Università degli studi di Roma
Tor Vergata

# Risoluzione di PDE tramite metodi MultiGrid in ambiente CUDA

**Relatore**
Ing. Daniele Carnevale

**Candidato**
Claudio Pupparo

**Correlatore**
Prof. Sergio Galeani

# Introduzione

- Studio di algoritmi per la risoluzione di Equazioni Differenziali alle Derivate Parziali (**PDE**)
- Studio di librerie per la parallelizzazione e l'esecuzione di tali algoritmi su GPU (**CUDA**)
- Implementazione degli algoritmi risolutori su CPU e GPU (CUDA)
- Test prestazionali: confronto risultati ottenuti con GPU e CPU

# PDE

**P**artial **D**ifferential **E**quations

$$u_{xx} = f(x, y) \qquad (1)$$

Perchè sono importanti?

Vengono utilizzate per descrivere l'evoluzione di sistemi:

- Fisici (propagazione suono e calore, fluidodinamica...)
- Biologia ( dinamica delle popolazioni)
- Mercati Finanziari (previsioni di mercato)
- Meteorologia (previsioni del tempo)
- Computer Graphics

# PDE

Una descrizione completa di una PDE è fornita tramite:
Problemi dei valori al contorno (**Problemi di Cauchy**)

$$\begin{cases} \boxed{u_{xx} = f(x,y)} & \boxed{\Omega: \quad 0 < x < 1,\ 0 < y < 1} & \\ \boxed{u = 0} & \boxed{x \in \partial\Omega} & \text{Dirichlet} \\ \boxed{u_x = 0} & \boxed{x \in \partial\Omega} & \text{Neumann} \end{cases}$$

PDE
Dominio
Condizioni al contorno

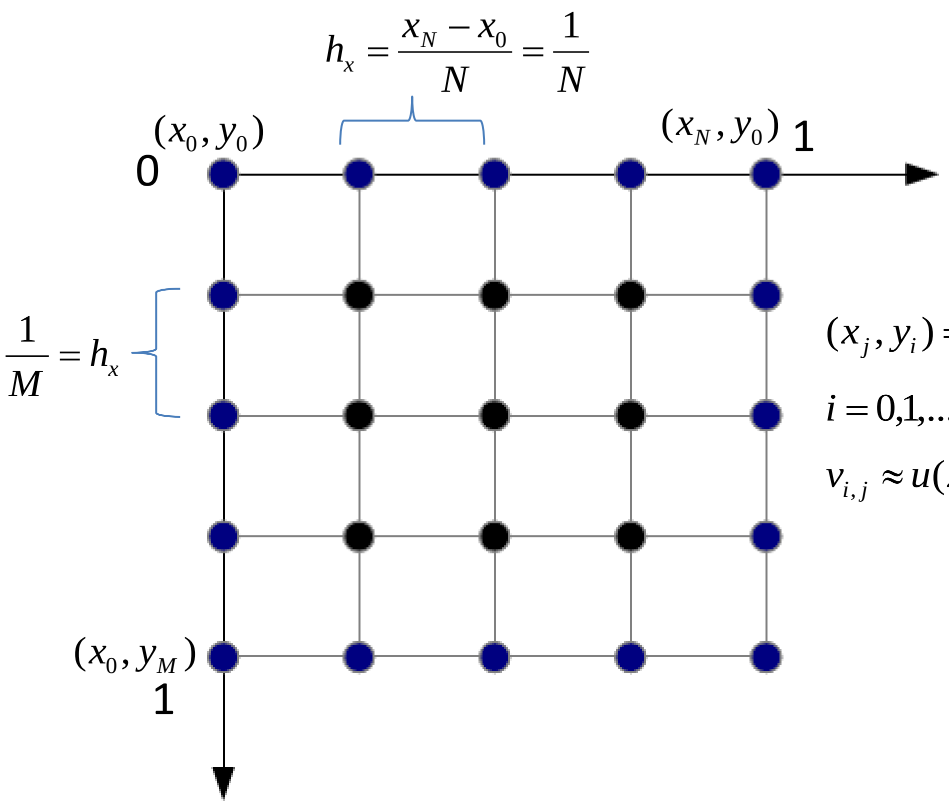
Per alcune PDE è molto difficile trovare una soluzione esplicita.

$$u(x,y) = ??$$

**Metodi Numerici**: forniscono il valore della soluzione in determinati punti del dominio.

# PDE – Metodi Numerici

**Discretizzazione**: Il dominio viene diviso in una **griglia** di punti




$$(x_j, y_i) = (x_0 + j \cdot h_x, y_0 + i \cdot h_y)$$

$$i = 0,1,\ldots M \qquad j = 0,1,\ldots N$$

$$v_{i,j} \approx u(x_j, y_i), \quad f_{i,j} = f(x_j, y_i)$$

- **N** numero di punti lungo asse x (escluso $x_0$)
- **M** numero di punti lungo asse y (escluso $y_0$)
- $\mathbf{h_x}$ intervallo fra due punti lungo asse x
- $\mathbf{h_y}$ intervallo fra due punti lungo asse y
- $\mathbf{v_{i,j}}$ soluzione approssimata in $(x_j, y_i)$
- $\mathbf{f_{i,j}}$ valore funziona nota in $(x_j, y_i)$

