GPU TECHNOLOGY CONFERENCE
MARCH 17-20, 2015 SILICON VALLEY

NVIDIA

Speeding up a Finite Element Computation on GPU

Nelson Inoue

PUC RIO
GTEP

Summary

- Introduction
- Finite element implementation on GPU
- Results
- Conclusions

University and Researchers

- Pontifical Catholic University of Rio de Janeiro – **PUC- Rio**
- Group of Technology in Petroleum Engineering - **GTEP**
- Research Team

PhD Sergio Fontoura
Leader Researcher

PhD Nelson Inoue
Senior Researcher

PhD Carlos Emmanuel
Researcher

MSc Guilherme Righetto
Researcher

MSc Rafael Albuquerque
Researcher

Introduction

- Research & Development (R&D) project with Petrobras
- The project began in 2010
- The subject of the project is Reservoir Geomechanics
- There are great interest by oil and gas industry in this subject
- This subject is still little researched

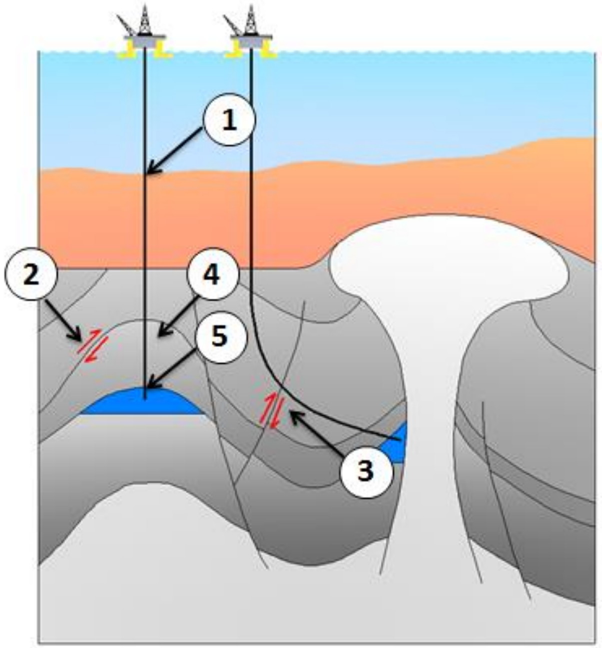
Introduction

- What is Reservoir Geomechanics?
  - Branch of the petroleum engineering that studies the coupling between the problems of **fluid flow** and **rock deformation** (stress analysis)
- Hydromechanical Coupling
  - Oil production causes rock deformation
  - Rock deformation contributes to oil production

Motivation

- Geomechanical effects during reservoir production

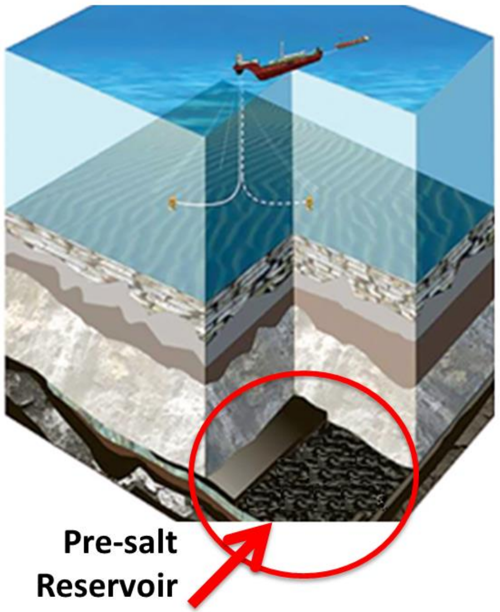
  1. Surface subsidence
  2. Bedding-parallel slip
  3. Fault reactivation
  4. Caprock integrity
  5. Reservoir compaction



Challenge

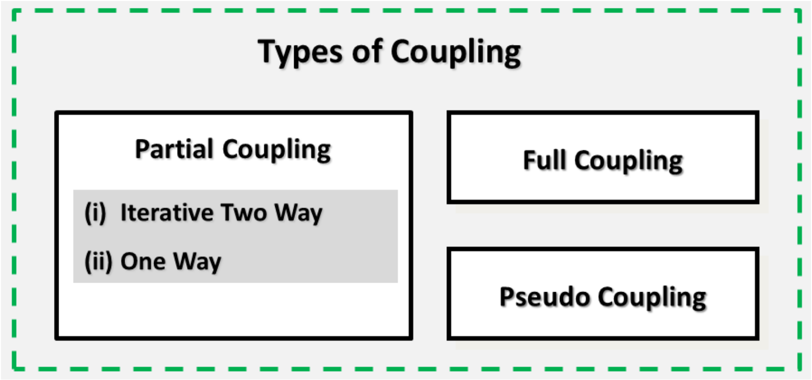
- Evaluate geomechanical effects in a real reservoir
- Overcome two major challenges
  1. To use a reliable coupling scheme between fluid flow and stress analysis
  2. **To speed up the stress analysis (Finite Element Method)**

