GPU solution of optimized wavenumber model for steady incompressible Navier-Stokes equations

Tony Wen-Hann Sheu

- Department of Engineering Science and Ocean Engineering, National Taiwan University
- Department of Mathematics, National Taiwan University
- Center for Advanced Studies on Theoretical Sciences (CASTS), National Taiwan University

CASTS-LJLL workshop on Applied Mathematics and Mathematical Sciences, Taipei, May 26-29, 2014

Acknowledgement : Neo Shih-Chao Kao, Wei-Chung Wang

Contents

I. Computational Challenges in solving 3D steady incompressible Navier-Stokes equations by finite element method

Mathematical model

Three-dimensional steady incompressible Navier-Stokes equations in primitive variables $(\underline{\mathbf{u}}, \mathbf{p})$ are as follows :

Elliptic Incompressible Navier-Stokes equations

$$\underset{\text{nonlinear convective term}}{\boxed{(\underline{\mathbf{u}} \cdot \nabla) \cdot \underline{\mathbf{u}}}} + \underset{\text{Pressure gradient}}{\boxed{\nabla \mathbf{p}}} = \underset{\text{Viscous term}}{\boxed{\frac{1}{\mathrm{Re}} \nabla^2 \underline{\mathbf{u}}}} + \underset{\text{Source term}}{\boxed{\underline{\mathbf{f}}(\underline{\mathbf{x}})}} \qquad (1)$$

$$\underset{\text{Divergence-free constrain condition}}{\boxed{\nabla \cdot \underline{\mathbf{u}} = 0}} \qquad (2)$$

$$\underline{\mathbf{u}}(\underline{\mathbf{x}}, \mathbf{t}) = \underline{\mathbf{u}}_{\Gamma} \quad \text{on} \quad \partial\Omega \qquad (3)$$

The Reynolds number $\mathrm{Re} = \rho u_{\mathrm{ref}} L_{\mathrm{ref}} / \mu$ comes out as the result of normalization

- The mixed FEM formulation is adopted to get the solution $(\underline{\mathbf{u}}, \mathbf{p})$ simultaneously

Claude-Louis Navier (1785-1836)

George Gabriel Stokes, (1819-1903)

Computational challenges in solving $(\underline{\mathbf{u}}, \mathbf{p})$ solutions from (1-3)-Motivation

C1 – Rigorous implementation of boundary condition and assurance of divergence-free constrain for the elliptic differential system of equations

C2 - Accurate approximation of convection term $\underline{\mathbf{u}} \cdot \nabla \underline{\mathbf{u}}$ in fixed grid to avoid oscillatory velocity field

C3- Proper storage of primitive variables $\underline{\mathbf{u}}$ and $\mathbf{p}$ to avoid even-odd pressure solution

C4- Effective calculation of $(\underline{\mathbf{u}}, \mathbf{p})$ from the unsymmetric and indefinite matrix equation using modern iterative solver

C5- Parallel implementation of finite element program

Resolving computational challenges C1~C5 -Objectives

O1- Adopt mixed formulation of equations cast in primitive variables

O2- Derive a streamline upwinding Petrov-Galerkin formulation featuring the numerical wavenumber error reducing property for the convection term

O3- Formulate finite element equations in weak sense in trilinear pressure /triquadratic element to satisfy the LBB compatibility condition

O4- Apply preconditioner to the normalized symmetric and positive definite matrix equations to ensure fast unconditional convergence for high Reynolds number simulation

O5- Implement CUDA programming finite element code in hybrid CPU-GPU (**G**raphic **P**rocess **U**nit) platform to largely reduce the simulation time

II. In-house developed finite element code

(Numerical Heat Transfer ; in press)

Tri-quadratic wavenumber optimized finite element model

The proposed Petrov-Galerkin finite element model is as follows

$$\int_\Omega (\underline{\mathbf{u}} \cdot \nabla \underline{\mathbf{u}}) \underline{\mathbf{w}} \mathrm{d}\Omega + \frac{1}{\mathrm{Re}} \int_\Omega \nabla \underline{\mathbf{u}} : \nabla \underline{\mathbf{w}} - \int_\Omega \mathbf{p} \nabla \cdot \underline{\mathbf{w}} \mathrm{d}\Omega = \int_\Omega \underline{\mathbf{f}}\ \underline{\mathbf{w}} \mathrm{d}\Omega$$

$$\int_\Omega (\nabla \cdot \underline{\mathbf{u}}) \mathbf{q} \mathrm{d}\Omega = 0$$

where $\underline{\mathrm{w}}$ and q are the **weighting functions** for the **momentum** and **continuity equations**, respectively

The weighted residual weak formulation for the convection term in one-dimensional

$$\sum_{e=1} \int_{\Omega^e} \Phi_{\mathbf{x}} \mathbf{w}_{\mathbf{i}} d\Omega^e$$

- Galerkin model : $\boxed{\mathbf{w}_{\mathbf{i}} = \mathbf{N}_{\mathbf{i}}}$
- Wavenumber optimized model : $\mathbf{w}_{\mathbf{i}} = \mathbf{N}_{\mathbf{i}} + \mathbf{B}_{\mathbf{i}}$

biased part

$$\left\{ \begin{array}{l} \mathbf{B}_{\mathbf{i}} = \tau u \dfrac{\partial \mathbf{N}_{\mathbf{i}}}{\partial \mathbf{x}} \\ \tau = \dfrac{\delta u \mathrm{H}}{2|u|^2} \end{array} \right.$$

Substituting the quadratic interpolation function into the above weak statement to derive the discrete equations at the center and corner nodes, respectively

The discretization equation for the convection term in a quadratic element is expressed as follows :

$$\Phi_x \approx \frac{1}{h}(a_1\Phi_{i-1} + a_2\Phi_i + a_3\Phi_{i+1})$$

$$\begin{cases} a_1 = -\frac{1}{2} - \delta \\ a_2 = 2\delta \\ a_3 = \frac{1}{2} + \delta \end{cases}$$

