

---

ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA  
SEDE DI CESENA  
FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI  
CORSO DI LAUREA IN SCIENZE DELL'INFORMAZIONE

**HIGH PERFORMANCE COMPUTING  
SU UNITA' GRAFICHE PROGRAMMABILI**

**Tesi di laurea in**  
Fisica Numerica

**Relatore**  
Prof. Renato Campanini

**Presentata da**  
Mauro De Carolis

**Co-relatore**  
Dott. Matteo Roffilli

Sessione III  
Anno Accademico 2004/2005

---

---

# HPC

## *High Performance Computing*

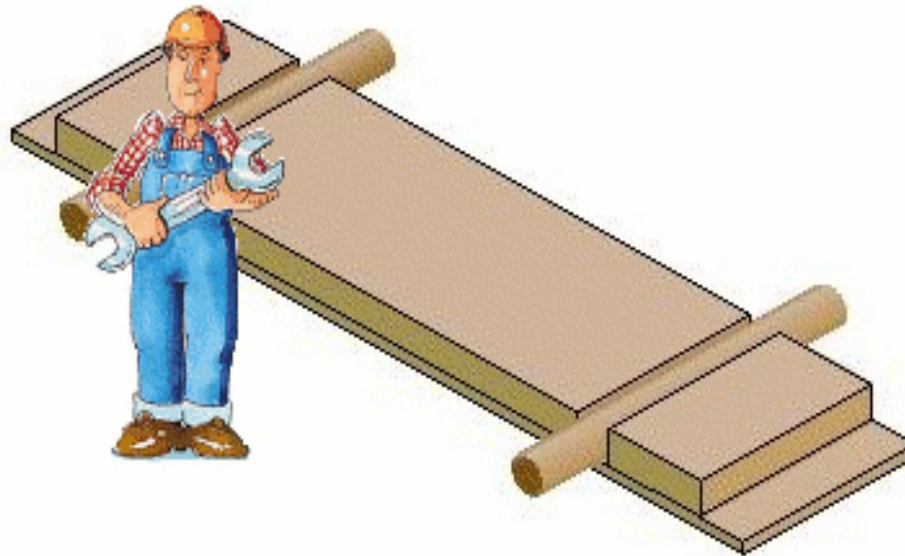
HPC = Ottimizzare ( HW + SW )