     Finite Element Analysis spends most part of the simulation time

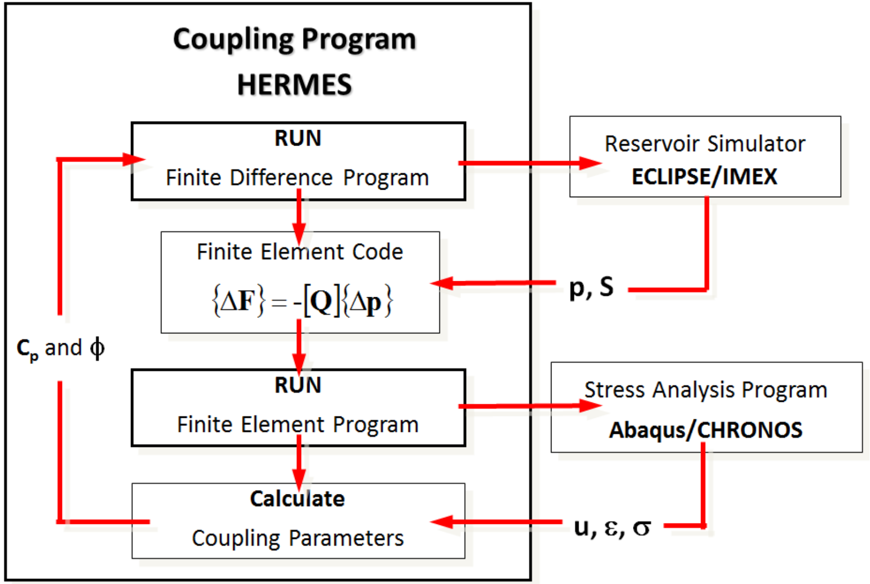


Hydromechanical coupling

- Theoretical Approach



Coupling program flowchart

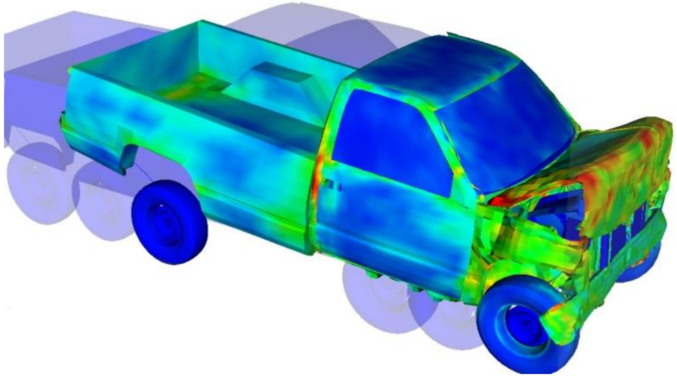


Finite Element Method

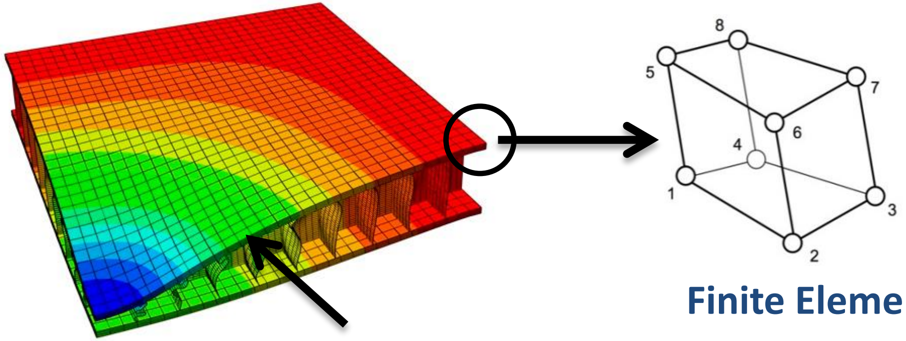
- **Partial Differential Equations** arise in the mathematical modelling of many engineering problems
- Analytical solution or exact solution is very complicated
- Alternative: **Numerical Solution**
  - **Finite element method**, finite difference method, finite volume method, boundary element method, discrete element method, etc.

Finite Element Method

- Finite element method (FEM) is widely applied in stress analysis
- The domain is an assembly of finite elements (FEs)



(http://www.mscsoftware.com/product/dytran)



Finite Element

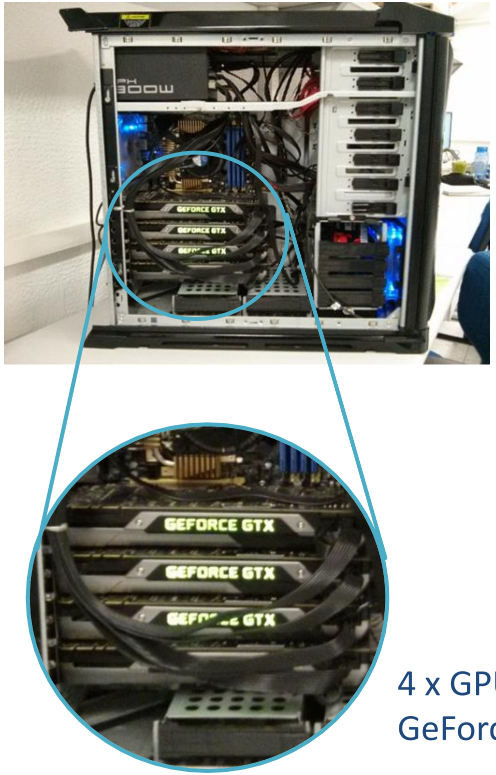
Domain

CHRONOS: FE Program

- **Chronos** has been implemented on GPU
  - **Motivation**: to reduce the simulation time in the hydromechanical analysis
  - **Why to use GPU?** Much more processing power

GPU		CPU
2880 cores	>>	4 - 8 cores

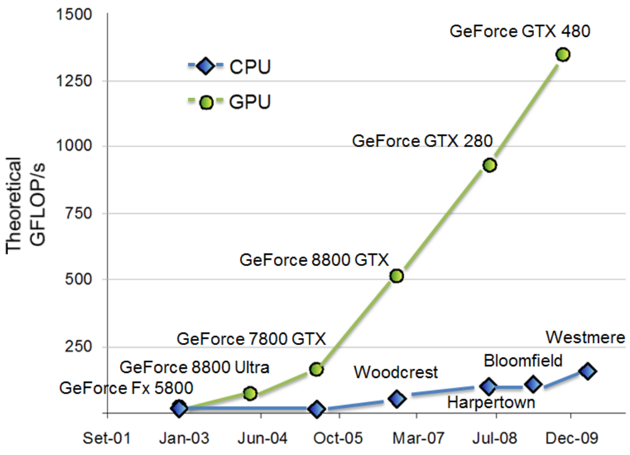
CETUS Computer with 4 GPUs



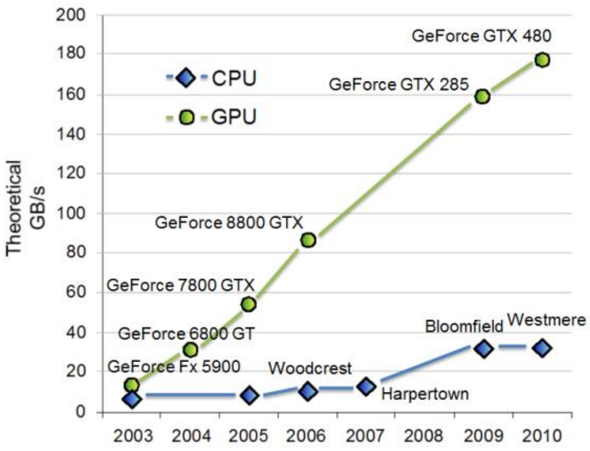
4 x GPUs
GeForce GTX Titan

Motivation

- GPU Features: (Cuda C Programming Guide)
  - Highly parallel, multithreaded and manycore processor
  - Tremendous computational horsepower and very high memory bandwidth