To minimize the numerical **wavenumber error**, the **Fourier transform** and its inverse

$$\tilde{\mathbf{\Phi}}(\alpha) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathbf{\Phi}(x)\exp(-\mathbf{i}\alpha\mathbf{x})\mathrm{d}\mathbf{x}$$

$$\mathbf{\Phi}(\mathbf{x}) = \int_{-\infty}^{\infty} \mathbf{\Phi}(\alpha)\exp(\mathbf{i}\alpha\mathbf{x})\mathrm{d}\alpha$$

are applied to derive the **numerical wavenumber**

$$\alpha \approx \frac{-\mathbf{i}}{h}\left[a_1\exp(-\mathbf{i}\alpha h) + a_2 + a_3\exp(\mathbf{i}\alpha h)\right]$$

- The **actual wavenumber** can be regarded as the right hand side of the numerical wavenumber

$$\tilde{\alpha} = \frac{-\mathbf{i}}{h}\Big[a_1 \exp(-\mathbf{i}\alpha h) + a_2 + a_3 \exp(\mathbf{i}\alpha h)\Big]$$

- We define the error function to minimize the discrepancy between α and $\tilde{\alpha}$

$$E = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} |\tilde{\alpha} - \alpha|^2 d(\alpha h) = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} |\tilde{\gamma} - \gamma|^2 \mathrm{d}\gamma$$

- The limiting condition $\frac{\partial E}{\partial a_3} = 0$ is enforced to get the expression

  of δ as $\tau = \frac{\delta u \mathrm{H}}{2|u|^2}$

$$\delta = \begin{cases} \dfrac{1}{2} & ;\ \text{at end node} \\ \dfrac{8-3\pi}{-22+6\pi} & ;\ \text{at center node} \end{cases}$$

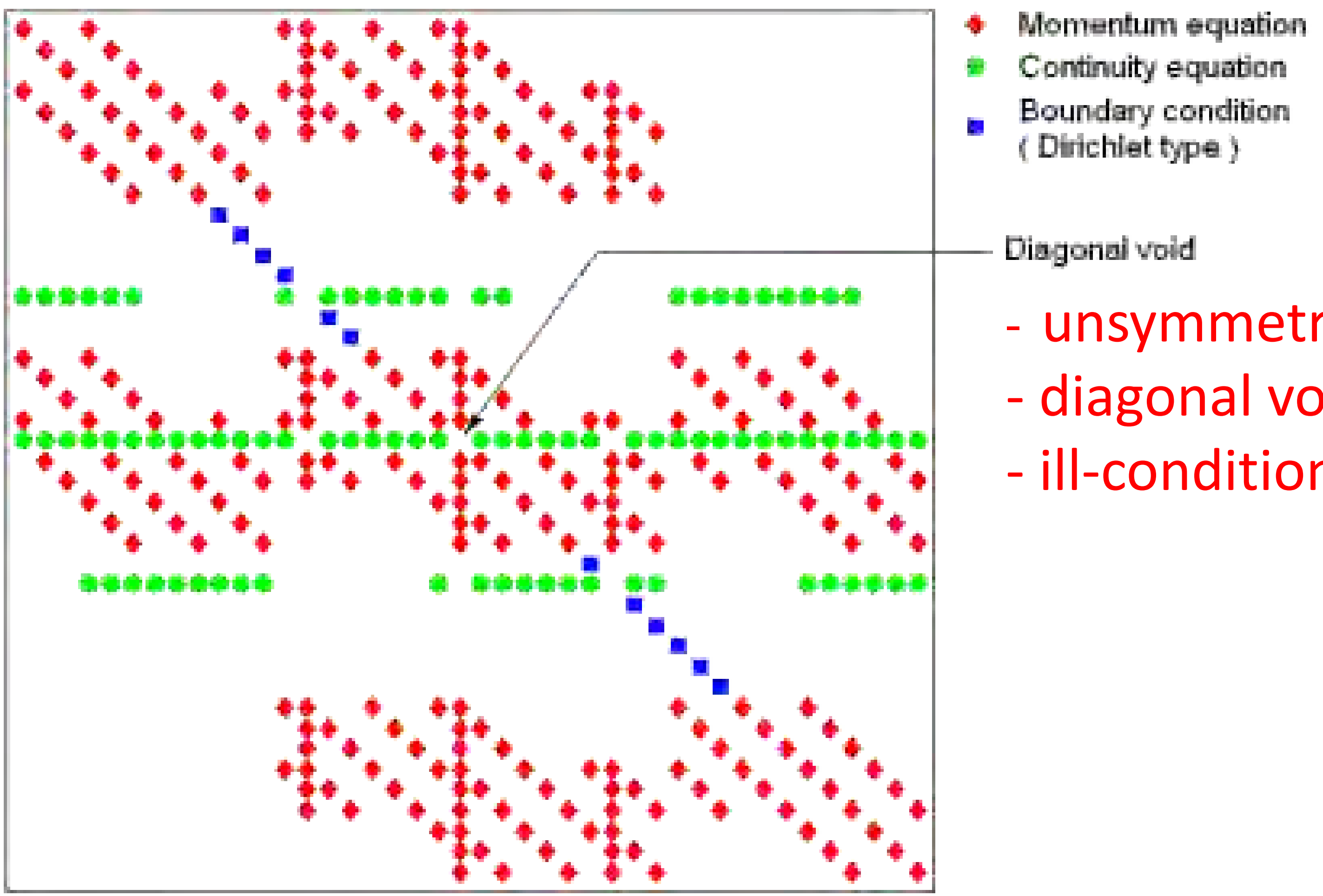
- In multi-dimensional problem, the method of **operator splitting** is adopted to get the stabilization term along the streamline direction

$$\tau = \frac{\delta_\xi \mathrm{V}_\xi \mathrm{H}_\xi + \delta_\eta \mathrm{V}_\eta \mathrm{H}_\eta + \delta_\zeta \mathrm{V}_\zeta \mathrm{H}_\zeta}{2\mathrm{V}_\mathrm{i}\mathrm{V}_\mathrm{j}}$$

