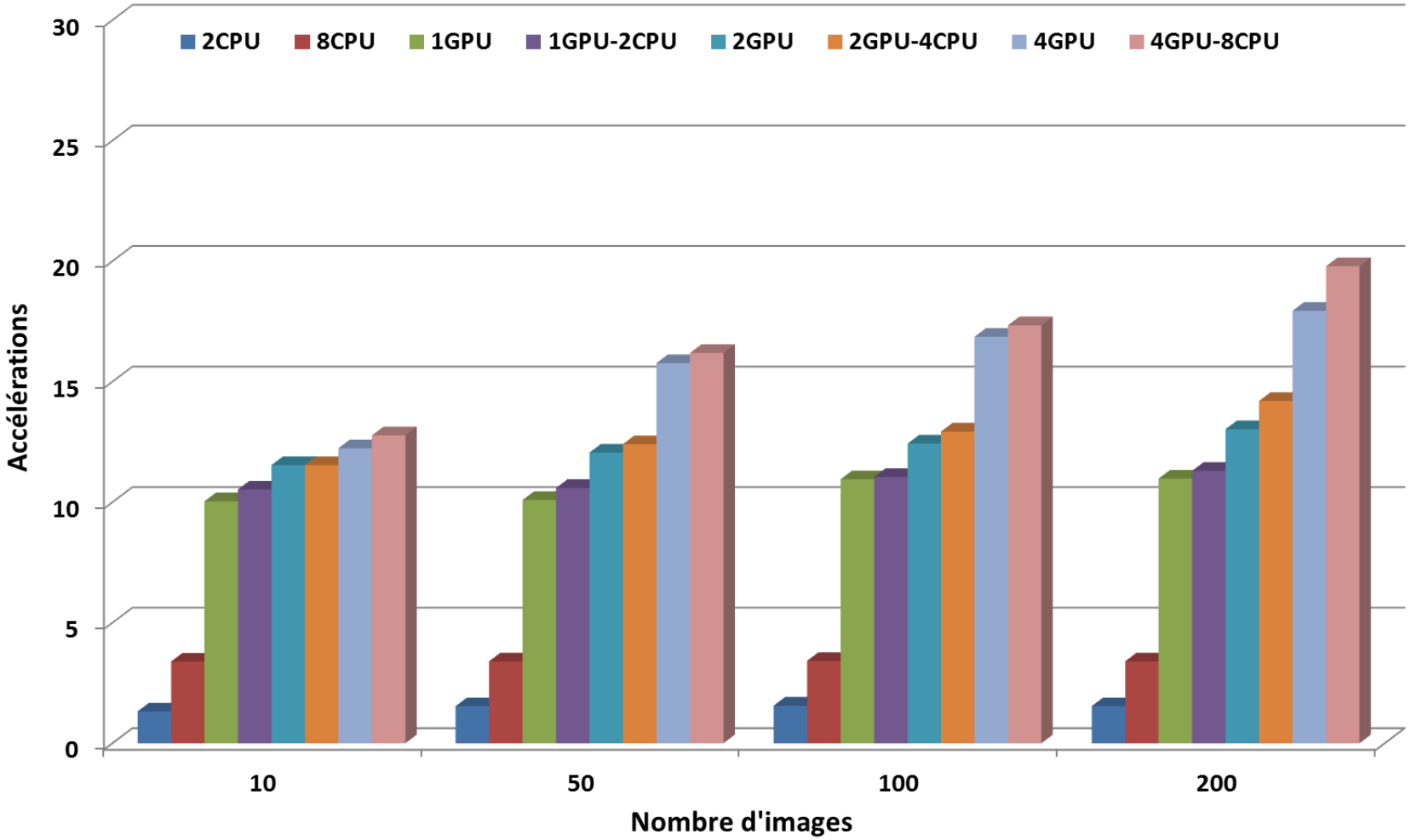
Edges and corners detection on GPU : **CUDA streaming**

Multiple images processing on heterogeneous platforms

StarPU: runtime for exploiting Heterogeneous platforms.