---

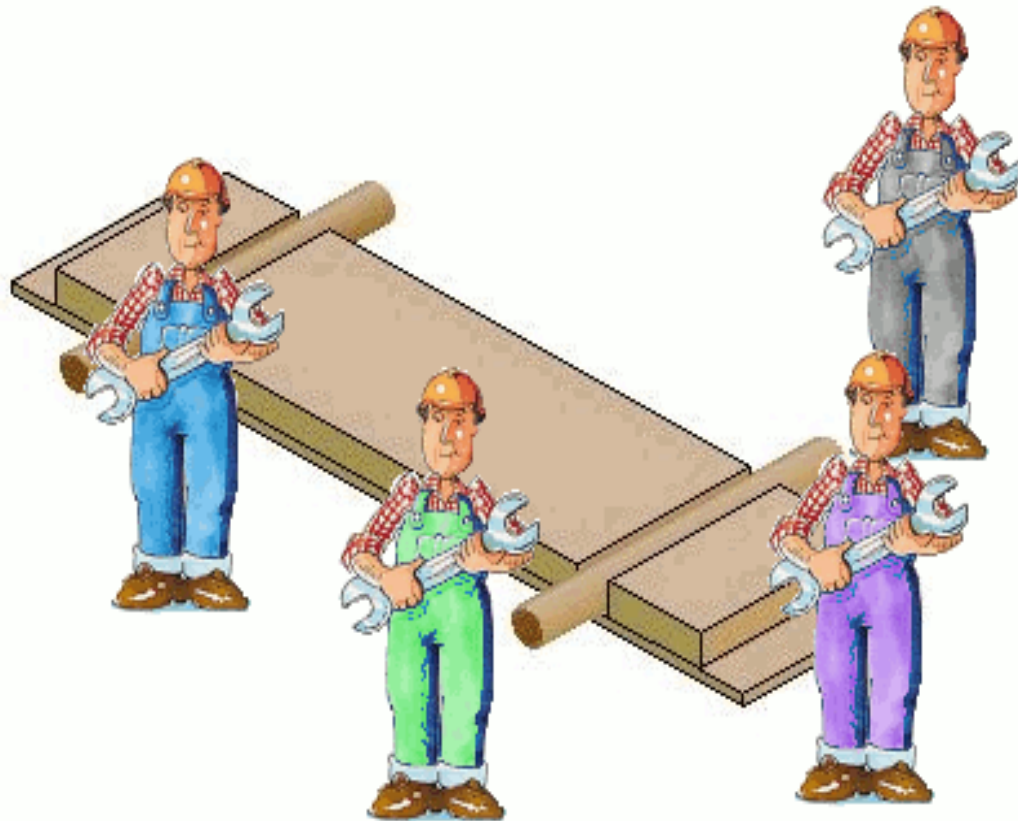
# Architetture per l'HPC

- Singolo processore
- Processore Vettoriale (MMX,SSE)
- Processori Paralleli

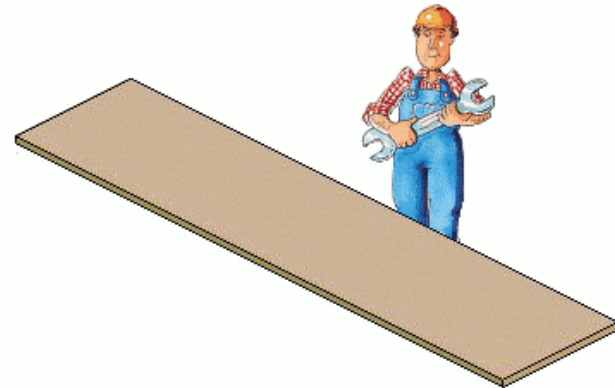
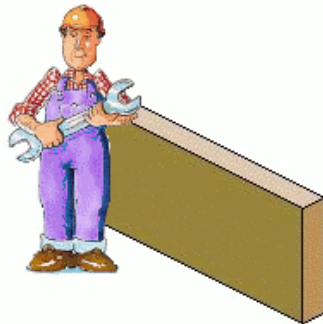
# Processore Singolo



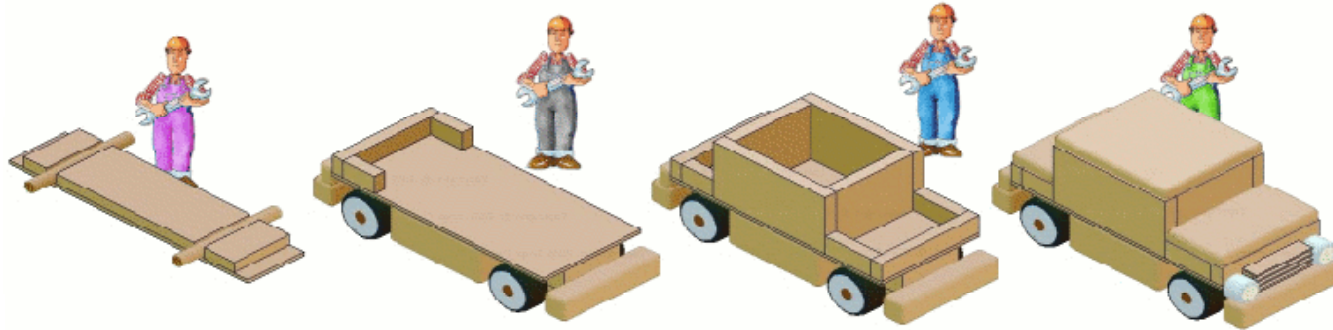
# Processore Vettoriale (MMX,SSE)



# Processori Paralleli



# Stream Processor



L'output di un kernel costituisce l'input del kernel successivo.

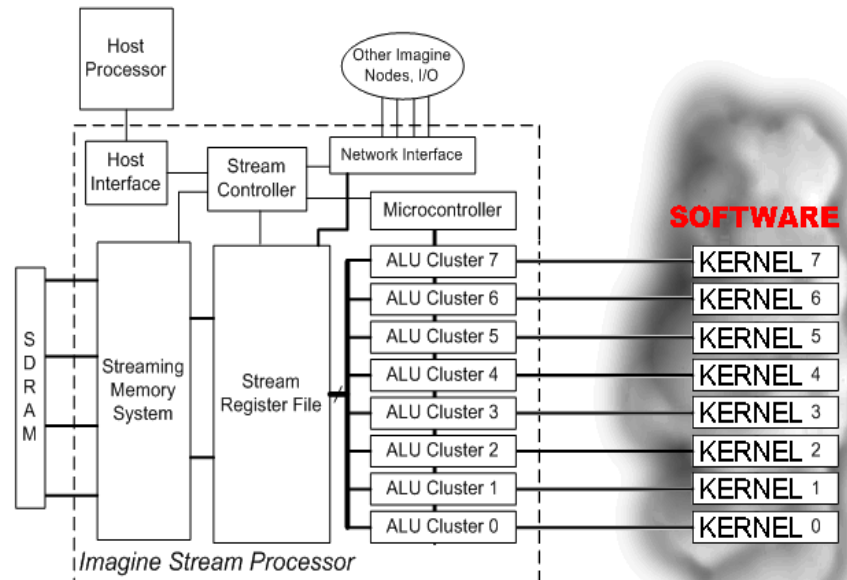
# Stream Processor

- **Imagine** - PROTOTIPO

( Stanford University, Texas Instruments, Aprile 2002 )