# PDE – Metodi Numerici

Come esprimere le derivate? **Espansione di Taylor**:

$$v_{i+1,j} = v_{i,j} + v'_{i,j} h_y + o(h_y^2) \longrightarrow v'_{i,j} = \frac{v_{i+1,j} - v_{i,j}}{h_y} + o(h_y) \qquad (2)$$

$$\left.\begin{aligned} v_{i+1,j} &= v_{i,j} + v'_{i,j} h_y + v''_{i,j} \frac{h_y^2}{2!} + v'''_{i,j} \frac{h_y^3}{3!} + o(h_y^4) \\ v_{i-1,j} &= v_{i,j} - v'_{i,j} h_y + v''_{i,j} \frac{h_y^2}{2!} - v'''_{i,j} \frac{h_y^3}{3!} + o(h_y^4) \end{aligned}\right\} v''_{i,j} = \frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{h_y^2} + o(h_y^2) \qquad (3)$$

$$u_{xx} = f(x,y) \longrightarrow \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{h_x^2} = f_{i,j} \qquad (4)$$

# PDE – Metodi Iterativi

Discretizzazione: $$\frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{h_x^2} = f_{i,j} \qquad (5)$$

Risolvendo rispetto a $v_{i,j}$ : $$v_{i,j} = \frac{v_{i,j+1} + v_{i,j-1} - h_x^2 \cdot f_{i,j}}{2} \qquad (6)$$

E' possibile sviluppare un'iterazione:
*Per ogni punto* ***interno*** *alla griglia applica il* ***Passo di Rilassamento*** *(6)*

Algoritmi Iterativi:

- **Jacobi**
- **Gauss-Seidel**
- **Red-Black Gauss-Seidel**

# PDE – Red Black Gauss Seidel

Ogni passo di iterazione

$$v_{i,j} = \frac{v_{i,j+1} + v_{i,j-1} - h_x^2 \cdot f_{i,j}}{2} \qquad (7)$$

dipende dagli immediati vicini:

- Punti pari dipendono unicamente da punti dispari
- Punti dispari dipendono unicamente da punti pari

**Red-Black Gauss-Seidel**:

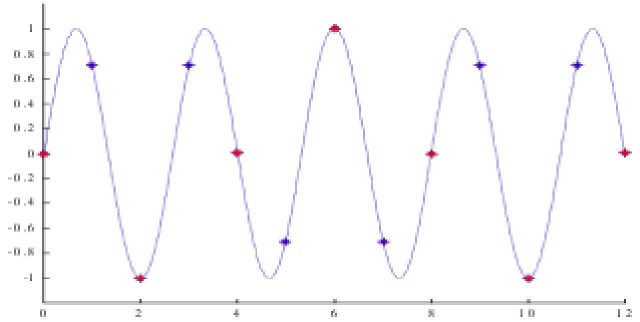
- Aggiorna prima tutti i punti pari
- Aggiorna dopo tutti i punti dispari

Tale metodo è **parallelizzabile**!

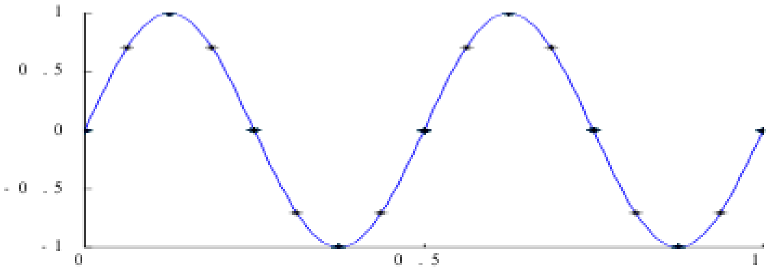
# PDE – Red Black Gauss Seidel

Efficienza RB Gauss-Seidel?

- Ottimo nell'eliminare la componente **oscillatoria** dell'errore
- Poco efficiente nell'eliminare la componente **smooth** dell'errore



Componente oscillatoria
(Alta frequenza)



Componente smooth
(Bassa frequenza)

# PDE – Metodi MultiGrid

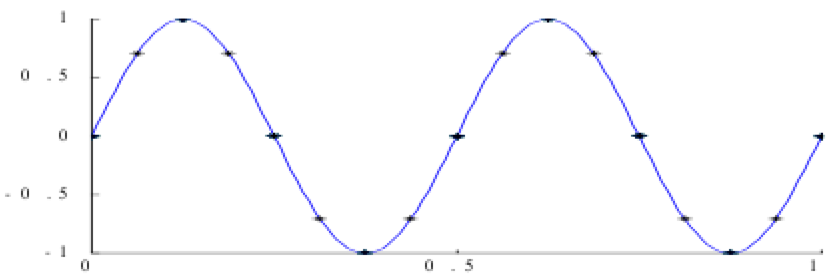
Come superare restrizioni RB Gauss-Seidel?

Introduzione di una gerarchia di Griglie:

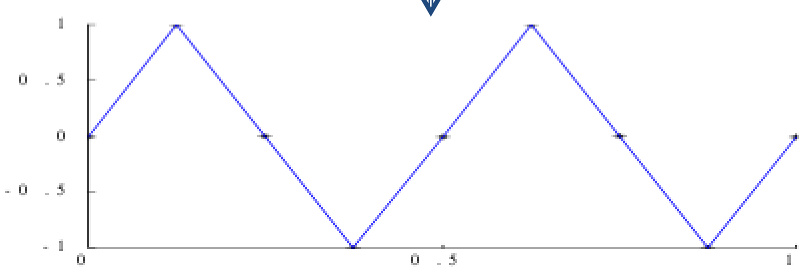
- Griglia **fine** - griglia iniziale
- Griglia **rada** - metà dei punti della griglia fine per asse

Perchè utilizzare una griglia rada?