Number of FLoating-point Operations Per Second



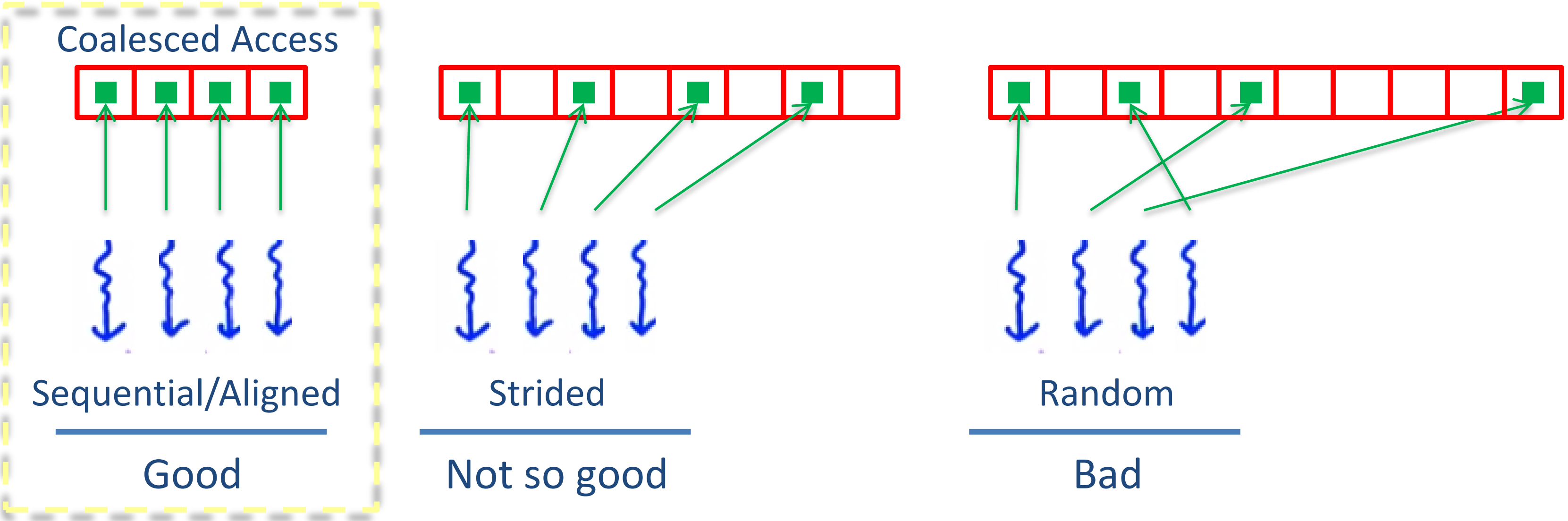
Bandwidth

Our Implementation

- GPUs have good performance
- We have developed and implemented an **optimized and parallel finite element program on GPU**
- Programming Language CUDA is used to implement the finite element code
- We have Implemented on GPU:
  - **Assembly of the stiffness matrix**
  - **Solution of the system of linear equation**
  - Evaluation of the strain state
  - Evaluation of the stress state

Global Memory Access on GPU

- Getting maximum performance on GPU

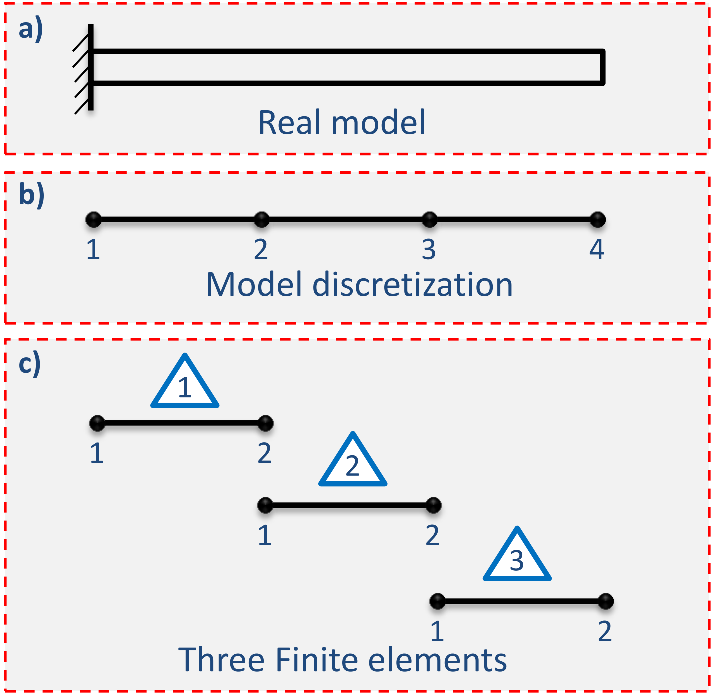


  - Memory accesses are fully coalesced as long as all threads in a warp access the same relative address

Development on CPU

- The assembly of the global stiffness matrix in the conventional FEM

  – Simple 1D problem



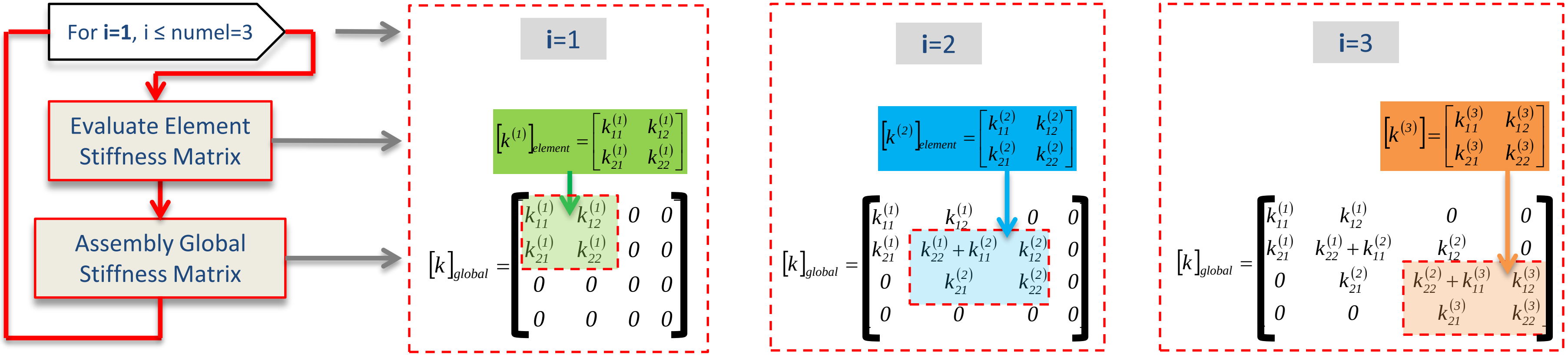
  – Element Stiffness Matrix

  - Element 1 → $\left[k^{(1)}\right]=\begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} \end{bmatrix}$
  - Element 2 → $\left[k^{(2)}\right]=\begin{bmatrix} k_{11}^{(2)} & k_{12}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} \end{bmatrix}$
  - Element 3 → $\left[k^{(3)}\right]=\begin{bmatrix} k_{11}^{(3)} & k_{12}^{(3)} \\ k_{21}^{(3)} & k_{22}^{(3)} \end{bmatrix}$