$\mathrm{V}_{\mathrm{Y}_\mathrm{i}} = \hat{\mathrm{e}}_{\mathrm{Y}_\mathrm{i}} \cdot \mathbf{u}$, where $(\mathrm{Y}_1, \mathrm{Y}_2, \mathrm{Y}_3) = (\xi, \eta, \zeta)$

The features of matrix equation

For the smallest problem (2³ elements), the indefinite matrix equations takes the following profile



- unsymmetric
- diagonal void
- ill-conditioned

- The linear system derived from the FEM is usually written in the form

$$\underline{\underline{\mathbf{A}}}\ \underline{\mathbf{X}} = \underline{\mathbf{b}} \qquad \text{---------} \ (*)$$

where $\underline{\underline{\mathbf{A}}}$: Global matrix

$\underline{\mathbf{X}}$: Solution vector

$\underline{\mathbf{b}}$: Right hand side

- 90% of computation time of FEM is used to solve (*) by GMRES or BICGSTAB
- It is a grand challenge to solve (*) efficiently using iterative solution solver for three-dimensional large size problems => our proposed iterative solver given below

- To avoid <u>**Lanczos**</u> or <u>**pivoting breakdown**</u>, the linear system has been <u>**normalized**</u> , leading to the following positive definite and symmetric matrix equation

$$\underline{\underline{\tilde{A}}}\ \underline{X} = \underline{\tilde{b}},\ \text{where}\ \boxed{\underline{\underline{\tilde{A}}}} = \underline{\underline{A}}^{T}\underline{\underline{A}}, \underline{\tilde{b}} = \underline{\underline{A}}^{T}\ \underline{b}$$

symmetric and positive definite

- Since the normalized linear system becomes **symmetric** and **positive definite,** the unconditionally congergent **Conjugate Gradient (CG)** iterative solver is applied
- <u>**Preconditioning of matrix**</u> is adopted to reduce the corresponding increase of condition number
- Two techniques will be adopted to accelerate matrix calculation
    - The <u>**mesh coloring**</u> technique (To avoid the <u>race-condition</u>)
    - The <u>**EBE**</u> technique (To avoid <u>global matrix assembling</u>)

PCG algorithm

Algorithm The PCG iterative solver for $\underline{\underline{A}}^{T}\underline{\underline{A}}\,\underline{x} = \underline{\underline{A}}^{T}\,\underline{b}$

Starting from an initial guess $\underline{x}_0$

$\underline{\underline{B}}$: Jacobi preconditioner

Compute $\underline{x}'_0 = \sum_e \underline{\underline{A}}^e \underline{x}_0,\ \ \underline{x}''_0 = \sum_e (\underline{\underline{A}}^e)^T \underline{x}'_0,\ \ \underline{b}' = \sum_e (\underline{\underline{A}}^e)^T \underline{b}$ $\longleftarrow$ **EBE step**

Compute the initial residual $\underline{r}_0 = \underline{b}' - \underline{x}''_0$

$\underline{z}_0 = \underline{\underline{B}}^{-1}\underline{r}_0$

$\underline{p}_0 = \underline{z}_0$

For $j = 1, 2, \ldots,$

$\underline{p}'_{j-1} = \sum_e \underline{\underline{A}}^e \underline{p}_{j-1}$ $\longleftarrow$ **EBE step**

$\underline{p}''_{j-1} = \sum_e (\underline{\underline{A}}^e)^T \underline{p}'_{j-1})$ $\longleftarrow$ **EBE step**

$\alpha_{j-1} = (\underline{p}_{j-1}, \underline{r}_{j-1}) / (\underline{p}_{j-1}, \underline{p}''_{j-1})$

$\underline{x}_j = \underline{x}_{j-1} + \alpha_{j-1}\underline{p}_{j-1}$ $\longleftarrow$ **Update step**

$\underline{r}_j = \underline{r}_{j-1} - \alpha_{j-1}\underline{p}''_{j-1}$ $\longleftarrow$ **Update step**

Convergence check

$\underline{z}_j = \underline{\underline{B}}^{-1}\underline{r}_j$

$\beta_{j-1} = (\underline{z}_j, \underline{r}_j) / (\underline{r}_{j-1}, \underline{r}_{j-1})$

$\underline{p}_j = \underline{z}_j + \beta_{j-1}\underline{p}_{j-1}$ $\longleftarrow$ **Update step**

End

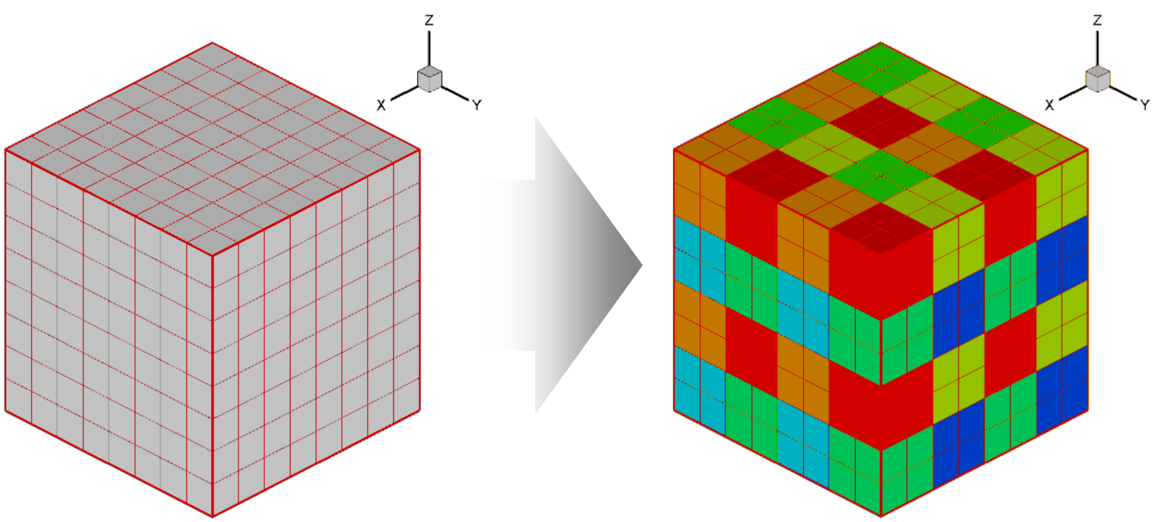
Mesh coloring technique

Mesh coloring algorithm

- In the "Update step" of PCG algorithm, provided that any two threads update the same vector component simultaneously, it will result in the so-called race-condition