Fine




Rada




Nel trasferimento di un'onda da una griglia fine ad una griglia rada le **componenti smooth diventano oscillatorie**!
(ottimo per RB Gauss-Seidel)

Alla base dei metodi MultiGrid

# PDE – Metodi MultiGrid

- Trasferimento Griglia Fine $\Longrightarrow$ Griglia Rada: **Restrizione**

$$I_h^{2h} v^h = v^{2h} \qquad (7)$$

- Trasferimento Griglia Rada $\Longrightarrow$ Griglia Fine: **Interpolazione**

$$I_{2h}^{h} v^{2h} = v^{h} \qquad (8)$$

Espressioni diverse a seconda delle dimensioni (1D, 2D, 3D)

# PDE – Metodi MultiGrid

Utilizzando griglie meno dense è possibile trovare una buona approssimazione dell'**errore**:

$$e = u - v \qquad (9)$$

Come?

Lavorando sull' **equazione dei residui** :

$$(10) \quad Au = A(v+e) = f \longrightarrow \boxed{Ae = f - Av = r} \qquad (11)$$

Risultato Importante:

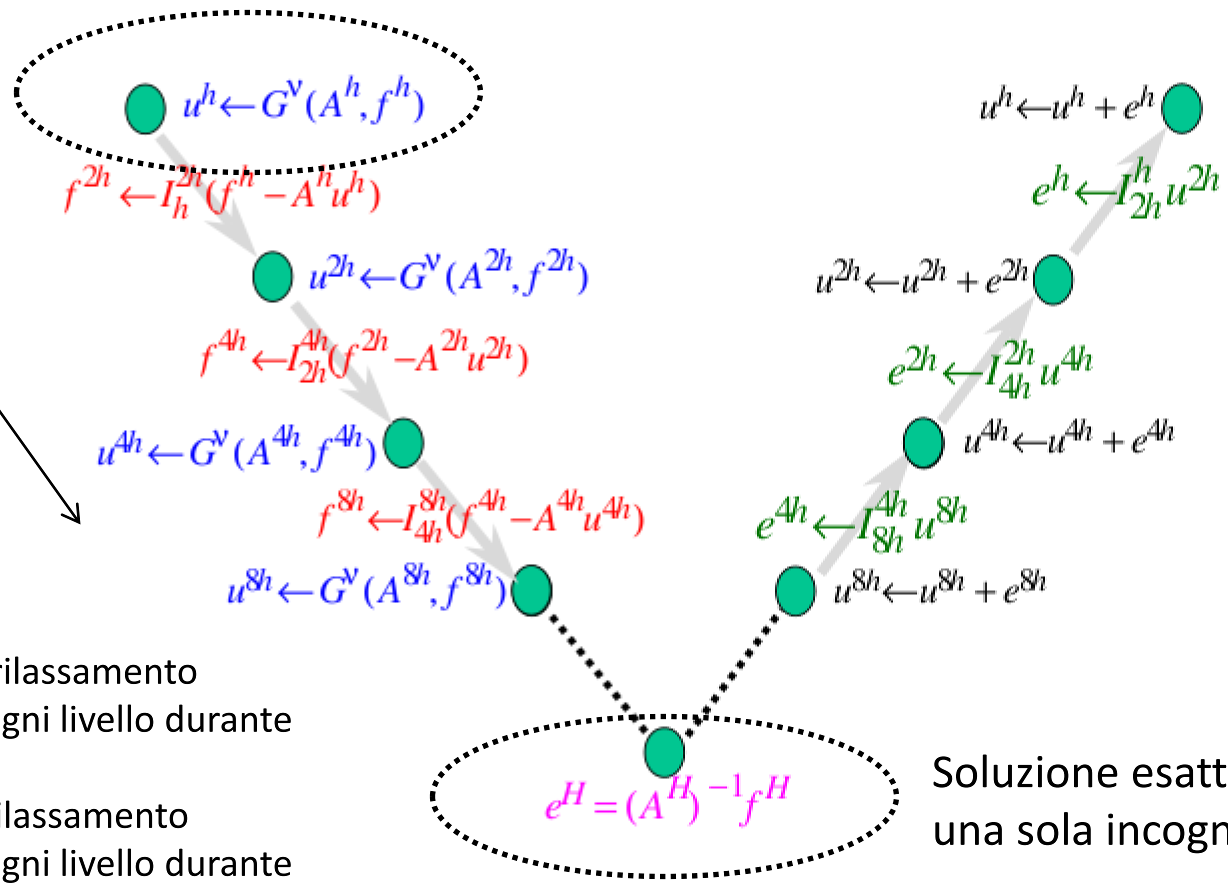
*Lavorare sull'equazione originale* $Au = f$ *con stima iniziale* $v$

*è equivalente a lavorare sull'equazione dei residui con stima iniziale* $e = 0$

# PDE – V Cycle

## Primo algoritmo MultiGrid: **V-Cycle**

Come ottenere una buona stima iniziale?