- **Cell** - Playstation3 & Supercomputer IBM

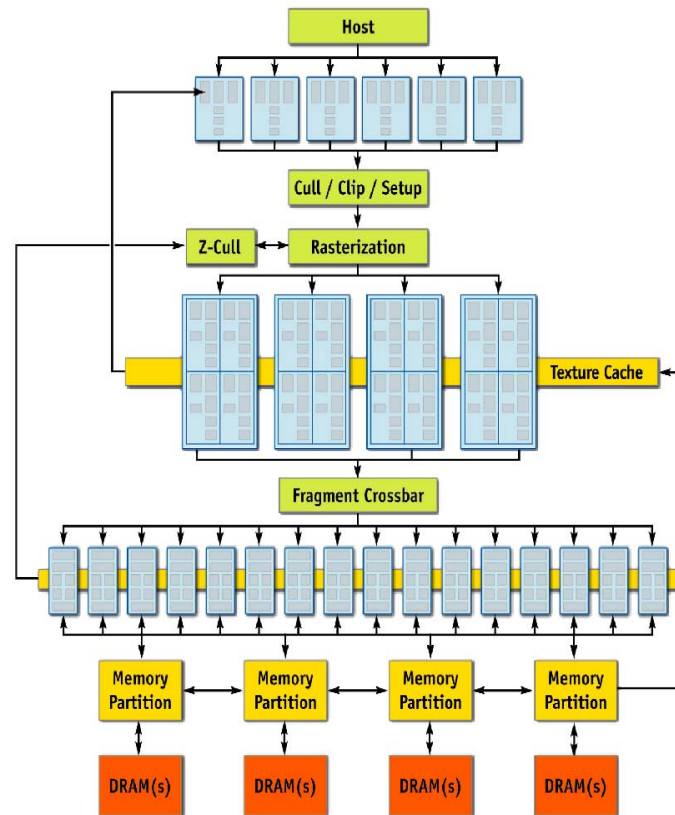
( Sony, IBM, 13 Marzo 2006 )