(If two men ride on a horse, one must ride behind) "一山不容二虎"

- To avoid the race-condition, we divide the all elements into a finite number of subsets such that any two elements in a subset are not allowed to share the same node



* Different colors will be proceeded in sequence

* Elements with the same color are proceeded simultaneously and in parallel

EBE algorithm

In FEM, the global matrix $\underline{\underline{\mathbf{A}}}$ can be formulated as :

$$\underline{\underline{\mathbf{A}}} = \sum_{\mathrm{e}=1}^{\mathrm{Nel}} (\underline{\underline{\mathbf{B}}}^{\mathrm{e}})^{\mathrm{T}} \underline{\underline{\mathbf{A}}}^{\mathrm{e}} \underline{\underline{\mathbf{B}}}^{\mathrm{e}}, \quad \underline{\underline{\mathbf{B}}}^{\mathrm{e}} : \text{ Boolean matrix}$$

Underlying this concept, we have

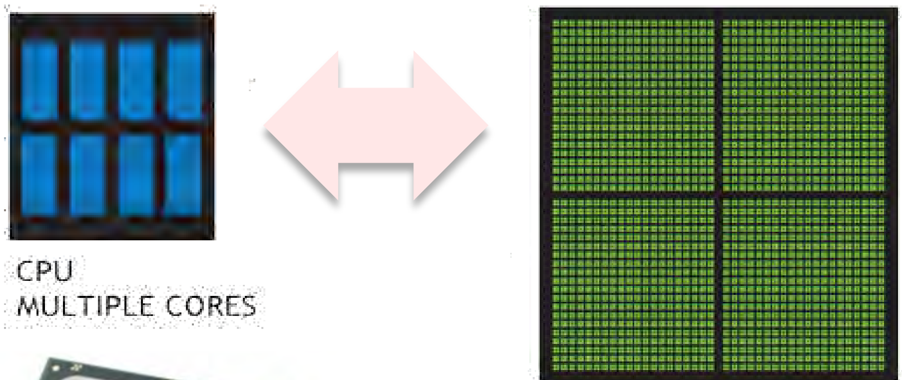
$$\underline{\underline{\mathbf{A}}}\,\underline{\mathbf{x}} = \left(\sum_{\mathrm{e}=1}^{\mathrm{Nel}} (\underline{\underline{\mathbf{B}}}^{\mathrm{e}})^{\mathrm{T}} \underline{\underline{\mathbf{A}}}^{\mathrm{e}} \underline{\underline{\mathbf{B}}}^{\mathrm{e}} \right) \underline{\mathbf{x}} = \sum_{\mathrm{e}=1}^{\mathrm{Nel}} (\underline{\underline{\mathbf{B}}}^{\mathrm{e}})^{\mathrm{T}} \left(\underline{\underline{\mathbf{A}}}^{\mathrm{e}} \underline{\mathbf{x}}^{\mathrm{e}} \right)$$

=> Time-consuming the matrix-vector product can be therefore implemented in an element-wise level

III. Code implementation in Kepler K20 GPUs

Introduction on CPU-GPU hybrid computing platform

- Computation is carried out on GPUs (**G**raphics **P**rocessing **U**nit) to get a good performance in scientific and engineering computing



CPU
MULTIPLE CORES

GPU
THOUSANDS OF CORES

* **Parallelism**

- CPU has 8 cores
- GPU has more than 1000 cores

* **Memory**

- CPU can reach 59 GB/s bandwidth
- GPU can reach 208 GB/s bandwidth

* **Performance**

- CPU can reach 59.2 GB **(double)**
- GPU can reach 3.52 TB/s **(single)**
  1.17 TB/s **(double)**

Intel i7 4820K
Processor ~ 3.7GHz

NVIDIA Tesla K series
Kepler K20
Processor clock rate = 705MHz
Bandwidth = 208 GB/s
Memory = 5 GB (DDR5)