**Restrizione**

**Interpolazione**

Soluzione esatta: una sola incognita

Parametri:

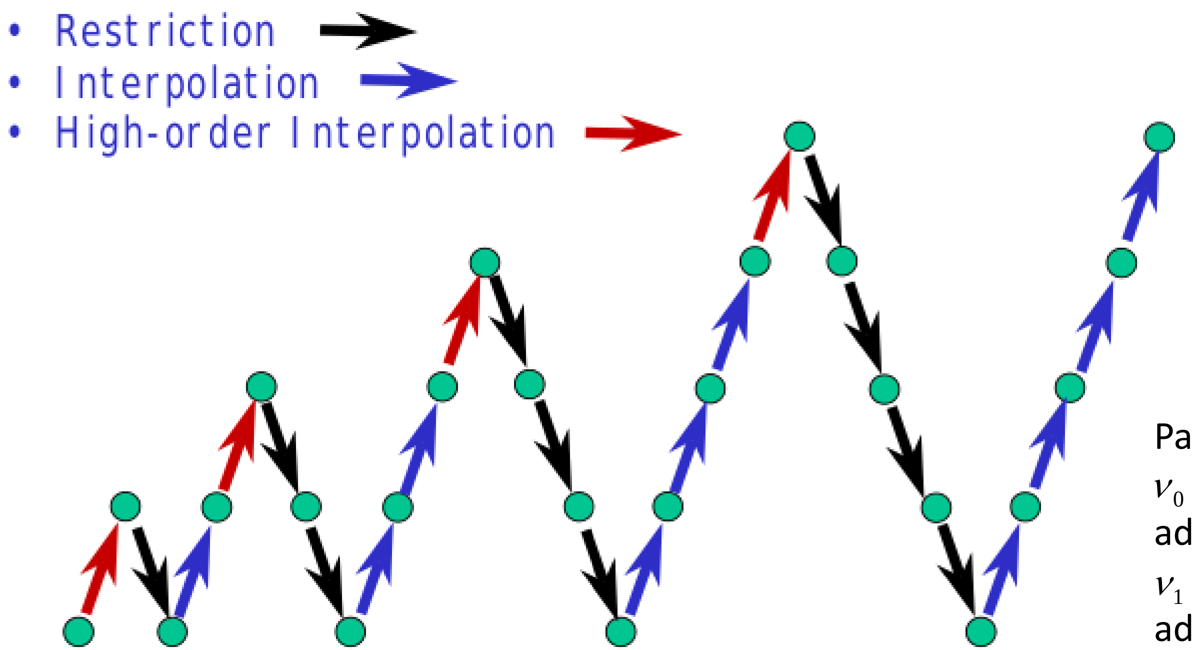
$\nu_1$ : passi di rilassamento eseguiti ad ogni livello durante la discesa

$\nu_2$ : passi di rilassamento eseguiti ad ogni livello durante la risalita

# PDE – Full MultiGrid V Cycle

Come ottenere una buona stima iniziale?

Partendo dalla griglia meno densa: **Full MultiGrid V-Cycle (FMG)**




FMG con $\nu_0$ = 1

Parametri:

$\nu_0$ : numero di V-Cycle da eseguire ad ogni livello

$\nu_1$ : passi di rilassamento eseguiti ad ogni livello durante la discesa

$\nu_2$ : passi di rilassamento eseguiti ad ogni livello durante la risalita

# PDE – Full MultiGrid V Cycle

Riepilogo:

- Full MultiGrid (**FMG**) V-Cycle è il miglior algoritmo MultiGrid per la risoluzione di PDE
- Il FMG richiede un elevatissimo numero di calcoli: ogni griglia deve essere visitata più volte per intero per aggiornare il valore della soluzione (**Collo di bottiglia**!)

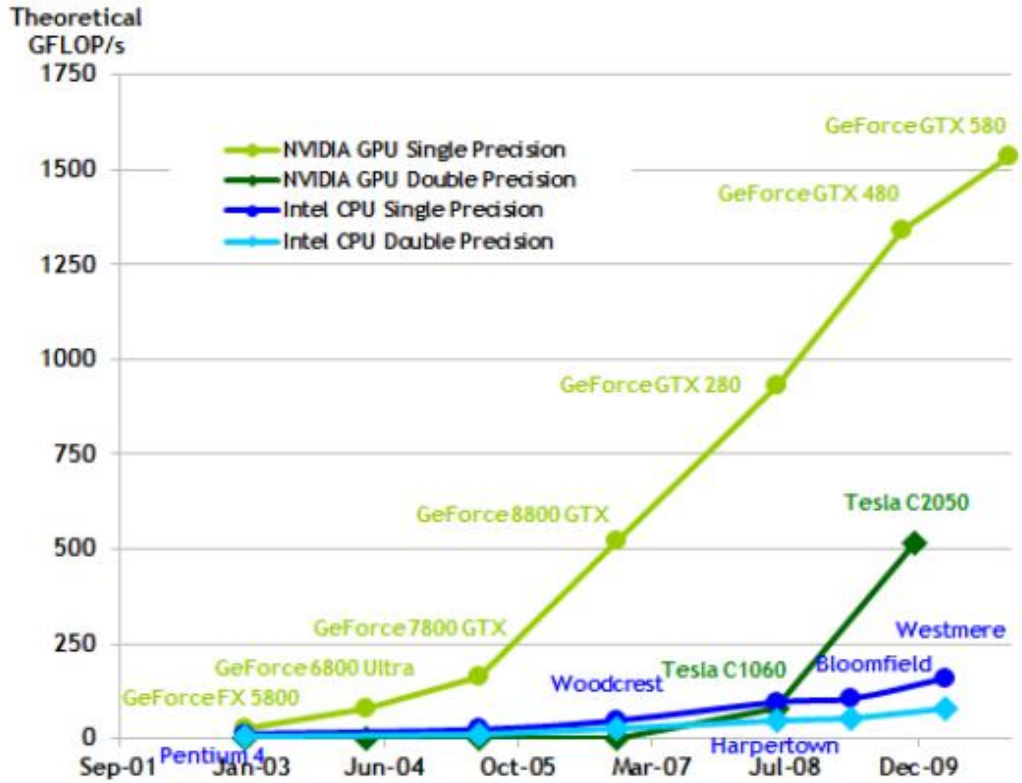
Soluzione: eseguire le iterazioni in parallelo $\longrightarrow$ **CUDA**

