# Attaining High Performance in General-Purpose Computations on Current Graphics Processors

Francisco Igual Rafael Mayo Enrique S. Quintana-Ortí

Departamento de Ingeniería y Ciencia de los Computadores.
University Jaume I.
Castellón (Spain)

# Motivation (I)

- GPUs have become the first widely available HPC platform
- Successfully used in linear algebra, image processing, data mining, simulations. . .
- Recently introduced advances:
  - Hardware level: **Unified Architecture**
  - Software level: **CUDA**
- Peak performance up to $10x$ (Core2Duo vs Nvidia 8800)

Is it always a benefit the use of GPUs?

# Motivation (I)

- GPUs have become the first widely available HPC platform
- Successfully used in linear algebra, image processing, data mining, simulations...
- Recently introduced advances:
  - Hardware level: **Unified Architecture**
  - Software level: **CUDA**
- Peak performance up to $10x$ (Core2Duo vs Nvidia 8800)

Is it always a benefit the use of GPUs?

# Motivation (II)

- GPUs are specific-purpose processors
- The GPU architecture is focused on graphics performance
- Not every general-purpose algorithm will fit correctly on GPU
- Newest GPU models introduce interesting features from the GPGPU point of view

Which algorithms fit better on the graphics processor?

Is it always better the use of GPU?

Impact of the improvements of last generations of GPUs

# Motivation (II)

- GPUs are specific-purpose processors
- The GPU architecture is focused on graphics performance
- Not every general-purpose algorithm will fit correctly on GPU
- Newest GPU models introduce interesting features from the GPGPU point of view

Which algorithms fit better on the graphics processor?

Is it always better the use of GPU?

Impact of the improvements of last generations of GPUs

# Motivation (II)

- GPUs are specific-purpose processors
- The GPU architecture is focused on graphics performance
- Not every general-purpose algorithm will fit correctly on GPU
- Newest GPU models introduce interesting features from the GPGPU point of view

Which algorithms fit better on the graphics processor?

Is it always better the use of GPU?

Impact of the improvements of last generations of GPUs

# Motivation (II)

- GPUs are specific-purpose processors
- The GPU architecture is focused on graphics performance
- Not every general-purpose algorithm will fit correctly on GPU
- Newest GPU models introduce interesting features from the GPGPU point of view

Which algorithms fit better on the graphics processor?

Is it always better the use of GPU?

Impact of the improvements of last generations of GPUs

# Outline

# Introduction and goals

## Methodology

- Microbenchmarks from BLAS and image processing
- Implement the benchmarks on CPU and both generations of GPUs
- Use only optimized benchmarks on both architectures

## Goals

- Extract the features of GPU-like algorithms through the evaluation of different routines
- Compare both generations of GPU architectures
- Extract the impact of data transfers on both generations
- Decide which algorithms are CPU or GPU-like

# Introduction and goals

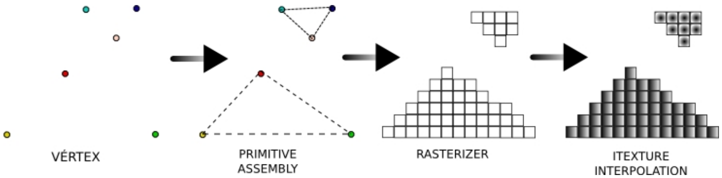
## Methodology

- Microbenchmarks from BLAS and image processing
- Implement the benchmarks on CPU and both generations of GPUs
- Use only optimized benchmarks on both architectures

## Goals

- Extract the features of GPU-like algorithms through the evaluation of different routines
- Compare both generations of GPU architectures
- Extract the impact of data transfers on both generations
- Decide which algorithms are CPU or GPU-like

# Classical GPU pipeline




- Each stage of the pipeline works with a different datatype
- Each stage is computed on a different type of processor (vertex processors and fragment processors)
- But the processors are programmable. . .