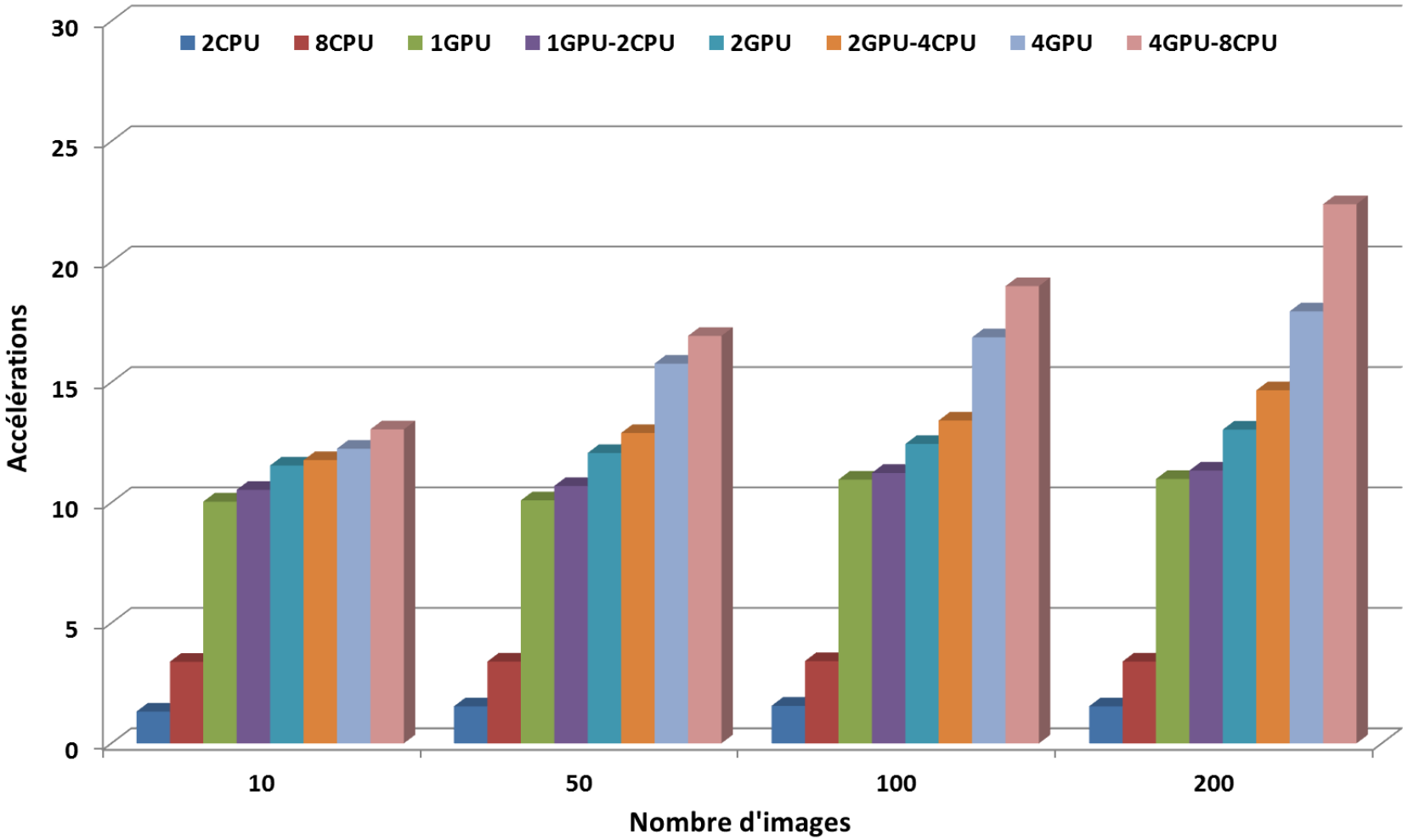
Heterogeneous treatment : results



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

Edge and corners detection on heterogeneous platforms

Heterogeneous treatment : results

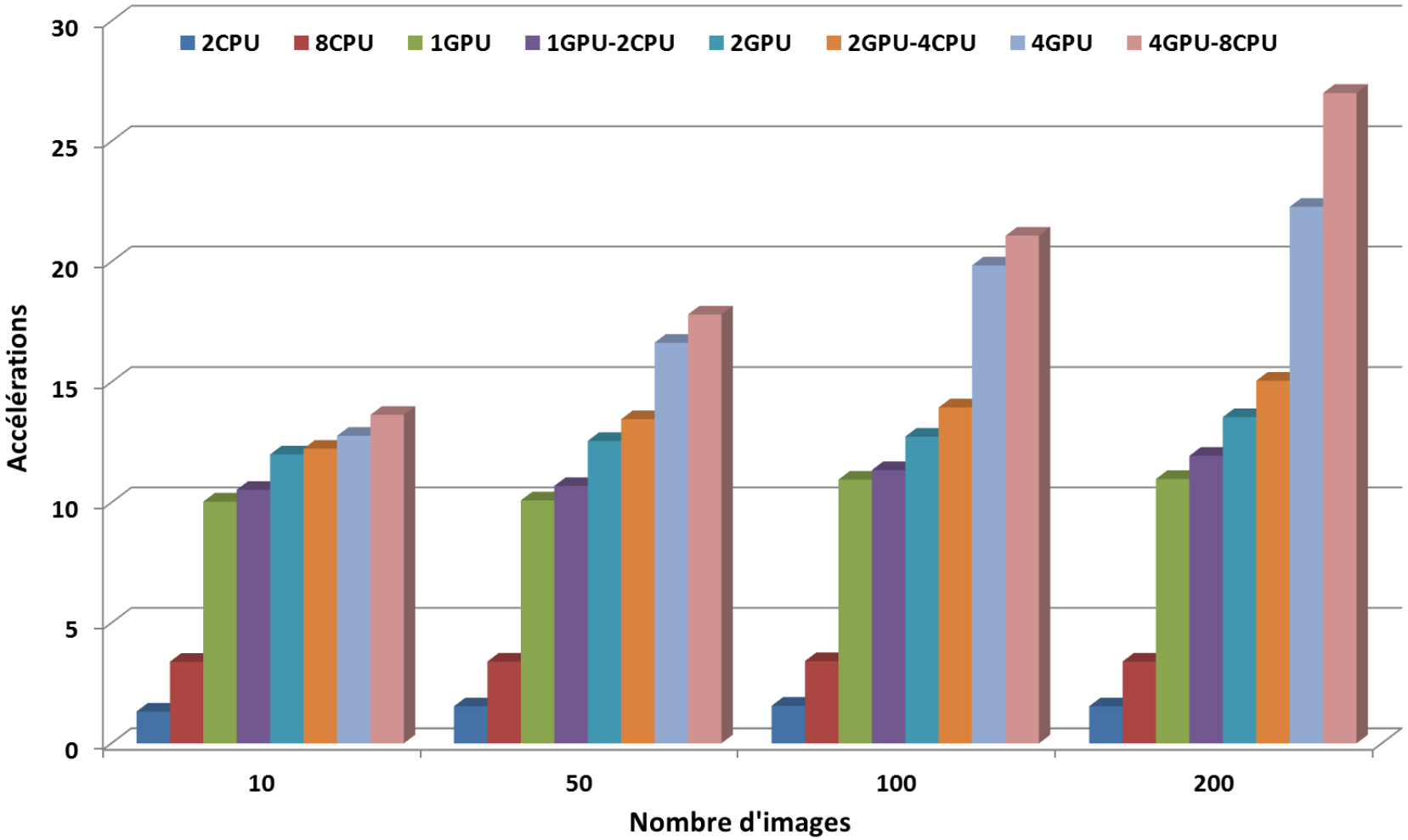


Résolution d'images : 512x512, 1024x1024, 2048x2048, 3936x3936

Edge and corners detection on heterogeneous platforms

Dynamic scheduling

Heterogeneous treatment : results



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