http://www.nvidia.com.tw/object/what-is-gpu-computing-tw.html

CUDA programming

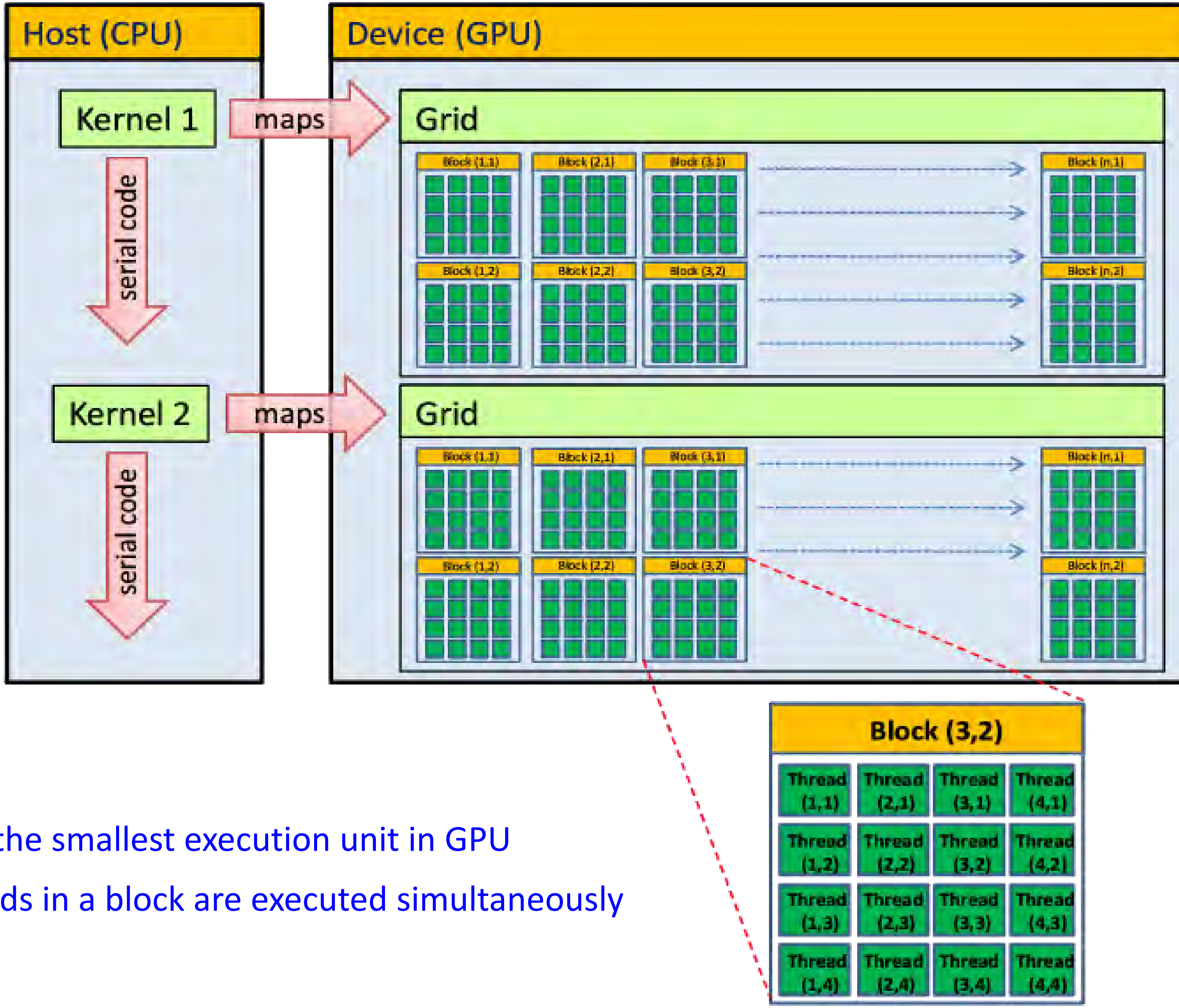
- (Compute Unified Device Architecture)
- CUDA is a general purpose parallel computational model released by NVIDIA in 2007
- CUDA only runs on NVIDIA GPUs
- Heterogeneous serial (CPU)-parallel (GPU) computing
- Scalable programming model
- CUDA permits Fortran (PGI) and C/C++ (NVIDIA NVCC) programming compliers

CUDA programming

- CPU **(host)** plays the role of "master" and GPUs **(device)** play the role of "workers"
  - CPU maps computational tasks onto GPUs
  - CPU can carry out computations at the same time when code is run on GPUs

- Basic flow chart
  - CPU transfers the data to the GPU
  - Parallel code (kernel) are run on the GPU
  - GPU transfers the computational results back to CPU