# Classical GPU overview

## Programmability

- Both vertex processors and fragments processors are programmable
- Usually fragment processors (FP) are programmed
- The replication of FP fits well with data-parallel routines
- SIMD architecture

## Disadvantages

- No memory hierarchy at all
- Just one memory direction could be written for each fragment
- Hard to program (OpenGL + Cg)

# Classical GPU overview

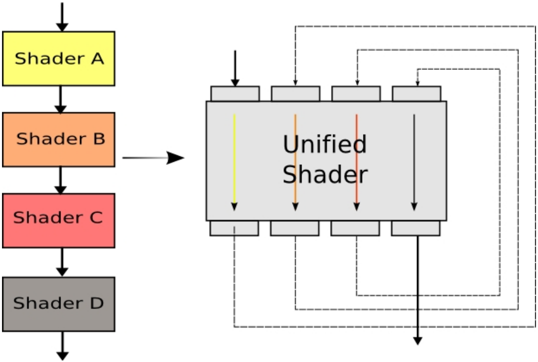
## Programmability

- Both vertex processors and fragments processors are programmable
- Usually fragment processors (FP) are programmed
- The replication of FP fits well with data-parallel routines
- SIMD architecture

## Disadvantages

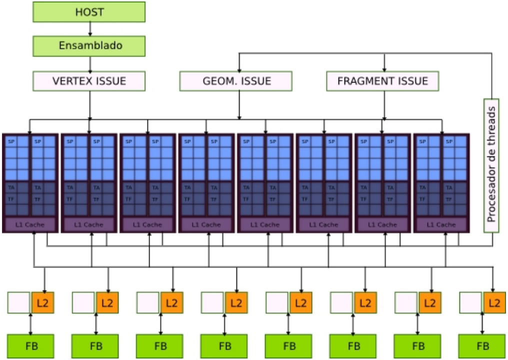
- No memory hierarchy at all
- Just one memory direction could be written for each fragment
- Hard to program (OpenGL + Cg)

# Unified GPU shader




- One unified shader with multiple functionality
- Works with any type of graphical data
- GPGPU: all programmable shaders are now available

# Unified GPU architecture: G80




**NVIDIA G80**

- Main unified implementation
- Up to 128 cores in the unified shader
- Introduction of memory hierarchy
- Higher clock frequency in the shader
- More flexibility in reading/writing to memory
- CUDA library

# Selection of benchmarks

## Features to be evaluated

1. Data parallelism
2. Input data reutilization
3. Computational intensity per stream element

Routines to be evaluated:

**SGEMM:** $C = \alpha AB + \beta C$

**SGEMV:** $y = \alpha Ax + \beta y$

**SAXPY:** $y = \alpha x + y$

**SSCAL:** $y = \alpha y$

**2D Convolution**

# Features of the benchmarks

- Selected routines:

| Routine | Type | Data parallelism | Input reutilization | Arith. intensity |
|---|---|---|---|---|
| **SGEMM** | BLAS 3 | High | High | High |
| **SGEMV** | BLAS 2 | High | Medium | Medium |
| **SAXPY** | BLAS 1 | High | None | Low |
| **SSCAL** | BLAS 1 | High | None | Lowest |
| **Convolution** | Image | High | Low | Medium |

- Hardware setup:

**Classical architecture**

AMD Athlon 2400+
+
Nvidia Geforce 6200

**Unified architecture**

Intel Core2Duo 1.86 Ghz
+
Nvidia Geforce 8800 Ultra

GotoBLAS / CUBLAS 1.0

# Features of the benchmarks

- Selected routines:

| Routine | Type | Data parallelism | Input reutilization | Arith. intensity |
|---|---|---|---|---|
| **SGEMM** | BLAS 3 | High | High | High |
| **SGEMV** | BLAS 2 | High | Medium | Medium |
| **SAXPY** | BLAS 1 | High | None | Low |
| **SSCAL** | BLAS 1 | High | None | Lowest |
| **Convolution** | Image | High | Low | Medium |

- Hardware setup:

| **Classical architecture** | **Unified architecture** |
|---|---|
| AMD Athlon 2400+ | Intel Core2Duo 1.86 Ghz |
| + | + |
| Nvidia Geforce 6200 | Nvidia Geforce 8800 Ultra |

GotoBLAS / CUBLAS 1.0

# Features of the benchmarks

- Selected routines:

| Routine | Type | Data parallelism | Input reutilization | Arith. intensity |
|---|---|---|---|---|
| **SGEMM** | BLAS 3 | High | High | High |
| **SGEMV** | BLAS 2 | High | Medium | Medium |
| **SAXPY** | BLAS 1 | High | None | Low |
| **SSCAL** | BLAS 1 | High | None | Lowest |
| **Convolution** | Image | High | Low | Medium |

- Hardware setup:

**Classical architecture**

AMD Athlon 2400+
+
Nvidia Geforce 6200

**Unified architecture**

Intel Core2Duo 1.86 Ghz
+
Nvidia Geforce 8800 Ultra

GotoBLAS / CUBLAS 1.0

# Features of the benchmarks

- Selected routines:

| Routine | Type | Data parallelism | Input reutilization | Arith. intensity |
|---|---|---|---|---|
| **SGEMM** | BLAS 3 | High | High | High |
| **SGEMV** | BLAS 2 | High | Medium | Medium |
| **SAXPY** | BLAS 1 | High | None | Low |
| **SSCAL** | BLAS 1 | High | None | Lowest |
| **Convolution** | Image | High | Low | Medium |

- Hardware setup:

| Classical architecture | Unified architecture |
|---|---|
| AMD Athlon 2400+ | Intel Core2Duo 1.86 Ghz |
| + | + |
| Nvidia Geforce 6200 | Nvidia Geforce 8800 Ultra |

GotoBLAS / CUBLAS 1.0

# Features of the benchmarks

- Selected routines:

| Routine | Type | Data parallelism | Input reutilization | Arith. intensity |
|---|---|---|---|---|
| **SGEMM** | BLAS 3 | High | High | High |
| **SGEMV** | BLAS 2 | High | Medium | Medium |
| **SAXPY** | BLAS 1 | High | None | Low |
| **SSCAL** | BLAS 1 | High | None | Lowest |
| **Convolution** | Image | High | Low | Medium |

- Hardware setup:

**Classical architecture**

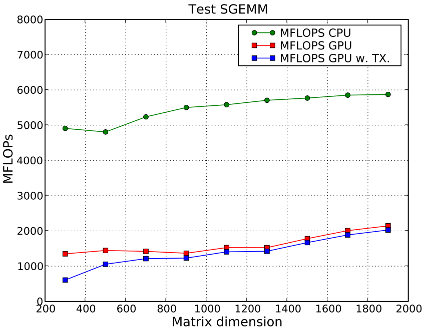
AMD Athlon 2400+
+
Nvidia Geforce 6200

**Unified architecture**

Intel Core2Duo 1.86 Ghz
+
Nvidia Geforce 8800 Ultra

GotoBLAS / CUBLAS 1.0

# Results on classical GPUs. SGEMM

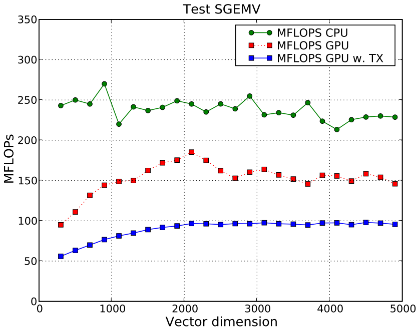



**Data reutilization impact**

- CPU outperforms GPU
- Speedup up to $x4$
- Impact of cache hierarchy on CPU
- Data transfers not significant

- Input data reutilization seems a key factor on GPU
- SGEMV exhibits less input data reutilization...

# Results on classical GPUs. SGEMV



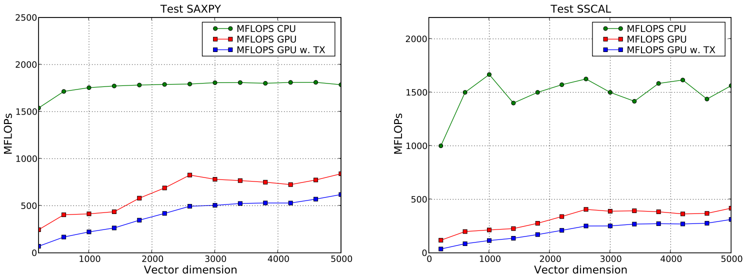

**Data reutilization impact**

- CPU outperforms GPU, but up to $x2$
- Data transfers more significant

Input data reutilization is a key factor on GPU

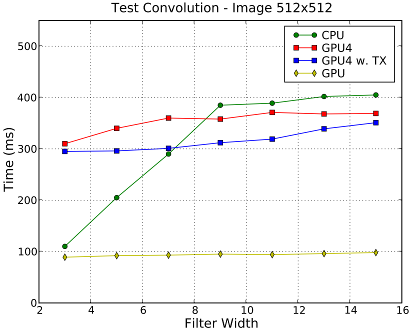
# Results on classical GPUs. SAXPY and SSCAL

## Arithmetic intensity




- SAXPY and SSCAL are stream-oriented operations
- They should attain high performance on GPUs
- However, CPU outperforms GPU for these operations
- Arithmetic intensity per stream element is another key factor