Edge and corners detection on heterogeneous platforms

Dynamic scheduling + CUDA streaming

PLAN

Introduction

I. GPU Presentation

II. GPU Programming

III. Exploitation of Multi-CPU-GPU Architectures

IV. **Application for Multimedia processing**

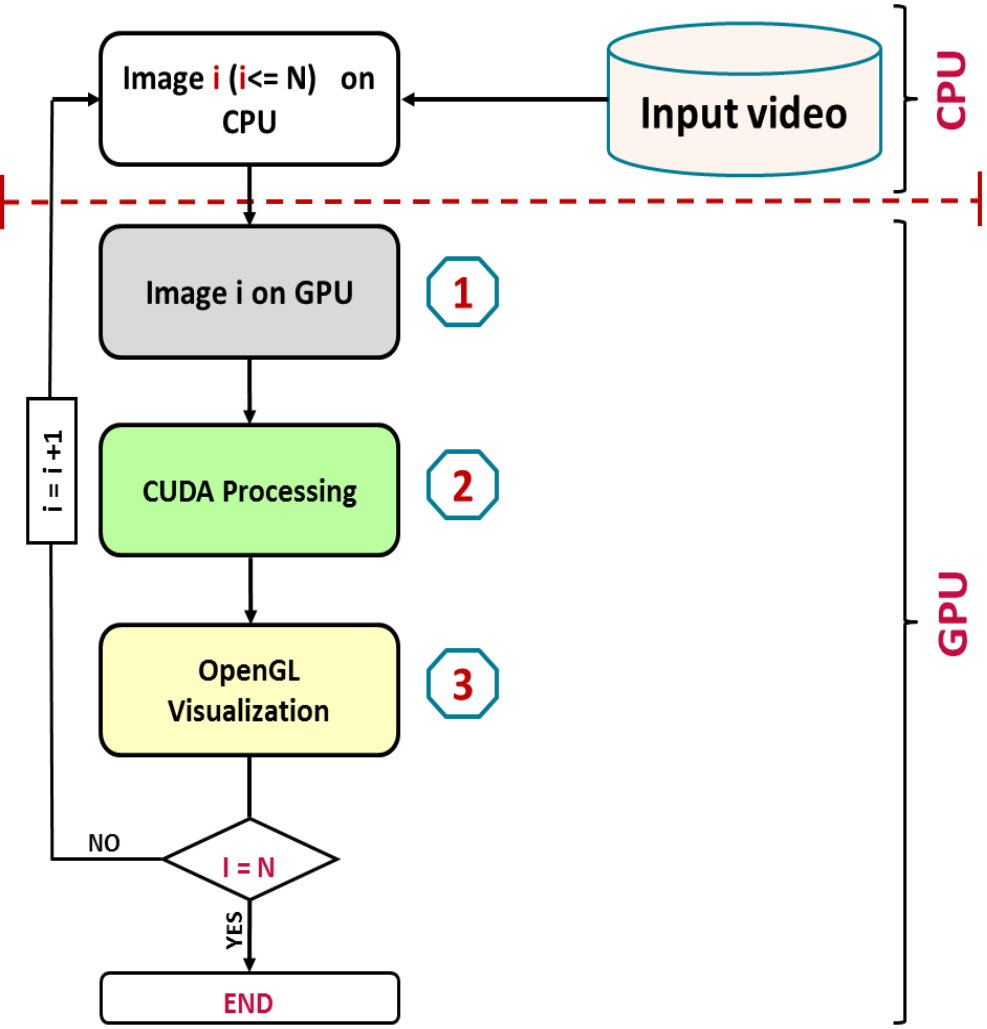
1. Multi-CPU/Multi-GPU based image processing
2. **Multi-GPU based video processing in Real Time**

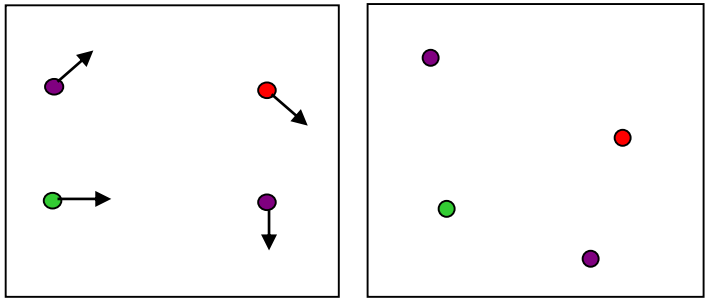
V. Multi-CPU/Multi-GPU based framework for HD image and video processing