# GPU

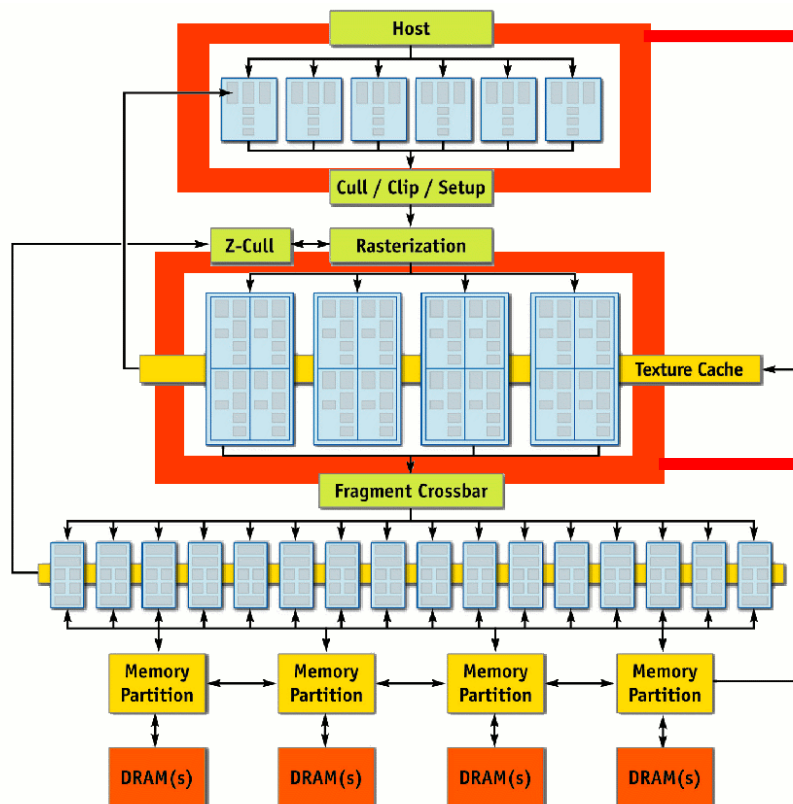
## *Graphic Processing Unit*



Stream Processor Grafico

# GPU

## *Graphic Processing Unit*



Vertex Unit



- ROTAZIONE
- TRASLAZIONE

Fragment Unit



- COLORE
- LUCE

## General-Purpose computation on GPUs

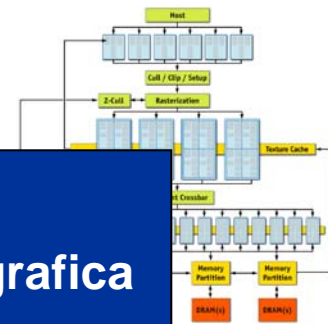
## Matrice A

0.8	0.5	0	1
1	0.6	1	0.7
0.5	1		
0	0		

## Input grafico



## Elaborazione

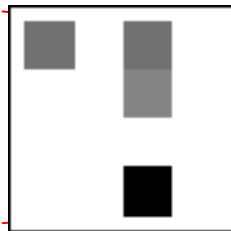


## applicazione matematica $\Rightarrow$ elaborazione grafica

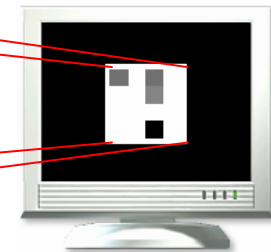
## Matrice C

0.5	1	0.5	1
1	1	0.7	1
1	1	1	1
1	1	0	1

## Lettura dal Framebuffer



## Output grafico



# Applicazione GPGPU



Apri una finestra



Carica le immagini come TEXTURE

Compila e carica sulla GPU i Kernel  
Definisci i parametri da passare



Attiva il Framebuffer Object

Disegna un rettangolo

Applica le TEXTURE

Renderizza  $K$  volte

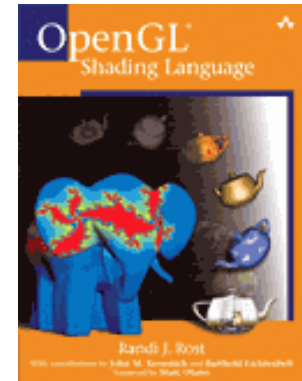
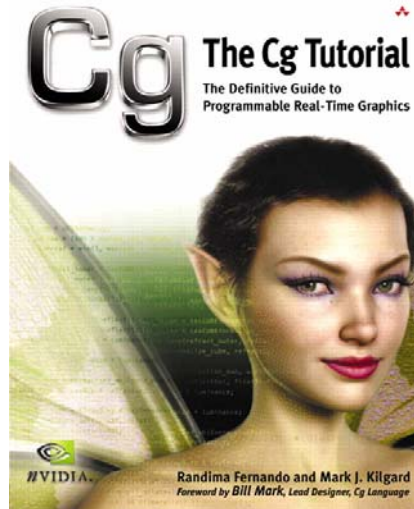
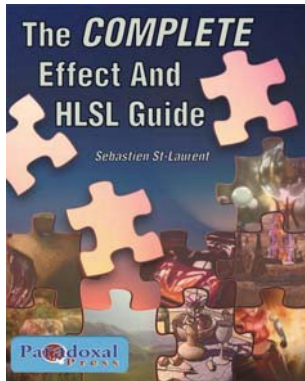
Leggi il Framebuffer Object

VERTEX PROGRAM



FRAGMENT PROGRAM

# Linguaggi di Shading



- ✓ Linguaggio sintatticamente simile al C
- ✓ Non è un linguaggio ad oggetti
- ✓ Non usa puntatori
- ✓ Non gestisce risorse esterne ( dischi, memoria centrale )