- Continuous model is discretized **by elements**

Development on CPU

- In terms of CPU implementation



  - The Storage in the memory

i=1 $[k]_{element} = [k_{11}^{(1)} \quad k_{12}^{(1)} \quad 0 \quad 0 \quad k_{21}^{(1)} \quad k_{22}^{(1)} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

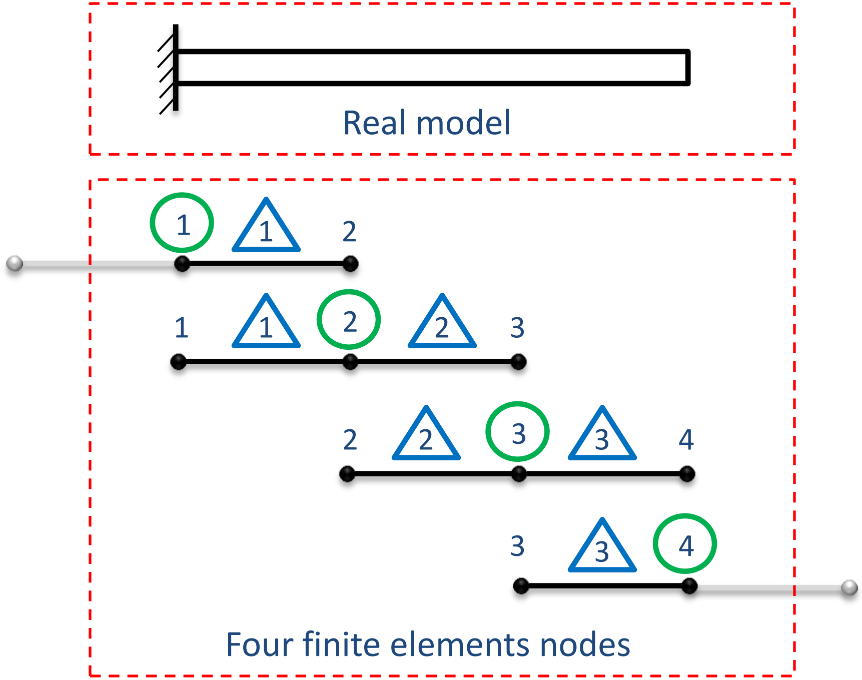
i=2 $[k]_{element} = [k_{11}^{(1)} \quad k_{12}^{(1)} \quad 0 \quad 0 \quad k_{21}^{(1)} \quad k_{22}^{(1)} + k_{11}^{(2)} \quad k_{12}^{(1)} \quad 0 \quad 0 \quad k_{21}^{(1)} \quad k_{22}^{(1)} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$

i=3 $[k]_{element} = [k_{11}^{(1)} \quad k_{12}^{(1)} \quad 0 \quad 0 \quad k_{21}^{(1)} \quad k_{22}^{(1)} + k_{11}^{(2)} \quad k_{12}^{(2)} \quad 0 \quad 0 \quad k_{21}^{(2)} \quad k_{22}^{(2)} + k_{11}^{(3)} \quad k_{12}^{(3)} \quad 0 \quad 0 \quad k_{21}^{(3)} \quad k_{22}^{(3)}]$

Memory access is not coalesced

Development on GPU

- The assembly of the global stiffness matrix on GPU
  - Simple 1D problem



  - Each row of the global stiffness matrix
    - Node 1 → $[k^{row=1}] = [k_{11}^{(\times)} \quad k_{22}^{(\times)} + k_{11}^{(1)} \quad k_{12}^{(1)}]$
    - Node 2 → $[k^{row=2}] = [k_{21}^{(1)} \quad k_{22}^{(1)} + k_{11}^{(2)} \quad k_{12}^{(2)}]$
    - Node 3 → $[k^{row=3}] = [k_{21}^{(2)} \quad k_{22}^{(2)} + k_{11}^{(3)} \quad k_{12}^{(3)}]$
    - Node 3 → $[k^{row=4}] = [k_{21}^{(3)} \quad k_{22}^{(3)} + k_{11}^{(\times)} \quad k_{12}^{(\times)}]$

- Continuous model is discretized by nodes

Development on GPU

- In terms of GPU implementation

Thread = 1

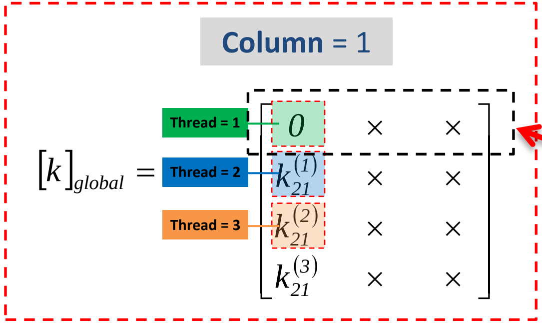
$[k^{row=1}] = [0 \quad k_{11}^{(1)} \quad k_{12}^{(1)}]$

Thread = 2

$[k^{row=2}] = [k_{21}^{(1)} \quad k_{22}^{(1)} + k_{11}^{(2)} \quad k_{12}^{(2)}]$

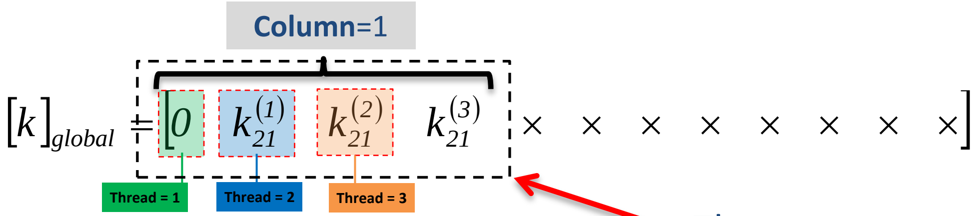
Thread = 3

$[k^{row=3}] = [k_{21}^{(2)} \quad k_{22}^{(2)} + k_{11}^{(3)} \quad k_{12}^{(3)}]$