# CUDA

**C**ompute **U**nified **D**evice **A**rchitecture: paradigma di programmazione parallela improntato all'utilizzo della Gpu in domini diversi dal rendering grafico.

Perchè utilizzare la Gpu?

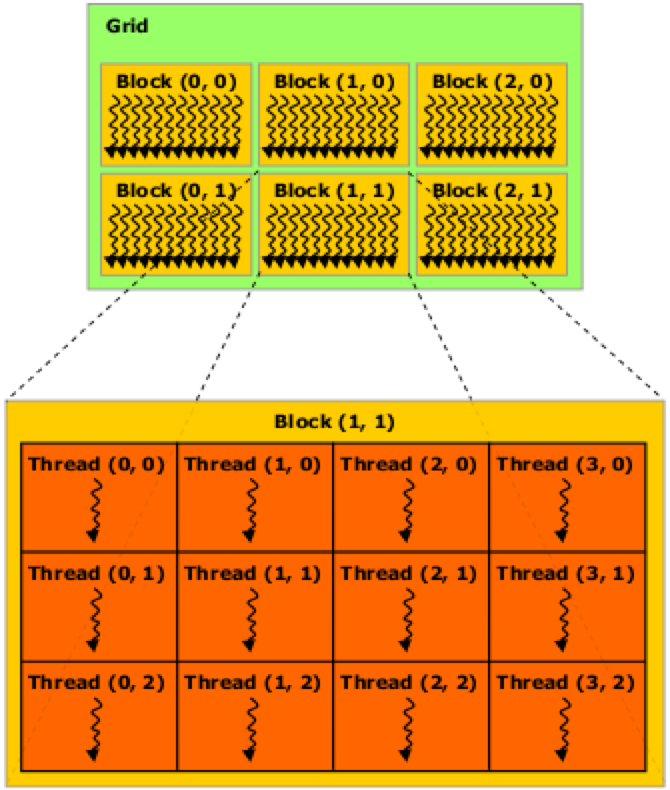
Si è evoluta come un’architettura fortemente parallela per via del suo usuale dominio applicativo: **Rendering 3D**.



# CUDA

Struttura applicazione Cuda (**Kernel**):

Un Kernel viene diviso in una **griglia** (fino a 2D) di **blocchi**, ogni blocco costituente un **vettore** (fino a 3D) di **thread**. Ogni elemento del vettore (**linearizzato**) da processare viene assegnato ad un thread.




La dimensione massima di un Kernel (2D) costituisce un limite per PDE in 3D: **è stato necessario sviluppare un algoritmo per la gestione dei thread all'interno dei blocchi in tale contesto.**

# Applicazioni

- **1D**: Equazione differenziale ordinaria

$$u' - \frac{u(x)}{e^x + 1} = e^x \tag{12}$$

- **2D**: Equazione di **Lyapunov**

$$\frac{\partial V}{\partial x_1}(a_{11}x_1 + a_{12}x_2) + \frac{\partial V}{\partial x_2}(a_{21}x_1 + a_{22}x_2) = -\alpha V \tag{13}$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$\alpha > 0$$

- **3D**: Equazione di **Poisson**

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = f(x, y, z) \tag{14}$$

# Applicazioni

**Piattaforma Hardware**:

- Processore: Pentium Dual Core Processor E5400: 2.7Ghz, 800 Mhz Fsb
- Memoria Ram: 2Gb
- Scheda video: Geforce GTX 550Ti 1Gb, 192 Cuda Cores