# Results on classical GPUs. Convolution



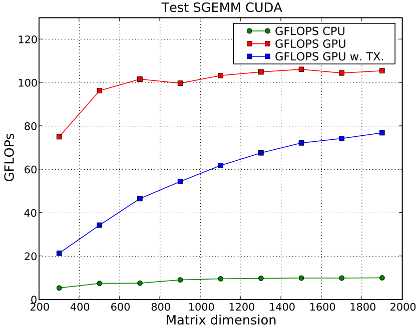

**Convolution performance**

- Attained performance similar to that on CPU
- Optimized implementation for vectorial capabilities (`GPU4`) attains $4x$ speedup

## Required features on GPU

- High data parallelism
- Low input data reutilization
- High arithmetic intensity per stream element

# Results on unified GPUs. SGEMM

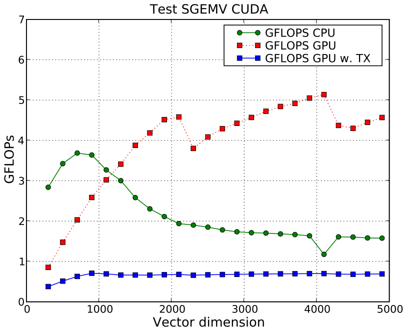



**Data reutilization impact**

- GPU outperforms CPU
- Speedup up to $x10$
- However, only attaining $20\%$ of the peak performance of the GPU
- Data transfers more significant than on previous generations

- Input data reutilization is still important on GPU, but less than on previous generations
- Reason: more sophisticated memory hierarchy

# Results on unified GPUs. SGEMV



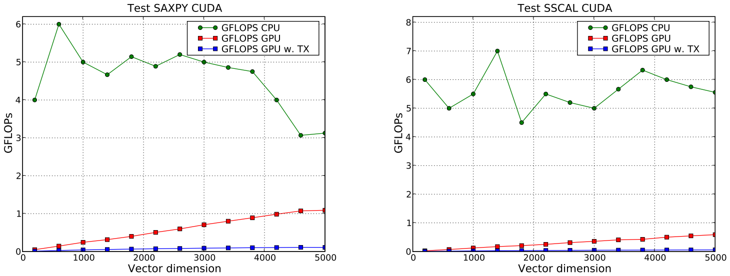

**Data reutilization impact**

- GPU outperforms CPU for big matrices
- Impact of cache hierarchy on CPU: faster for small streams of data
- Data transfers very significant

- Data transfer stage is a very important factor now
- G6 $\rightarrow$ G80 $(x20)$ but AGP $\rightarrow$ PCIExpress $(x2)$
- Modern GPUs work better with big streams of data

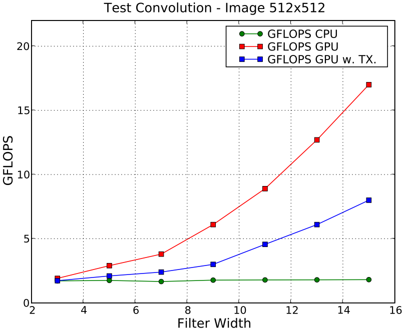
# Results on unified GPUs. SAXPY and SSCAL

## Arithmetic intensity




- However, arithmetic intensity per element is still a key factor
- CPU outperforms GPU for these operations

# Results on unified GPUs. Convolution



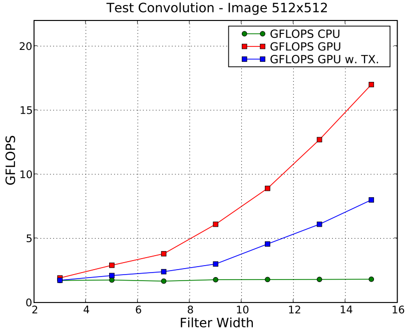

**Convolution performance**

- Attained the highest speedup of the benchmarks
- Highest data transfer impact

## Required features on unified GPUs

- High data parallelism, low reutilization, high intensity
- Low ratio transfer/calculation
- Big data streams

# Results on unified GPUs. Convolution




**Convolution performance**

- Attained the highest speedup of the benchmarks
- Highest data transfer impact

## Required features on unified GPUs

- High data parallelism, low reutilization, high intensity
- Low ratio transfer/calculation
- Big data streams

# Conclusions

- The GPU is an interesting platform for HPC
- However, not every algorithm extracts all the performance
- Desirable features:
  - High data parallelism
  - Low input data reutilization
  - High arithmetic intensity
  - Operating with big streams of data
- Necessary to design algorithm minimizing data transfers

# Thank you...

**For more information. . .**

http://www3.uji.es/˜figual
figual@icc.uji.es