All the threads do the same calculation

- The Storage in the memory



The memory access is sequential and aligned

Development on GPU

- In terms of GPU implementation

Thread = 1

$$[k^{row=1}] = [0 \quad k_{11}^{(1)} \quad k_{12}^{(1)}]$$

Thread = 2

$$[k^{row=2}] = [k_{21}^{(1)} \quad k_{22}^{(1)} + k_{11}^{(2)} \quad k_{12}^{(2)}]$$

Thread = 3

$$[k^{row=3}] = [k_{21}^{(2)} \quad k_{22}^{(2)} + k_{11}^{(3)} \quad k_{12}^{(3)}]$$

Column = 2

$$[k]_{global} = \begin{bmatrix} & k_{12}^{(1)} & \times \\ k_{21}^{(1)} & k_{22}^{(1)} + k_{11}^{2} & \times \\ k_{21}^{(2)} & k_{22}^{(2)} + k_{11}^{3} & \times \\ k_{21}^{(3)} & k_{22}^{(3)} & \times \end{bmatrix}$$

Thread = 1 → $k_{12}^{(1)}$; Thread = 2 → $k_{22}^{(1)} + k_{11}^{2}$; Thread = 3 → $k_{22}^{(2)} + k_{11}^{3}$

  - The Storage in the memory

Memory access is coalesced

Column=2

$$[k]_{global} = [0 \quad k_{21}^{(1)} \quad k_{21}^{(2)} \quad k_{21}^{(3)} \quad k_{12}^{(1)} \quad k_{22}^{(1)} + k_{11}^{(2)} \quad k_{22}^{(2)} + k_{11}^{(3)} \quad k_{22}^{(3)} \quad \times \quad \times \quad \times \quad \times]$$

Thread = 1 → $k_{12}^{(1)}$; Thread = 2 → $k_{22}^{(1)} + k_{11}^{(2)}$; Thread = 3 → $k_{22}^{(2)} + k_{11}^{(3)}$

Development on GPU

- Solution of the systems of linear equations **Ax = b**
  - Direct solver
  - Iterative Solver
  - **A** = stiffness matrix, **x** = nodal displacement vector (unknown values) and **b** = nodal force vector
  - **A** is a symmetric and positive-definite
- It was chosen the Conjugate Gradient Method
  - Iterative algorithm
  - Parallelizable algorithm on GPU
  - The operations of a conjugate gradient algorithm is suitable to implement on GPU

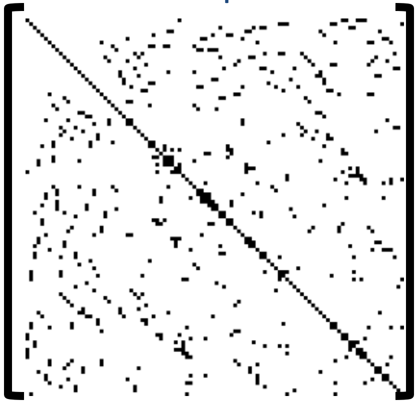
Conjugate Gradient Algorithm

$$
\begin{array}{l}
i \leftarrow 0;\ \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x};\ \mathbf{d} \leftarrow \mathbf{M}^{-1}\mathbf{r}; \\
\delta_{new} \leftarrow \mathbf{r}^T\mathbf{d};\ \delta_0 \leftarrow \delta_{new}; \\
\textbf{while } i < i_{max} \textbf{ and } \delta_{new} > \varepsilon^2\delta_0 \textbf{ do} \\
\qquad \mathbf{q} \leftarrow \mathbf{A}\mathbf{d};\ \alpha \leftarrow \frac{\delta_{new}}{\mathbf{d}^T\mathbf{q}}; \\
\qquad \mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{d};\ \mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{q}; \\
\qquad \mathbf{s} \leftarrow \mathbf{M}^{-1}\mathbf{r};\ \delta_{old} \leftarrow \delta_{new}; \\
\qquad \delta_{new} \leftarrow \mathbf{r}^T\mathbf{s};\ \beta \leftarrow \frac{\delta_{new}}{\delta_{old}}; \\
\qquad \mathbf{d} \leftarrow \mathbf{r} + \beta\mathbf{d};\ i \leftarrow i + 1; \\
\textbf{end}
\end{array}
$$

Development on GPU

- Additional remarks
  - Stiffness matrix **K** → sparse matrix
  - Sparse matrix = most of the elements are zero
  - Assembling the stiffness matrix by nodes = compressed stiffness matrix

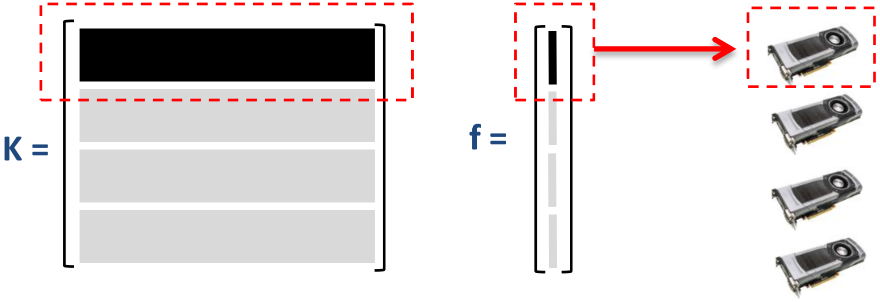
Stiffness Matrix – sparse matrix



  - The bottleneck → Compressed Matrix-Vector Multiplication
    - to map the compressed stiffness matrix

Development on GPU

- Conjugate Gradient Method on GPU
  - To show two operations of the Conjugate Gradient Method
  - The algorithm has been implemented on 4 GPUs
  - Each GPU receives a fourth part of the **K** and **f**