**Microsoft®**



# GPGPU

## *General-Purpose computation on GPUs*

### Prodotto ijk

$$C[i, j] += A[i, k] * B[k, j]$$

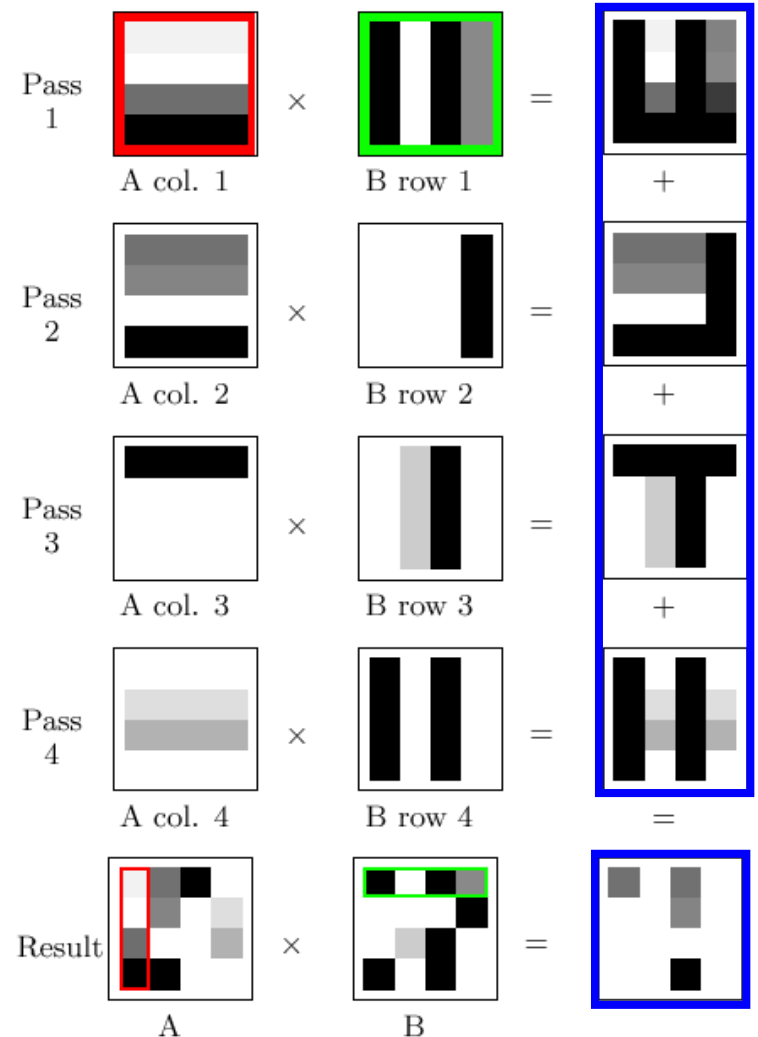
### GPGPU

Passo 1 -> k=1

$$A_k(i, j) = A(i, 1)$$

$$B_k(i, j) = B(1, j)$$

$$A_k(i, j) * B_k(i, j) = C_k(i, j)$$



# GPGPU

## *General-Purpose computation on GPUs*

### Fragment Program

```
texCoordA.y=texCoord.y;  
texCoordA.x=(k+0.5)/dimC;  
texCoordB.y=texCoordA.x;  
texCoordB.x=texCoord.x;
```

spalma K-esima colonna di A  
spalma K-esima riga di B

```
blockA=tex2D(cgA, texCoordA);  
blockB=tex2D(cgB, texCoordB);
```

leggi il colore di A e B

```
OUT.color = tex2D (Ck, texCoord)  
            +( blockA.xyzw * blockB.xwxw )  
            +( blockA.yxwz * blockB.zyzy );
```