**Software realizzato**:

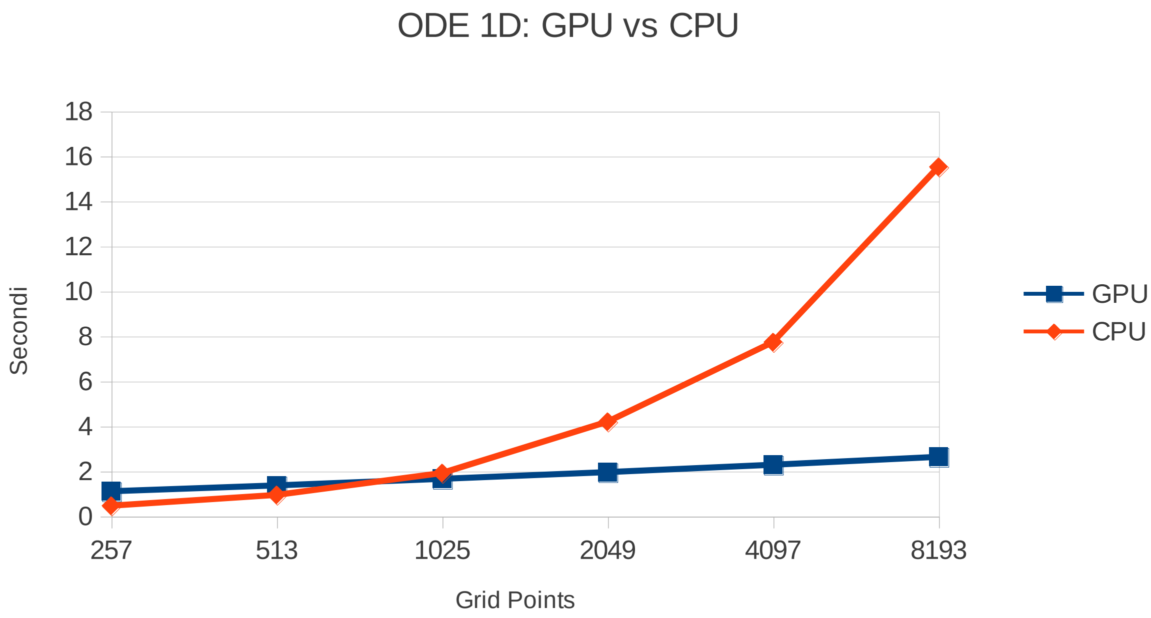
Implementazione di algoritmi MultiGrid con codice CPU e GPU (**CUDA**)

# Applicazioni – Gpu vs Cpu

- **1D**: Equazione differenziale ordinaria

$$u' - \frac{u(x)}{e^x + 1} = e^x \qquad (15)$$

$$v_j = \frac{v_{j+1}(e^{x_j}+1) - e^{x_j} h_x (e^{x_j}+1)}{e^{x_j} + 1 + h_x} \qquad (16)$$




$\Omega : [0,1]$

$v_0 = 2$

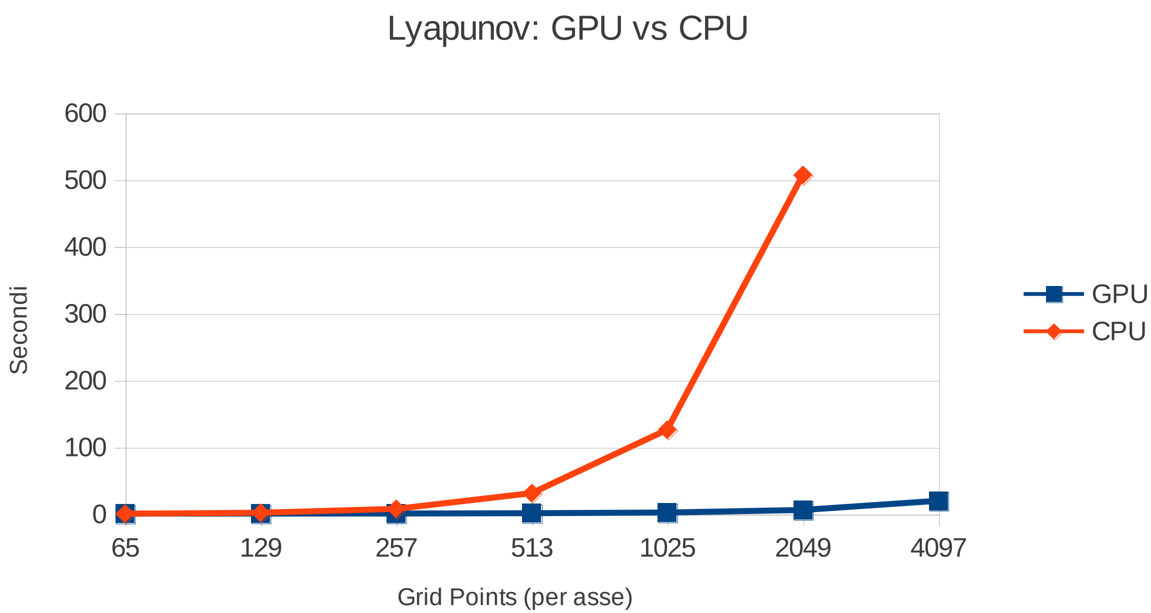
$v_1 = v_2 = 1000$

# Applicazioni – Gpu vs Cpu

- **2D**: Equazione di **Lyapunov**

$$\frac{\partial V}{\partial x_1}\overbrace{(-x_1+-2x_2)}^{K_1}+\frac{\partial V}{\partial x_2}\overbrace{(0\cdot x_1+-3x_2)}^{K_2}=-2\cdot V \quad (16)$$

$$v_{i,j}=\frac{u_{i,j+1}h_yK_1+v_{i+1,j}h_xK_2}{-2h_xh_y+K_1h_y+K_2h_x} \quad (17)$$