Current CPU-GPU hardware architecture

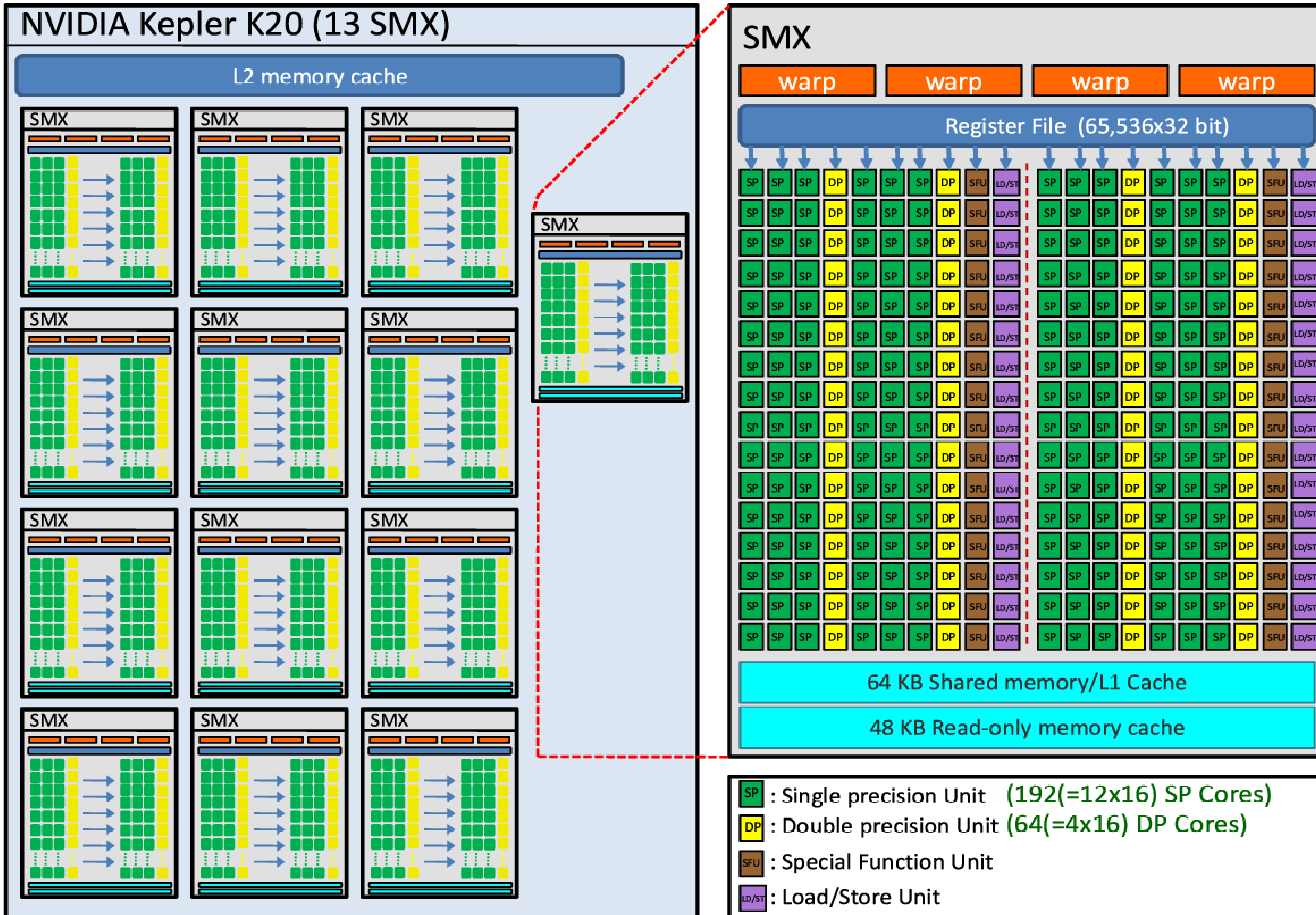


Note

1. Block is the smallest execution unit in GPU

2. All threads in a block are executed simultaneously

GPU Kepler K20 architecture



Classification of Memory in K20

1. Global memory (5GB DDR5) (**D**ouble **D**ata **R**ate)
2. Shared memory (48KB/SMX)
3. Cache memory (1536KB L_2)
4. Texture memory (48KB)

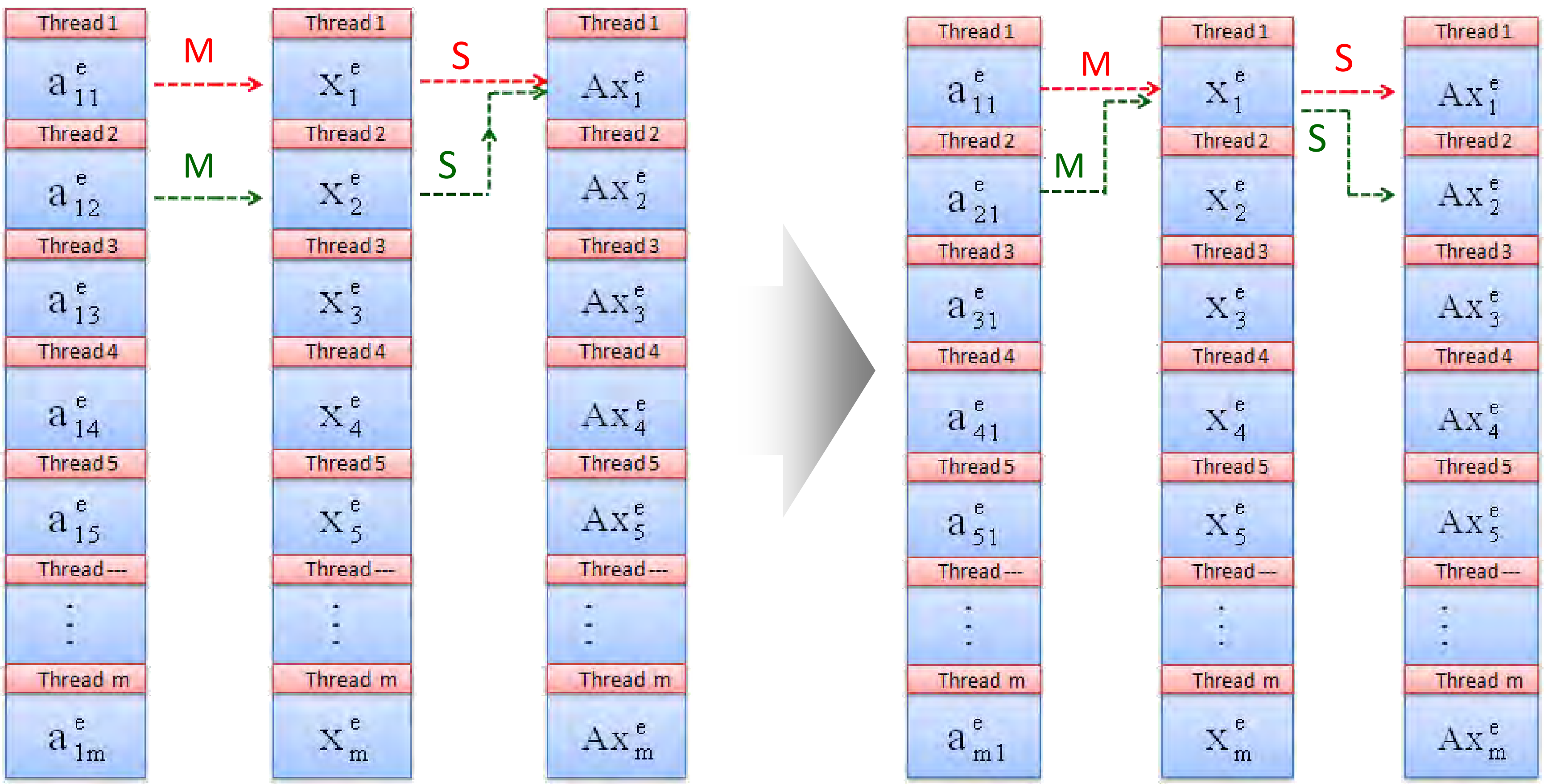
EBE implementation in GPU (1)