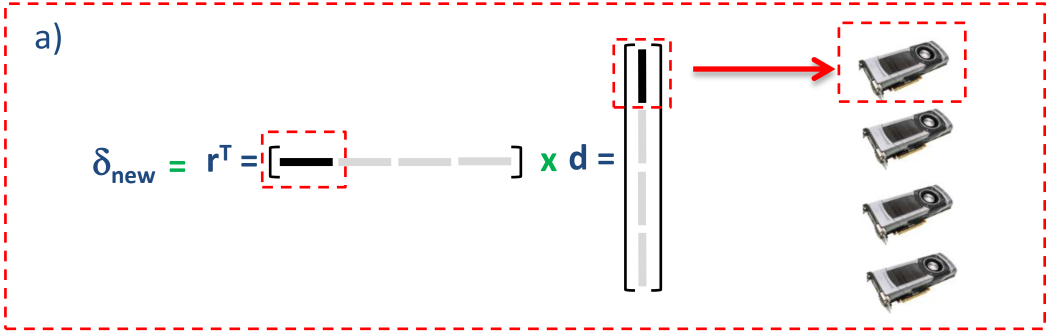


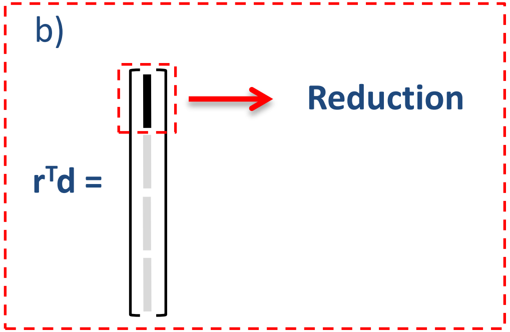
Stiffness Matrix 128 columns

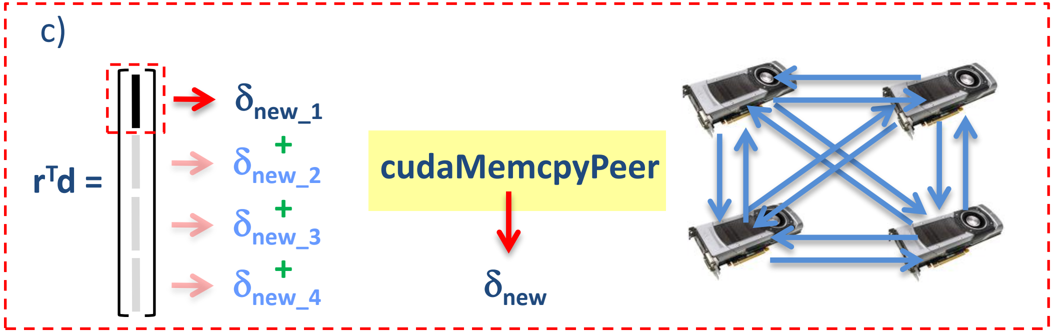
Nodal Force Vector

Development on GPU

- Conjugate Gradient Method on GPU
  - Vector-Vector Multiplication $\delta_{new} = r^T d$







Conjugate gradient algorithm

$i \leftarrow 0;\ \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x};\ \mathbf{d} \leftarrow \mathbf{M}^{-1}\mathbf{r};$

$\delta_{new} \leftarrow \mathbf{r}^T\mathbf{d};\ \delta_0 \leftarrow \delta_{new};$

while $i < i_{max}$ **and** $\delta_{new} > \varepsilon^2\delta_0$ **do**

$\quad \mathbf{q} \leftarrow \mathbf{A}\mathbf{d};\ \alpha \leftarrow \frac{\delta_{new}}{\mathbf{d}^T\mathbf{q}};$

$\quad \mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{d};\ \mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{q};$

$\quad \mathbf{s} \leftarrow \mathbf{M}^{-1}\mathbf{r};\ \delta_{old} \leftarrow \delta_{new};$

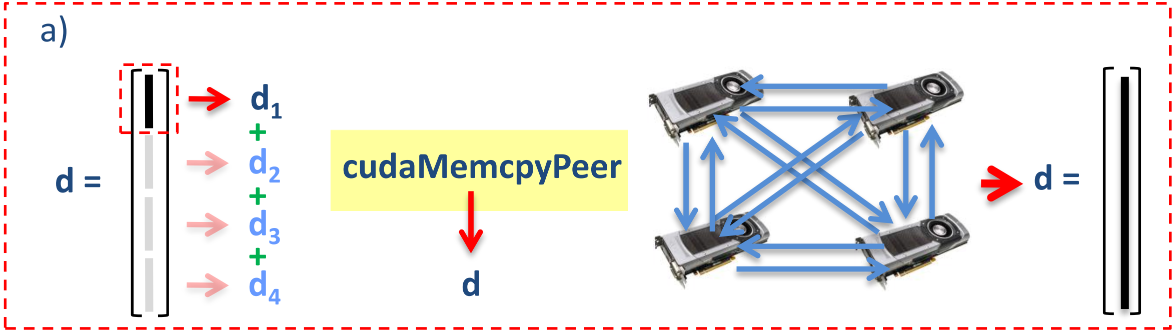
$\quad \delta_{new} \leftarrow \mathbf{r}^T\mathbf{s};\ \beta \leftarrow \frac{\delta_{new}}{\delta_{old}};$

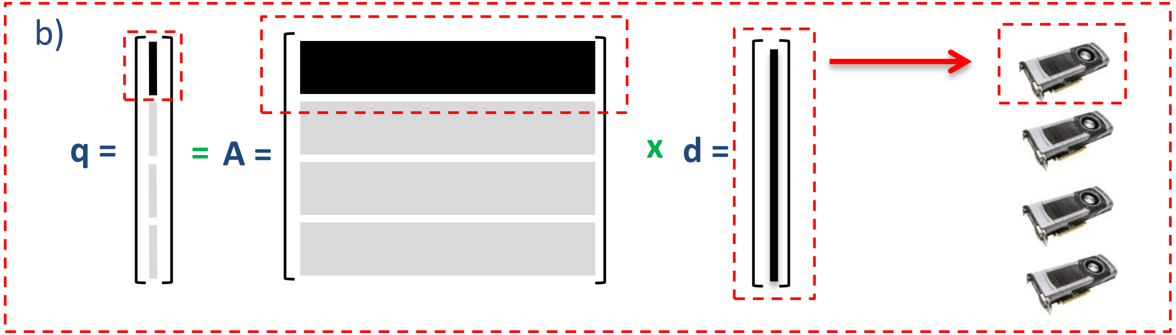
$\quad \mathbf{d} \leftarrow \mathbf{r} + \beta\mathbf{d};\ i \leftarrow i + 1;$