Output Colore =  $C_k + (A \times B)$

# Hardware



4 a 3.00GHz  
1024 Kb di cache



*PCI Express Bus ( 16x )*  
*8 GB/sec*



**NVIDIA 7800 GT**  
**256Mb di RAM**

---

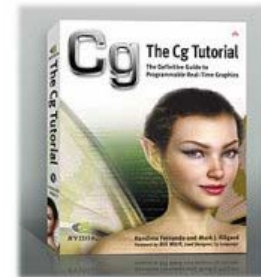
**gcc 3.4.4**



**OpenGL version 2.0**



**NVIDIA Cg Compiler**  
**Release 1.4**



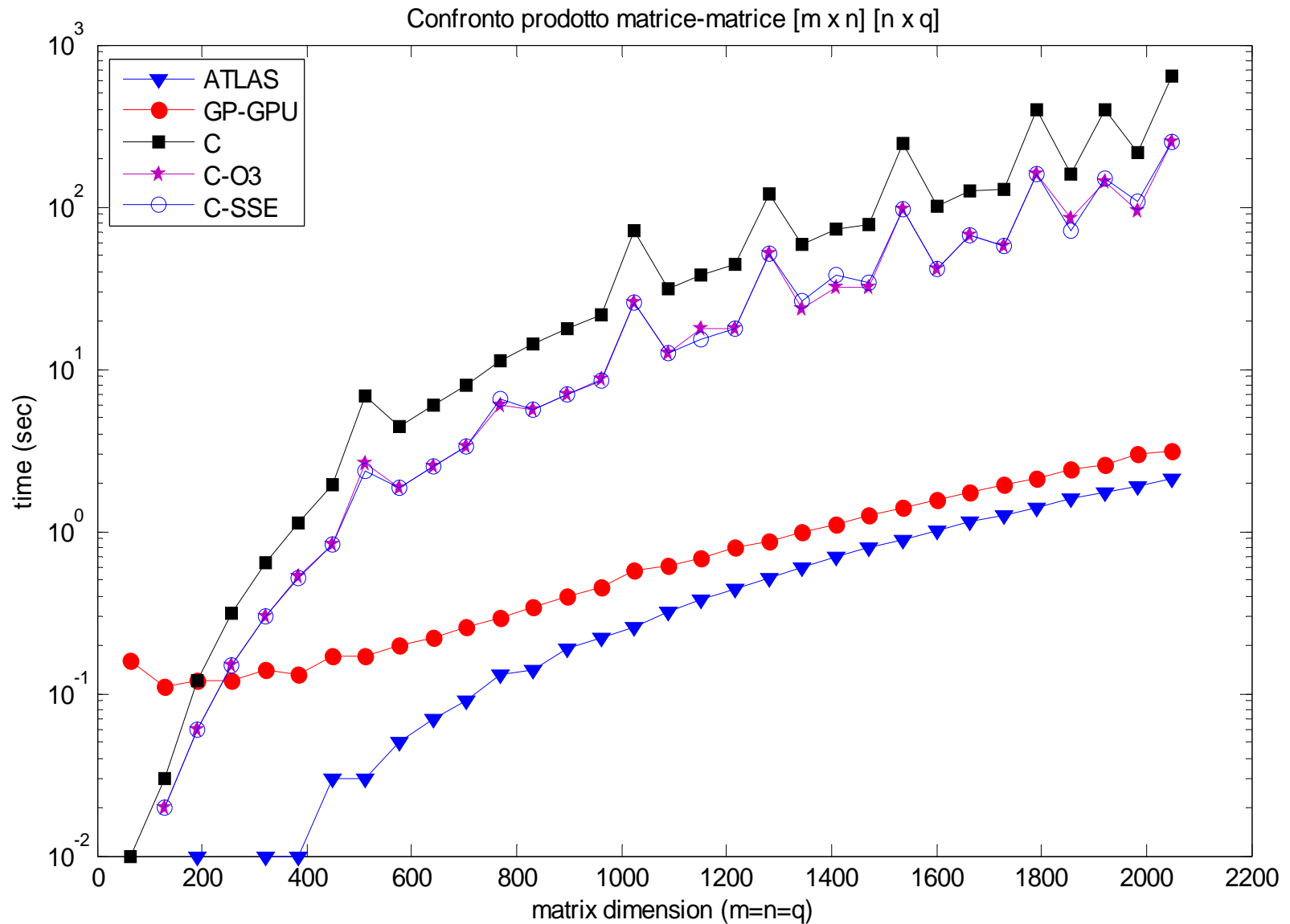
# Software



## **Matrici quadrate 2048x2048**

- IJK standard.
- IJK con compilazione ottimizzata O3
- IJK con supporto vettoriale SSE
- ATLAS

# VALUTAZIONE DELLE PRESTAZIONI



# ATLAS vs GPGPU

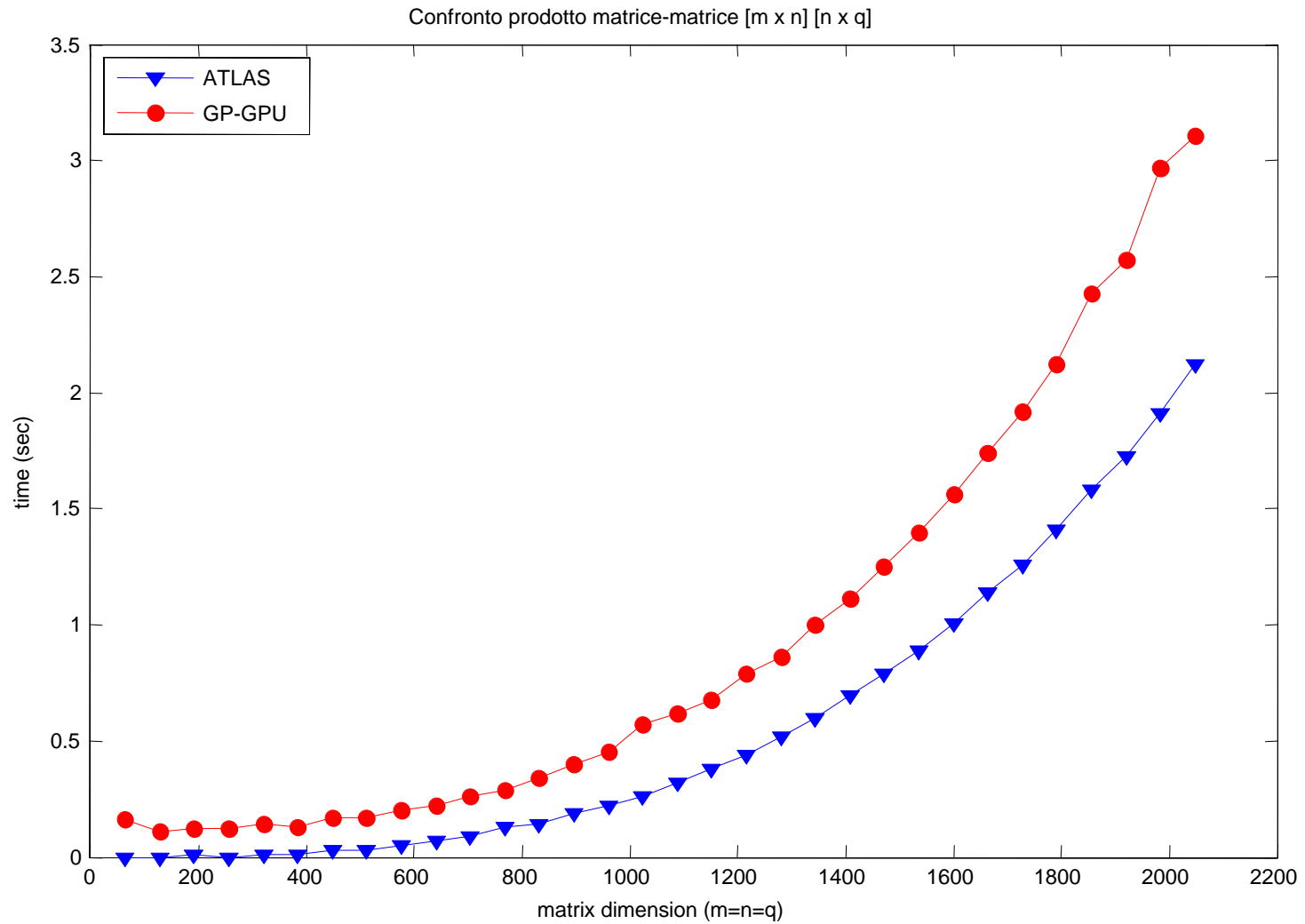
## **ATLAS**

- Set di funzioni fisso e limitato
- Migliore strumento disponibile
- Sviluppato da team mondiale
- Sviluppato in 4 anni