VI. Experimental Results

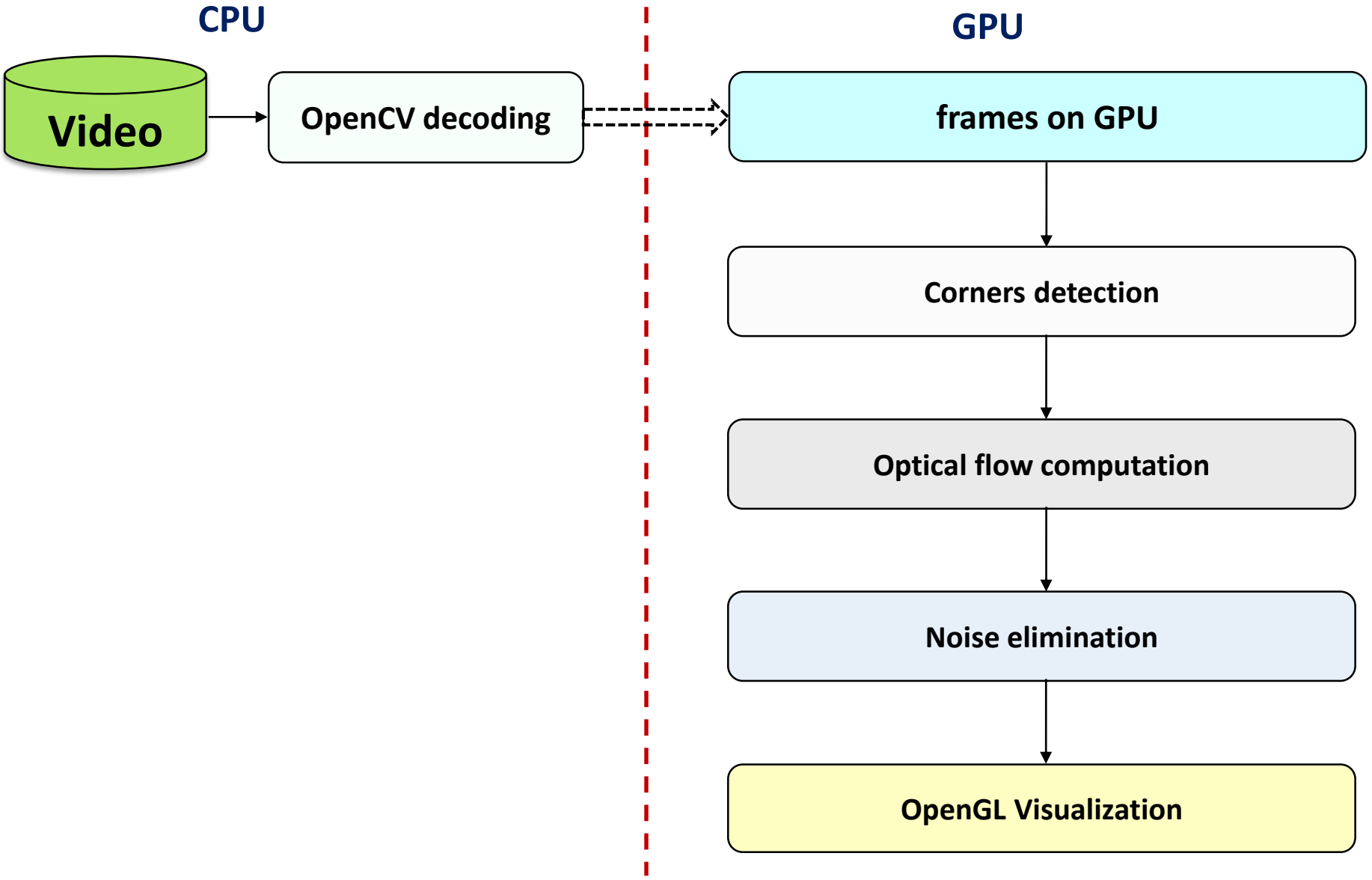
Conclusion

Video Processing on GPU : scheme development

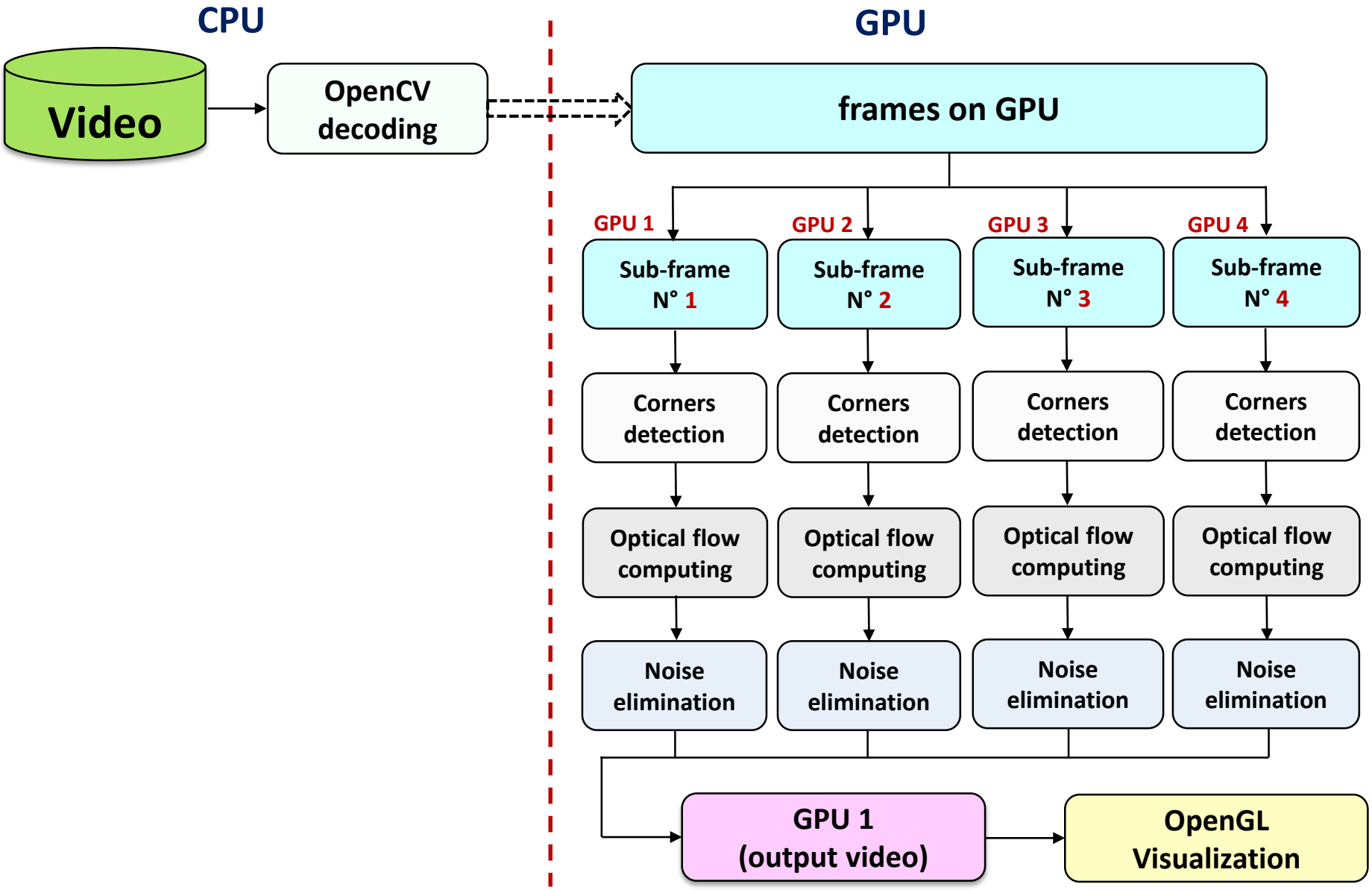




GPU based motion tracking



Multi-GPU based motion tracking



CPU based motion tracking

OpenCV CPU

10 FPS

CPU based motion tracking in Full HD videos (1920x1080)

GPUCV based motion tracking

OpenCV GPU

Noise

Noise

Noise

Noise

19 FPS

GPUCV based motion tracking in Full HD videos (1920x1080)

GPU based motion tracking

GPU

Noise elim

Noise elim

Noise elim

Noise elim

62 FPS

GPU based motion tracking in Full HD videos (1920x1080)