end

Development on GPU

- Conjugate Gradient Method on GPU
  - Matrix-Vector Multiplication $q = Ad$



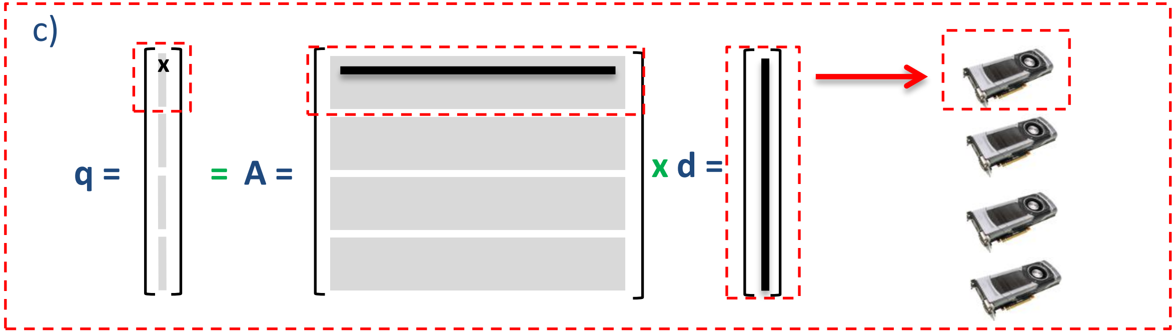


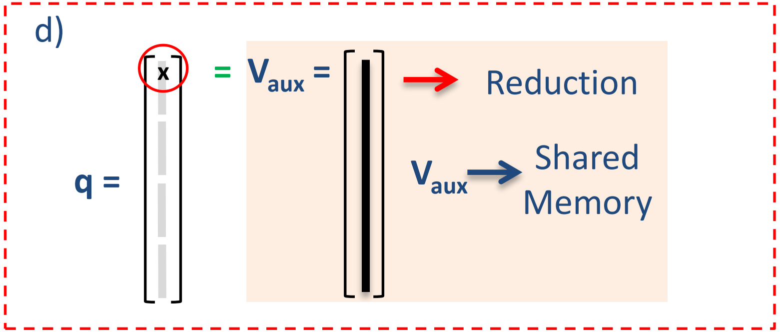
Conjugate gradient algorithm

$$
\begin{aligned}
& i \leftarrow 0;\ \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x};\ \mathbf{d} \leftarrow \mathbf{M}^{-1}\mathbf{r}; \\
& \delta_{new} \leftarrow \mathbf{r}^T\mathbf{d};\ \delta_0 \leftarrow \delta_{new}; \\
& \textbf{while } i < i_{max} \textbf{ and } \delta_{new} > \varepsilon^2\delta_0 \textbf{ do} \\
& \quad \mathbf{q} \leftarrow \mathbf{A}\mathbf{d};\ \alpha \leftarrow \frac{\delta_{new}}{\mathbf{d}^T\mathbf{q}}; \\
& \quad \mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{d};\ \mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{q}; \\
& \quad \mathbf{s} \leftarrow \mathbf{M}^{-1}\mathbf{r};\ \delta_{old} \leftarrow \delta_{new}; \\
& \quad \delta_{new} \leftarrow \mathbf{r}^T\mathbf{s};\ \beta \leftarrow \frac{\delta_{new}}{\delta_{old}}; \\
& \quad \mathbf{d} \leftarrow \mathbf{r} + \beta\mathbf{d};\ i \leftarrow i + 1; \\
& \textbf{end}
\end{aligned}
$$

Development on GPU

- Conjugate Gradient Method on GPU
  - Matrix-Vector Multiplication $q = Ad$





Conjugate gradient algorithm

$$
\begin{aligned}
& i \leftarrow 0;\ \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x};\ \mathbf{d} \leftarrow \mathbf{M}^{-1}\mathbf{r}; \\
& \delta_{new} \leftarrow \mathbf{r}^T\mathbf{d};\ \delta_0 \leftarrow \delta_{new}; \\
& \textbf{while } i < i_{max} \textbf{ and } \delta_{new} > \varepsilon^2\delta_0 \textbf{ do} \\
& \quad \mathbf{q} \leftarrow \mathbf{A}\mathbf{d};\ \alpha \leftarrow \frac{\delta_{new}}{\mathbf{d}^T\mathbf{q}}; \\
& \quad \mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{d};\ \mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{q}; \\
& \quad \mathbf{s} \leftarrow \mathbf{M}^{-1}\mathbf{r};\ \delta_{old} \leftarrow \delta_{new}; \\
& \quad \delta_{new} \leftarrow \mathbf{r}^T\mathbf{s};\ \beta \leftarrow \frac{\delta_{new}}{\delta_{old}}; \\
& \quad \mathbf{d} \leftarrow \mathbf{r} + \beta\mathbf{d};\ i \leftarrow i + 1; \\
& \textbf{end}
\end{aligned}
$$

Previous Results

- Linear Equation Solution
  - Conjugate Gradient Solution for an Optimized GPU and Naïve CPU Algorithm (2010)

TABLE 1: Hardware Configuration

Device	Type	Number of cores	Memory size
GPU	GeForce GTX 285 1.476 GHz	240	1 GB Global Memory
CPU	Intel Xeon X3450 2.67GHz	4	8 GB

TABLE 2: Results

Number of Elements	Simulation Time (s)			
	CPU	8600 GT	9800 GTX	GTX 285
10.000	1.26	1.21	0.37	0.36 (3.5 x)
40.000	10.90	9.05	0.99	0.61 (17.87 x)
250.000	130.5	136.3	13.13	5.38 (24.25 x)

Previous Results

- Assembly of the Stiffness Matrix
  - Comparison for an Optimized GPU and Naïve CPU Algorithm (2011)

TABLE 3: Hardware Configuration

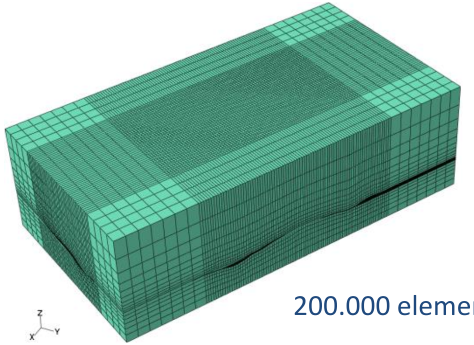
Device	Type	Number of cores	Memory size
GPU	GeForce GTX 460M 1.35 GHz	192	1 GB Global Memory
CPU	Intel Core i7-740QM 1.73 GHz	4	6 GB

TABLE 4: Results

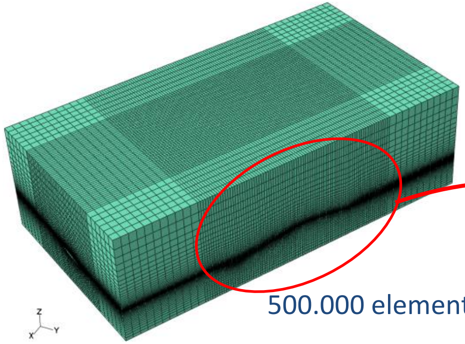
	Simulation Time (ms)	
Number of nodes	CPU	GTX 460M
6400	82.28	0.86 (96 x)
8100	122.77	1.02 (120 x)
10000	323.20	1.24 (261 x)