- To implement the EBE on GPU, the new formulation will be adopted. Given an e-th element matrix, RHS and its product are as follows

$$\underline{\underline{A}}^e = \begin{pmatrix} a_{11}^e & a_{12}^e & \cdots & \cdots & a_{1m}^e \\ a_{21}^e & a_{22}^e & \cdots & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \ddots \\ a_{m1}^e & & & & a_{mm}^e \end{pmatrix}, \underline{x}^e = \begin{pmatrix} x_1{}^e \\ x_2{}^e \\ \vdots \\ \vdots \\ x_m{}^e \end{pmatrix}, \underline{Ax}^e = \begin{pmatrix} Ax_1{}^e \\ Ax_2{}^e \\ \vdots \\ \vdots \\ Ax_m{}^e \end{pmatrix}$$

- One needs to compute the product $\underline{Ax}_1^e, \ldots, \underline{Ax}_m^e$, The resulting matrix-vector product results in a race-condition

EBE technique in GPU (2)



Race-condition algorithm

(Traditional)

Non-race-condition algorithm

(New operation)

Elements with the same color will be executed in parallel in a non-race-condition algorithm simultaneously

M : Multiplication operation

S : Sum operation

IV. Verification and numerical results

Verification study

- The analytic solutions which satisfy the steady-state 3D Navier-Stokes equations are given as follows :

$$\mathbf{u} = \frac{1}{2}(\mathbf{y}^2 + \mathbf{z}^2), \mathbf{v} = -\mathbf{z}, \mathbf{w} = \mathbf{y}$$

The corresponding exact pressure is given below

$$\mathbf{p} = \frac{1}{2}(\mathbf{y}^2 + \mathbf{z}^2) + \frac{2}{\mathrm{Re}}\mathbf{x}$$

- Problem setting
  - Re=1,000
  - Non uniform grid size : 11^3,21^3,41^3,61^3
  - Matrix Solver : PCG solver (iterative)

Verification study

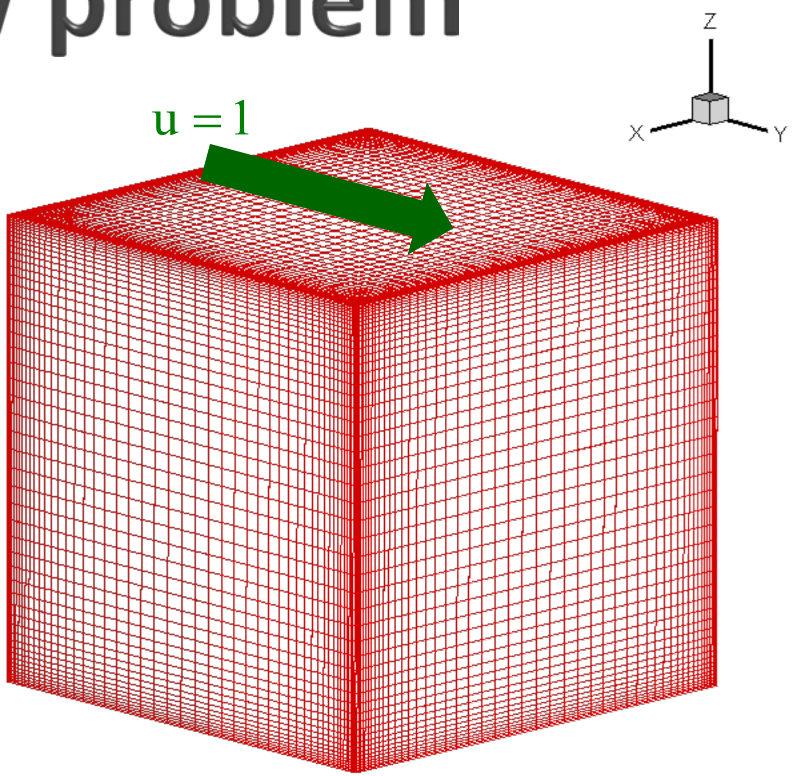
- The L_2 error norms are listed as follows :

	L2U	L2V	L2W	L2P
11^3	1.366×10^{-3}	2.835×10^{-3}	3.847×10^{-3}	6.186×10^{-3}
21^3	1.420×10^{-4}	3.207×10^{-4}	5.606×10^{-4}	1.713×10^{-3}
41^3	3.485×10^{-5}	7.357×10^{-5}	1.120×10^{-4}	4.378×10^{-4}
61^3	1.861×10^{-5}	3.462×10^{-5}	5.040×10^{-5}	1.683×10^{-4}
R.O.C.	2.26	2.32	2.30	1.90

- R.O.C. (**R**ate **O**f **C**onvergence)
- The predicted solutions show good agreement with the exact solutions

Lid-driven cavity flow problem

- Problem domain : $\Omega = [0,1]\times[0,1]\times[0,1]$
- Boundary condition
  - u=1,v=w=0 at upper plane
  - Non-slip at other planes
- Non-uniform grid number : 51^3
- Re=100,400,1000,2000
- Jacobi preconditioner is adopted
- Newton-linearization is adopted



Schematic of the lid-driven cavity flow problem

- D. C. Lo, K. Murugesan, D. L. Young, Numerical solution of three-dimensional velocity Navier-Stokes equations by finite difference method, *International Journal for numerical methods in Fluids,**47**,1469-1487,2005*