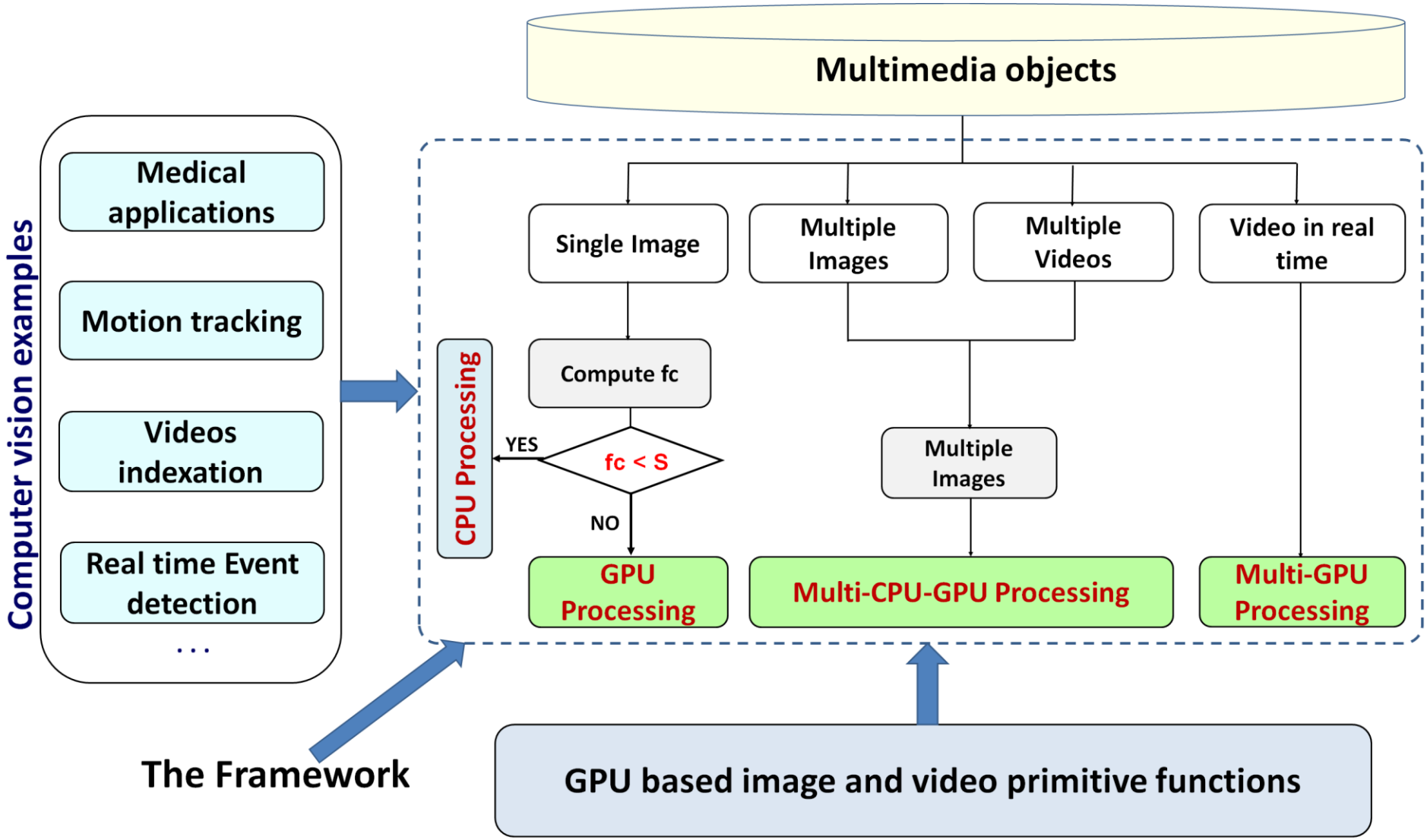
PLAN

Introduction

1. GPU Presentation
2. GPU Programming
3. Exploitation of Multi-CPU-GPU Architectures
4. Application for Multimedia processing
    1. Multi-CPU/Multi-GPU based image processing
    2. Multi-GPU based video processing in Real Time
    1. **Multi-CPU/Multi-GPU based framework for HD image and video processing**
5. Experimental Results

Conclusion

The proposed framework



AGENDA

Introduction

I. GPU Programming

II. Our Context

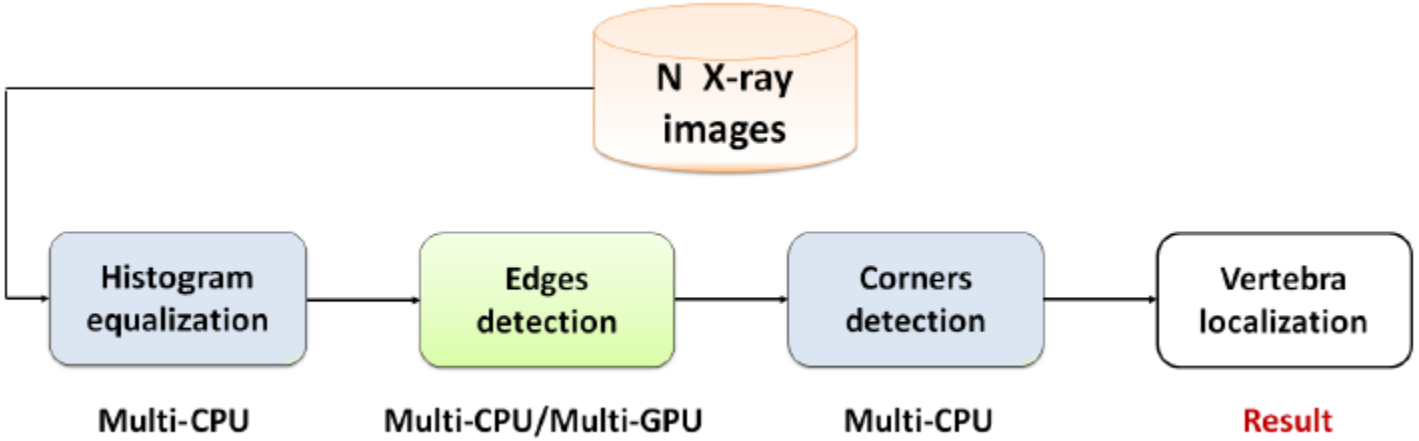
III. Exploitation of Multi-CPU-GPU platforms in Multimedia processing