## **Matrix GPGPU**

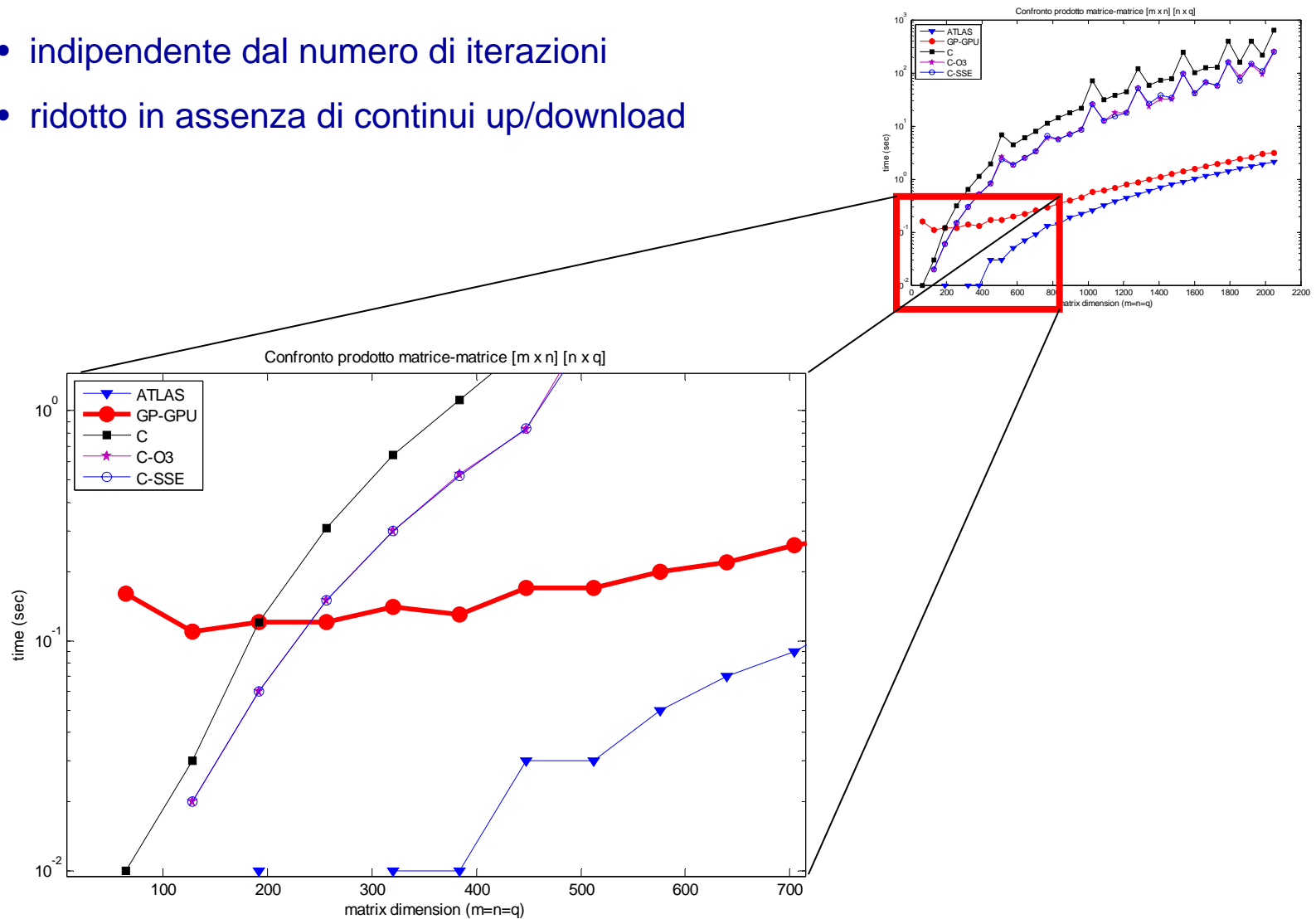
- Applicazione modificabile
- Nuovo approccio
- Sviluppato da me
- Sviluppato in 6 mesi