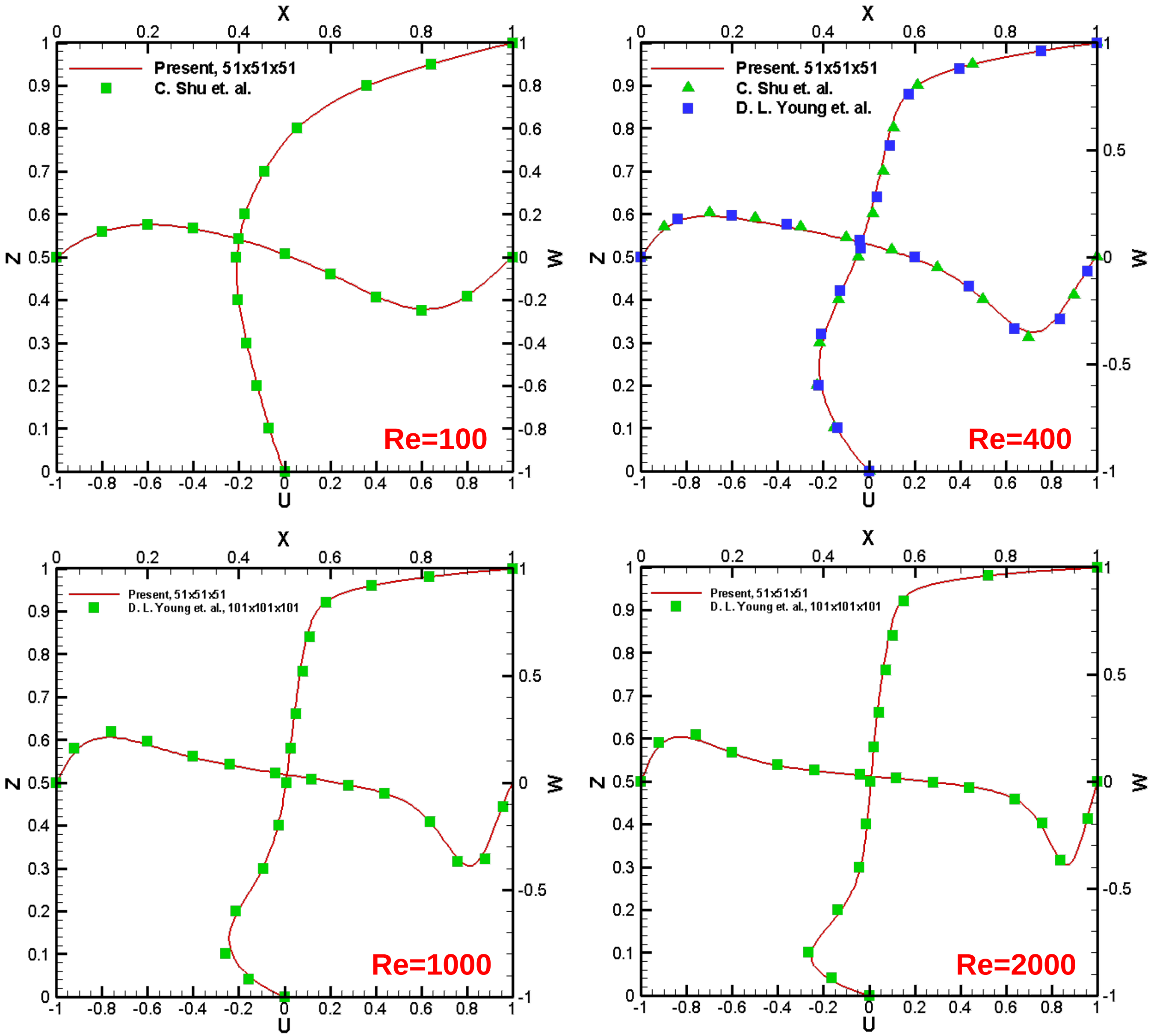
- C. Shu, L. Wang and Y. T. Chew Numerical computation of three-dimensional incompressible Navier–Stokes equations in primitive variable form by DQ method*, International Journal for Numerical methods in Fluids, **43** :345–368, 2005.*

Velocity comparison

Compare the velocity profiles

u(0.5,y,0.5)

w(x,0.5,0.5)



Performance study

- Computations are implemented on the following two different platforms

Platform #1 specification	Platform #2 specification
Single CPU platform CPU : Intel i7-4820K GPU : Non	Hybrid CPU/GPU platform CPU : Intel i7-4820K GPU : NVIDIA Kepler K20

- The lid-driven cavity problem at different Reynolds and grid numbers is used to access the performance of the current hybrid CPU/GPU calculation

Performance study

- Grid number = 31^3

Platform	Re=100	Re=400	Re=600	Re=800	Re=1000
Platform # 1 (a)	416.5	860.4	1292.8	1769.5	2437.2
Platform # 2 (b)	4555.4	10087.9	15075.5	21515.7	29836.2
Speedup (b)/(a)	10.93	11.72	11.66	12.15	12.24

- Grid number = 41^3

Platform	Re=100	Re=400	Re=600	Re=800	Re=1000
Platform # 1 (a)	1377.1	2599.9	3424.1	4687.4	6502.3
Platform # 2 (b)	19823.2	36639.0	54635.3	65215.7	91780.6
Speedup (b)/(a)	14.3	14.0	15.9	13.9	14.11

Performance study

- Grid number = 51^3

Platform	Re=100	Re=400	Re=600	Re=800	Re=1000
Platform # 1 (a)	3433.0	5983.7	7113.0	9606.1	12487.2
Platform # 2 (b)	50782.0	91012.0	113921.9	144358.3	196173.9
Speedup (b)/(a)	14.79	15.20	16.01	15.02	15.71

- Grid number = 61^3

Platform	Re=100	Re=400	Re=600	Re=800	Re=1000
Platform # 1 (a)	7702.8	11598.7	13939.7	17684.7	22222.5
Platform # 2 (b)	111338.8	168420.8	202867.8	256204.9	331559.7
Speedup (b)/(a)	14.45	14.52	14.55	14.48	14.92

IV. Concluding remarks

Concluding remarks

- A new streamline upwind FEM model accommodating the optimized numerical wavenumber along the streamline has been developed
- The unconditionally convergent finite element solution can be iteratively obtained from the normalized symmetric and positive definite matrix equation using the PCG solver
- Novel non-race-condition and EBE techniques have been successfully implemented on GPU architecture
- Computational time has been reduced more that ten times in a hybrid CPU/GPU platform

Thank for your attention !
Q/A ?