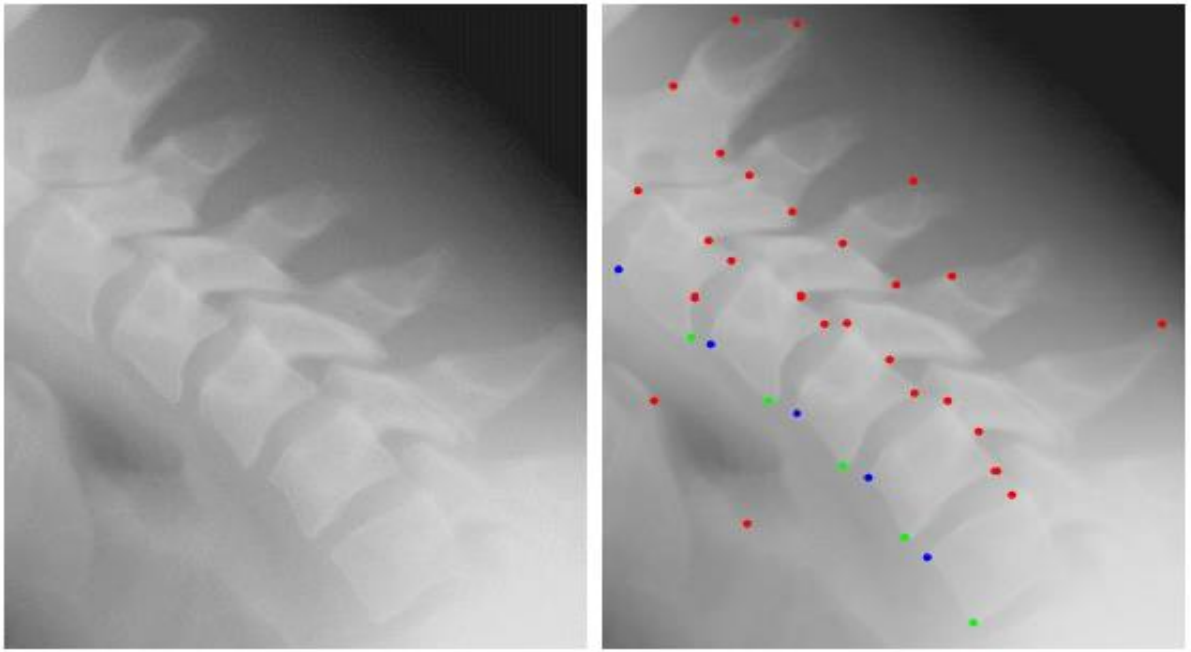
1. Multi-CPU/Multi-GPU based image processing
2. Real time Multi-GPU processing of HD/Full HD videos

IV. **Experimental results : use cases**

Conclusion

1st use case : Vertebra segmentation





(a) Original Image (b) Detection of the vertebræ

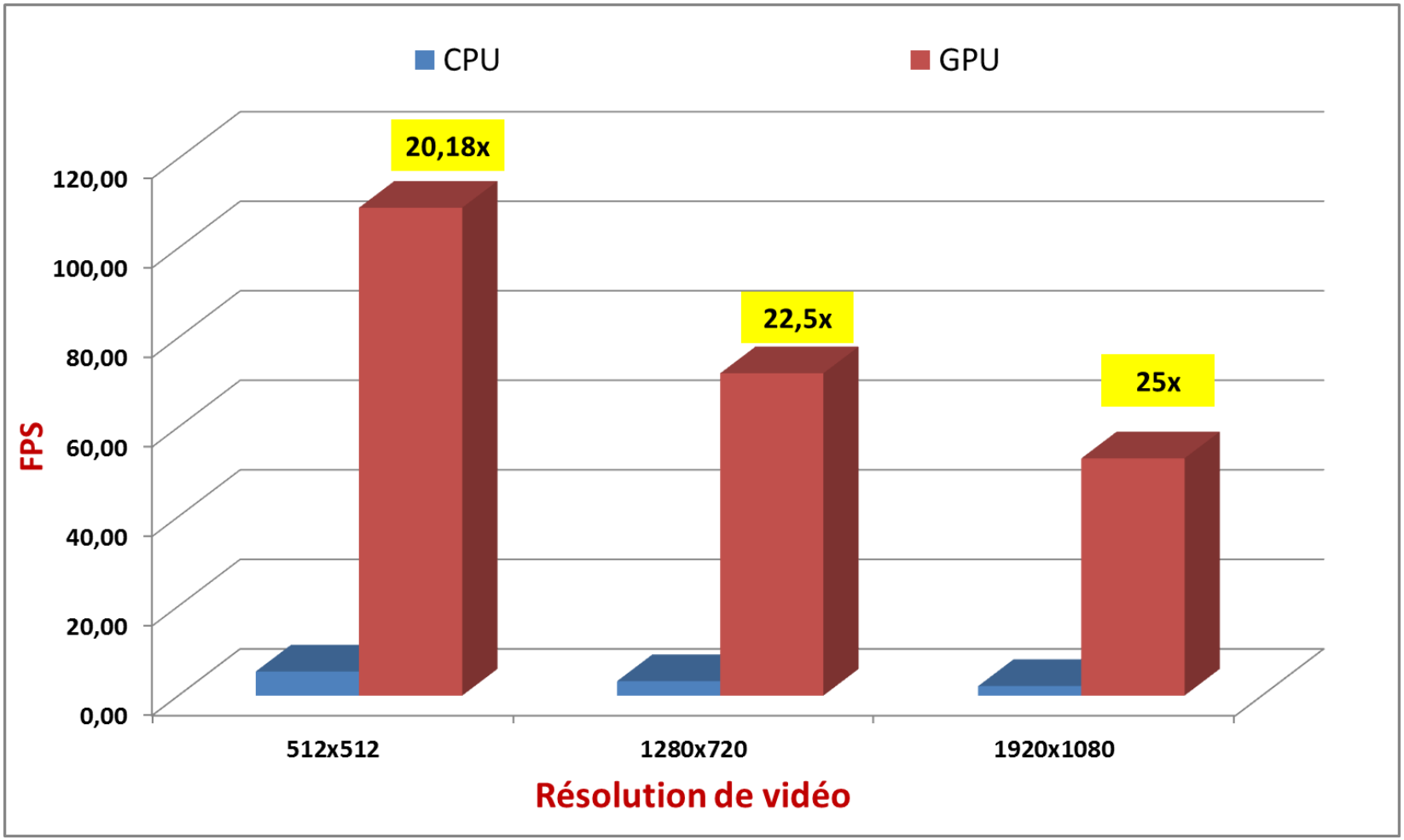
1st use case : Vertebra segmentation

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 of heterogeneous vertebra detection using 200 images (1476x1680)