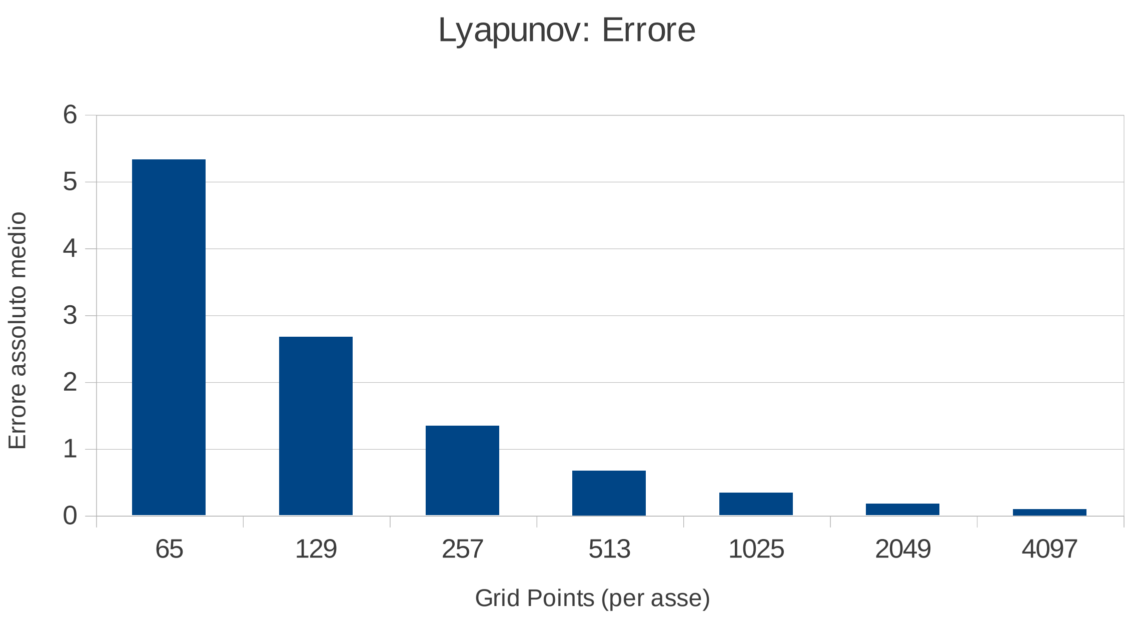

$$\Omega:[0,20]^2$$
$$\nu_0=2$$
$$\nu_1=\nu_2=500$$

# Applicazioni – Gpu vs Cpu

- **2D**: Equazione di **Lyapunov**

$$\frac{\partial V}{\partial x_1}\overbrace{(-x_1 + -2x_2)}^{K_1} + \frac{\partial V}{\partial x_2}\overbrace{(0\cdot x_1 + -3x_2)}^{K_2} = -2\cdot V \quad (16)$$

$$v_{i,j} = \frac{u_{i,j+1}h_y K_1 + v_{i+1,j}h_x K_2}{-2h_x h_y + K_1 h_y + K_2 h_x} \quad (17)$$



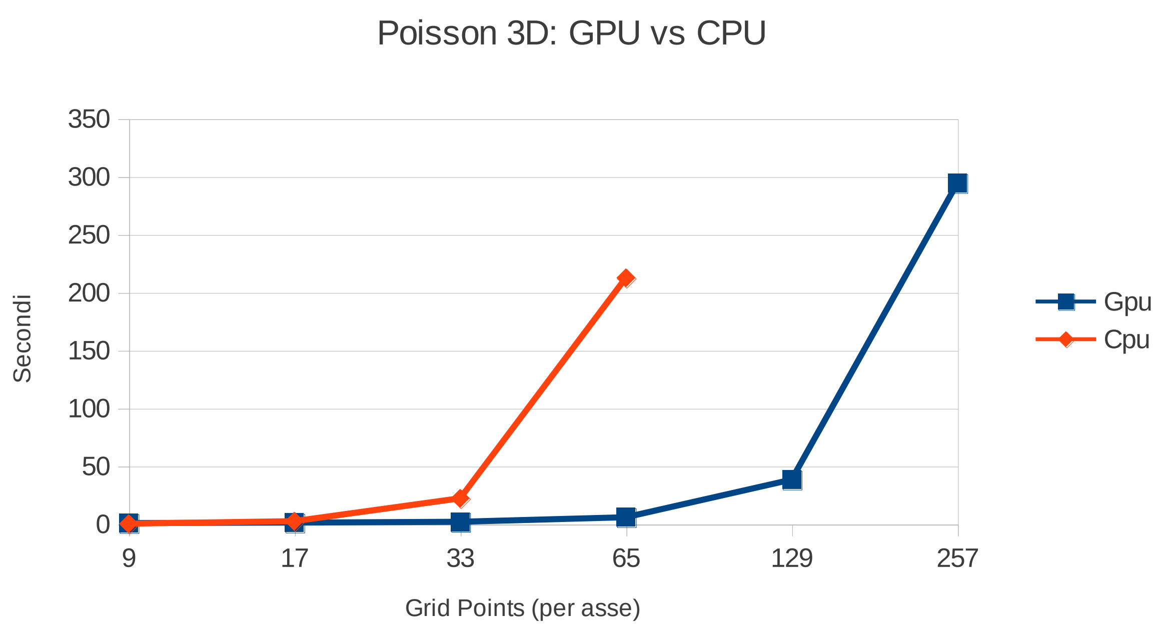

$$\Omega : [0, 20]^2$$
$$\nu_0 = 2$$
$$\nu_1 = \nu_2 = 500$$

# Applicazioni – Gpu vs Cpu

- **3D**: Equazione di **Poisson**

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -3\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z) \qquad (18)$$