# ATLAS vs GPGPU

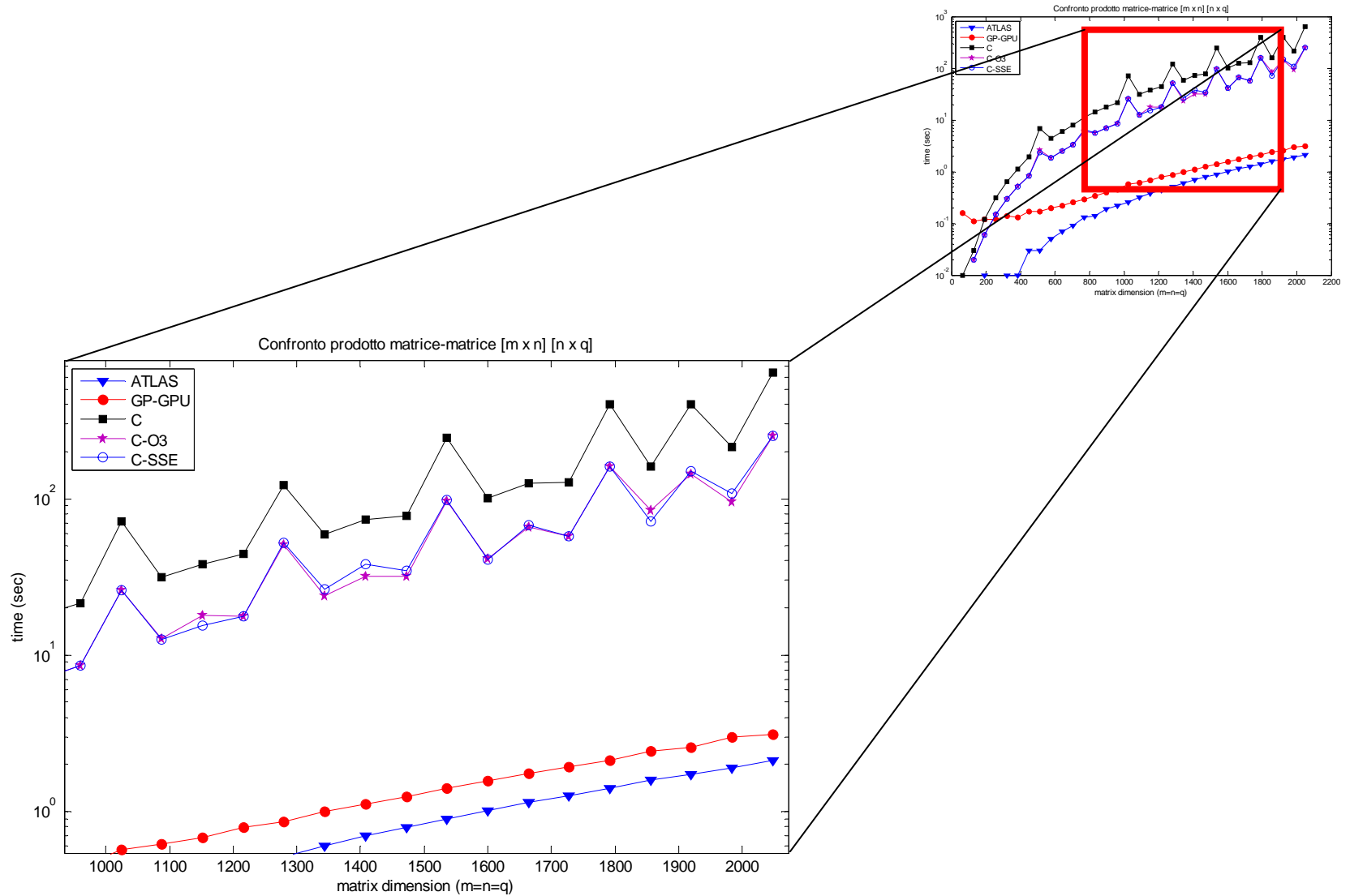


# TEMPO DI START-UP

- indipendente dal numero di iterazioni
- ridotto in assenza di continui up/download



# CACHE TRASHING



# GPU - TECNOLOGIA SCALABILE

0

	NVIDIA	NVIDIA	ATI
	7800 GT	7800 GTX	X1800XT
Processo produttivo	0.11 micron	0.11 micron	0.09 micron
Frequenza chip	400 Mhz	430 Mhz	625 Mhz
Frequenza memoria	1000 Mhz	1200 Mhz	1500 Mhz
Vertex Unit	7	8	8
Fragment Unit	20	24	16
Fill Rate	8000 MPixel	10320 MPixel	10000 MPixel
Banda Passante	32 GB	38,4 GB	48 GB

# Considerazioni

- ⇒ La tecnologia GPU è molto giovane.
- ⇒ Le GPU sono altamente scalabili.
- ⇒ Il mercato dei videogiochi 3D è ricco e con una forte domanda.
- ⇒ Si possono montare due GPU in parallelo su singola macchina.
- ⇒ La GPU può lavorare in parallelo alla CPU.



---

ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA  
SEDE DI CESENA  
FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI  
CORSO DI LAUREA IN SCIENZE DELL'INFORMAZIONE

**HIGH PERFORMANCE COMPUTING  
SU UNITA' GRAFICHE PROGRAMMABILI**

**Tesi di laurea in**  
Fisica Numerica

**Relatore**  
Prof. Renato Campanini

**Presentata da**  
Mauro De Carolis

**Co-relatore**  
Dott. Matteo Roffilli

Sessione III  
Anno Accademico 2004/2005

---