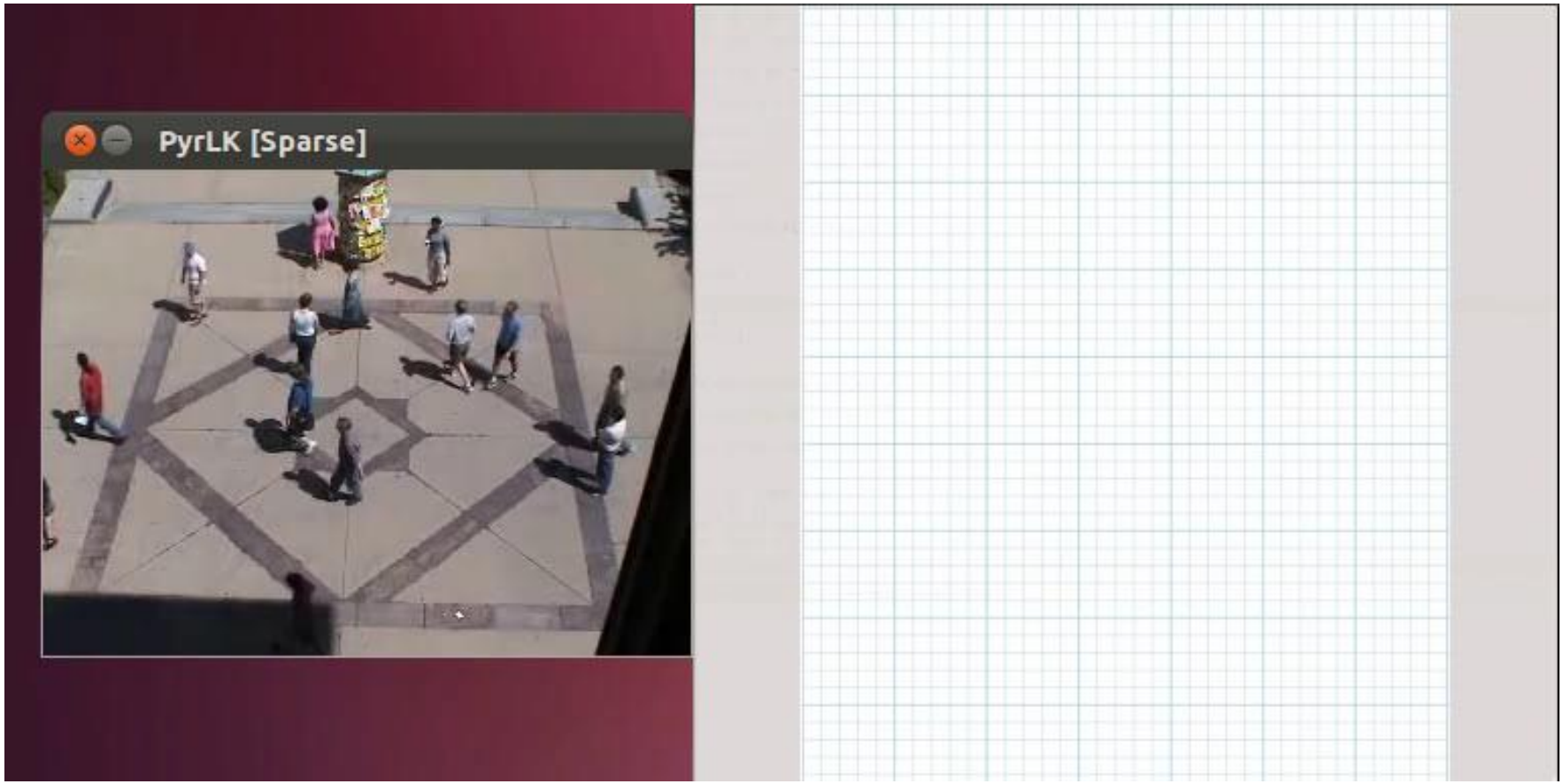
2nd use case : Movement detection within mobile camera