$$v_{i,j,k} = \frac{(v_{i-1,j,k}h_x^2h_z^2 + v_{i+1,j,k}h_x^2h_z^2) + (v_{i,j-1,k}h_y^2h_z^2 + v_{i,j+1,k}h_y^2h_z^2) + (v_{i,j,k-1}h_y^2h_x^2 + v_{i,j,k-1}h_y^2h_x^2)}{2(h_x^2h_z^2 + h_y^2h_z^2 + h_y^2h_x^2)} \qquad (19)$$




$$\Omega : [0,1]^3$$
$$\nu_0 = 2$$
$$\nu_1 = \nu_2 = 3000$$

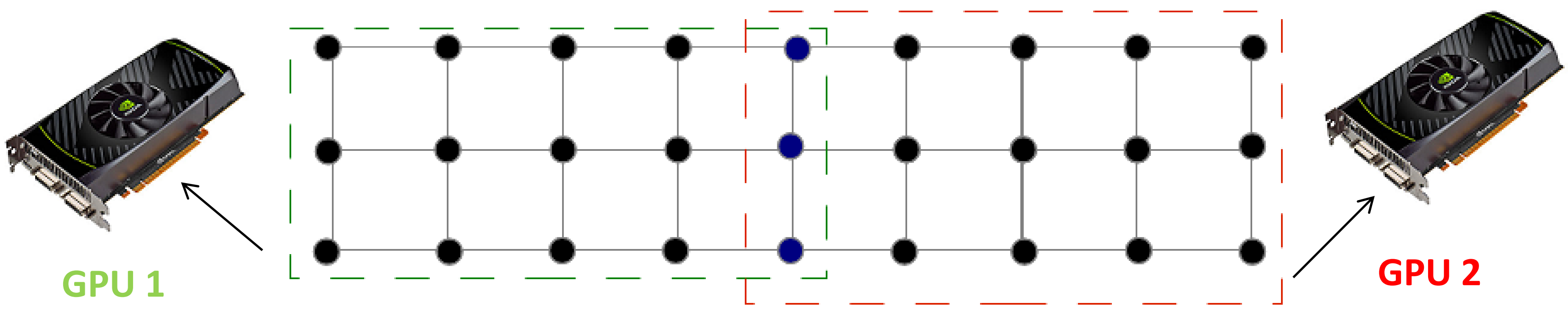
# Conclusioni

- Sono stati studiati metodi per la risoluzione di PDE
- Sono state studiate librerie per il calcolo parallelo su GPU: CUDA
- Gli algoritmi Risolutori sono stati implementati su Gpu tramite CUDA (e su Cpu per il confronto)
- Il Full MultiGrid V-Cycle costituisce un ottimo metodo per la risoluzione di PDE (il migliore dei metodi MultiGrid)
- Tramite l’utilizzo delle CUDA il tempo necessario ad eseguire il FMG è drasticamente ridotto

# Sviluppi futuri

E’ possibile fare di meglio?

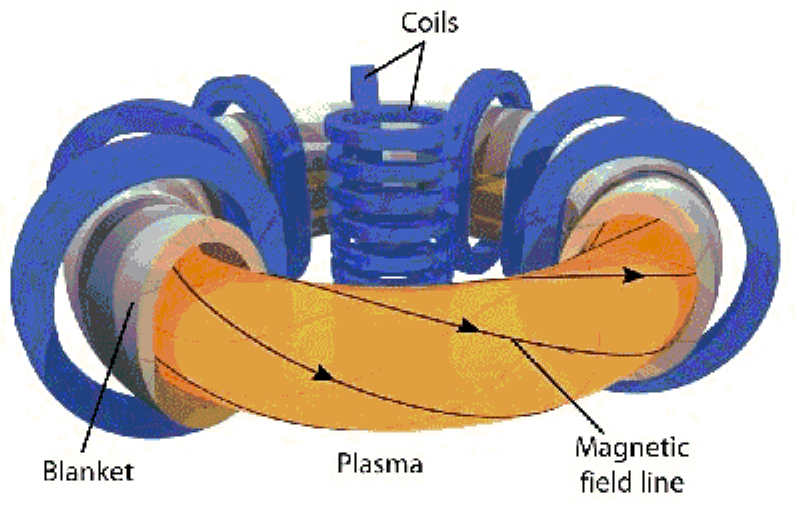
- Utilizzo di più Gpu in parallello: ogni griglia viene divisa in sottogriglie assegnate a Gpu diverse



# Sviluppi futuri

- Risoluzione di **Pde non Lineari**
- Equazione di **Grad Shafranov**

$$\Delta^{*}\psi = -\mu_0 R^2 \frac{dp}{d\psi} - \frac{1}{2}\frac{dF^2}{d\psi} \qquad (20)$$




Tokamak

**Magnetofluidodinamica**: dinamica dei fluidi elettricamente conduttori

Regola movimento del plasma all'interno di una forma toroidale (**Tokamak** – fusione termonucleare)