2nd use case : Movement detection within mobile camera

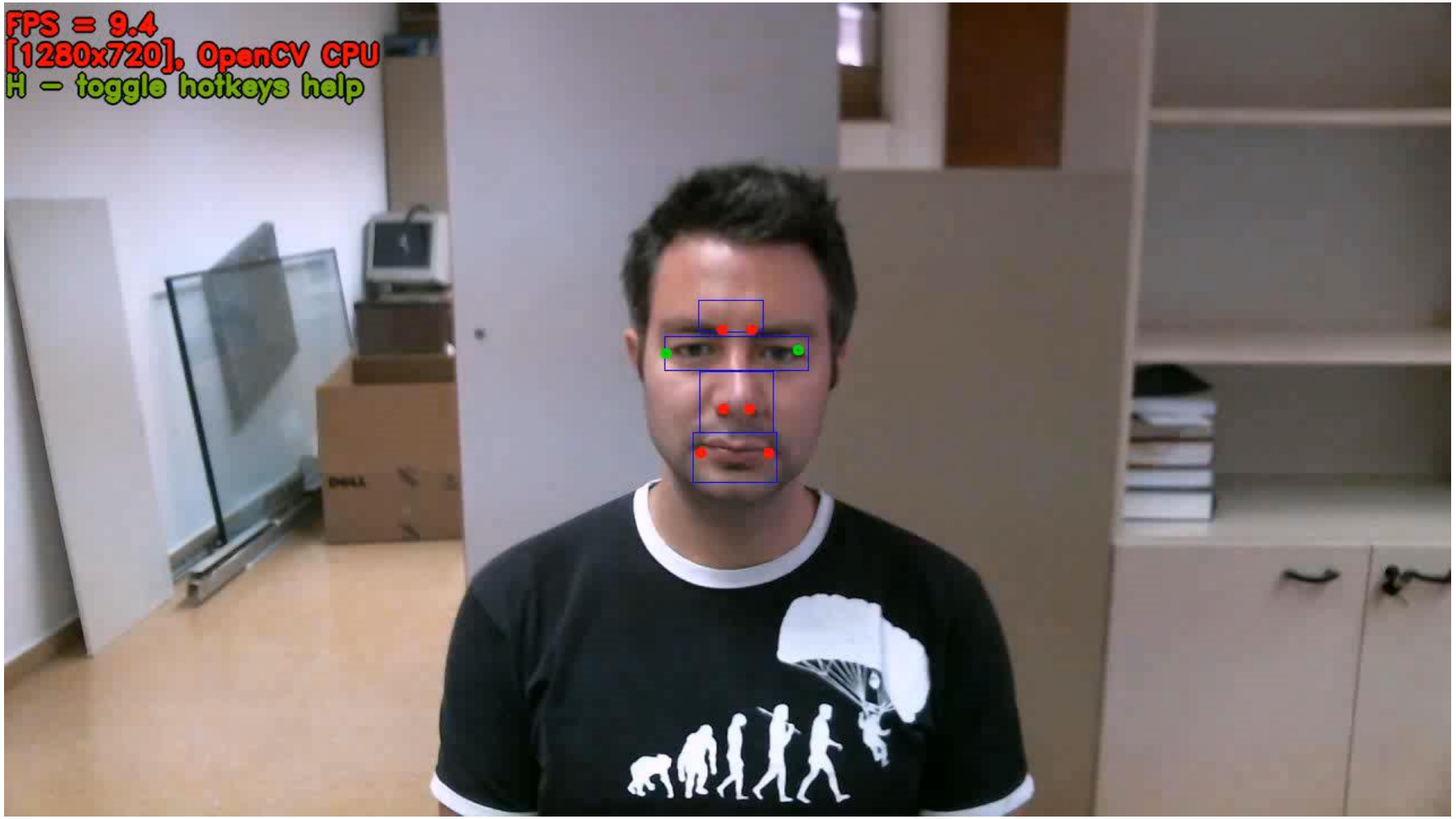


GPU performance : movement detection within mobile camera

3rd use case: real time event detection



4th use case: eyes, nose and frontal face detection



4th use case: eyes, nose and frontal face tracking

Conclusion

- Parallel processing between image pixels
- Parallel and heterogeneous processing between images
- Efficient exploitation of GPU memories
- Multi-GPU treatments for HD/Full HD videos
- Efficient selection of resources (CPUs or/and GPUs)

References

[1] Mahmoudi Sidi Ahmed, Belarbi Mohammed Amin, Mahmoudi Said, Belalem Ghalem, "**Towards a Smart Selection of Resources in the Cloud for Low-energy Multimedia Processing**" in Concurrency & Computation : Practice & Experience (2017)

[2] CloudTech : Belarbi Mohammed Amin, Mahmoudi Said, Belalem Ghalem, Mahmoudi Sidi, "**Web-based Multimedia Research and Indexation for Big Data Databases** ", *in CloudTech17: 3rd International Conference on Cloud Computing Technologies and Applications*, Rabat, Maroc (2017)

[3] Mahmoudi Sidi, Ammar Mohammed, Luque Joris Guillaume, Abbou Amine, "**Real Time GPU-Based Segmentation and Tracking of the Left Ventricle on 2D Echocardiography**« , *In International work-conference on bioinformatics and bimedical engineering (2016)*.

[4] Mahmoudi Sidi, Manneback Pierre, "**Multi-CPU/Multi-GPU Based Framework for Multimedia Processing**« , *in IFIP Advances in Information and Communication Technology. Computer Science and Its Applications,* 456, 2015, 54-65 (2015)

[5] Mahmoudi Sidi, "**Multi-GPU based framework for real-time motion analysis and tracking in multi-user scenarios**« , *in EAI Endorsed Transactions. Creative Technologies , 2, 2, 1-10, e5* (2015

References

[6] Possa Paulo, Mahmoudi Sidi, Harb Naim, Valderrama Carlos, Manneback Pierre, "**A Multi-Resolution FPGA-Based Architecture for Real-Time Edge and Corner Detection**" *in IEEE Transactions on Computers, 63, 10, Oct 2014, 2376-2388, DOI10.1109/TC.2013.130 (2014)*

[7] Mahmoudi Sidi, Kierzynka Michal, Manneback Pierre, Kurowski Krzysztof, "**Real-time motion tracking using optical flow on multiple GPUs**" *in Bulletin of the Polish Academy of Sciences. Technical Sciences, 62, 1, 139-150, 10.2478/bpasts-2014-0016 (2014)*

[8] Mahmoudi Sidi, Ozkan Erencan, Manneback Pierre, Tosun Souleyman, "**Taking Advantage of Heterogeneous Platforms in Image and Video Processing**" *in Complex HPC book" , Wiley, 978-1-118-71205-4* (2014)

[9] Da Cunha Possa Paulo, Mahmoudi Sidi, Harb Naim, Valderrama Carlos, "**A New Self-Adapting Architecture for Feature Detection**" in *Lecture Notes in Computer Science, 978-1-4673-2257-7, 2012(22) Oslo, 643 - 646, 10.1109/FPL.2012.6339149* (2012)

[10] Mahmoudi Sidi, Manneback Pierre, Augonnet C., Thibault S., "**Traitements d'images sur architectures parallèles et hétérogènes**" *in Technique et Science Informatiques, 31/8-10 - 2012, 8-9-10/2012, 1183-1203, 10.3166/tsi.31.1183-1203* (2012)

References

[11] Mahmoudi Sidi Ahmed, Manneback Pierre, Augonnet C., Thibault S., "**Détection optimale des coins et contours dans des bases d'images volumineuses sur architectures multicoeurs hétérogènes***", in "20èmes Rencontres Francophones de l'Informatique Parallèle , RenPar'20, Saint-Malo, France* (2011)

[12] Mahmoudi Sidi Ahmed, Manneback Pierre, « **Multi-GPU based Event Detection and Localization using High Definition Videos** » *", in "The 4th International Conference on Multimedia Computing and Systems (ICMCS'14)* , Marrakesch, Morocco (2014)

THANK YOU