Current Results

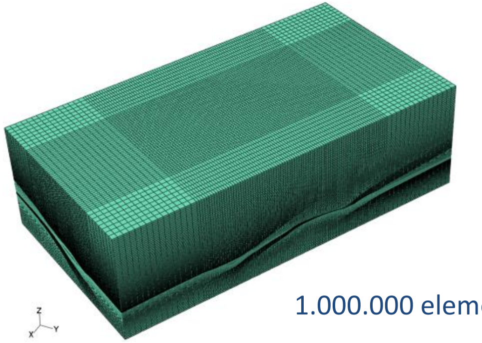
- Finite Element Mesh - 4 discretization



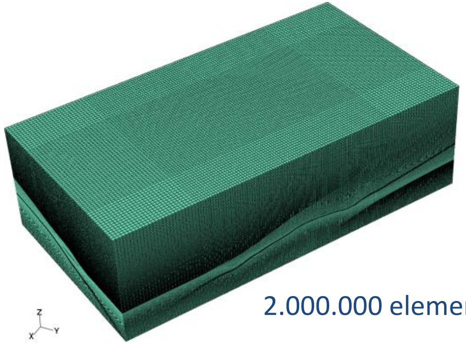
200.000 elements



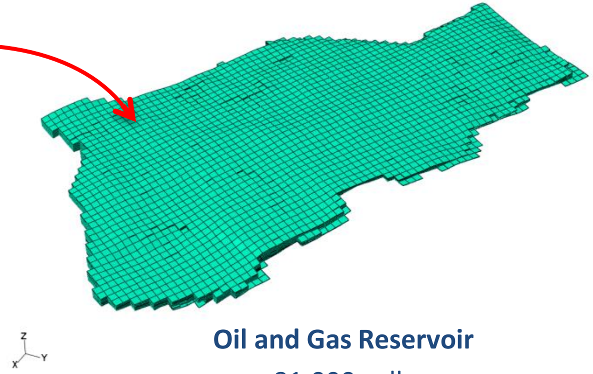
500.000 elements



1.000.000 elements



2.000.000 elements



Oil and Gas Reservoir

81.000 cells

Current Results

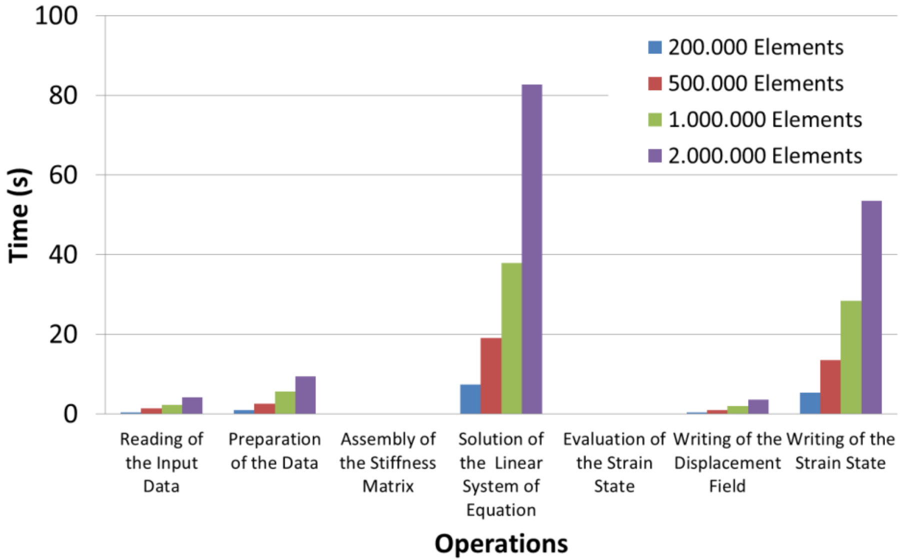
- The time spent in each operation in Chronos

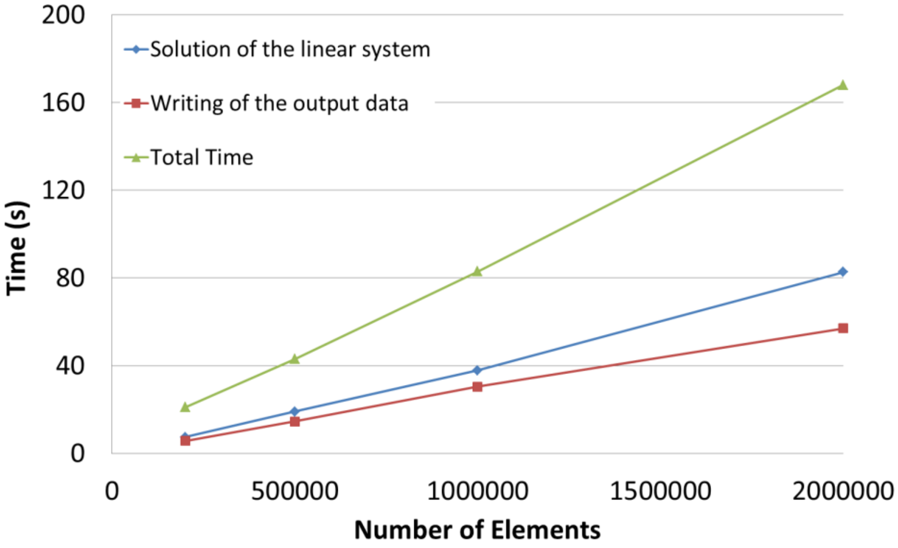
TABLE 5: Time of each operation

Operations	Elements							
	200.000		500.000		1.000.000		2.000.000	
	Time (s)	Time (%)	Time (s)	Time (%)	Time (s)	Time (%)	Time (s)	Time (%)
Reading of the Input Data	0,390	2,70	1,407	3,75	2,253	2,96	4,145	2,70
Preparation of the Data	0,985	6,81	2,616	6,97	5,600	7,36	9,468	6,17
Assembly of the Stiffness Matrix	0,001	0,01	0,001	0,00	0,001	0,00	0,001	0,00
Solution of the System of Linear Equation	7,375	50,99	18,985	50,59	37,841	49,74	82,697	53,93
Evaluation of the Strain State	0,001	0,01	0,001	0,00	0,001	0,00	0,001	0,00
Writing of the Displacement Field	0,402	2,78	0,950	2,53	1,923	2,53	3,521	2,30
Writing of the Strain State	5,311	36,72	13,568	36,15	28,463	37,41	53,506	34,89
Total Time	14	100	38	100	76	100	153	100

Current Results

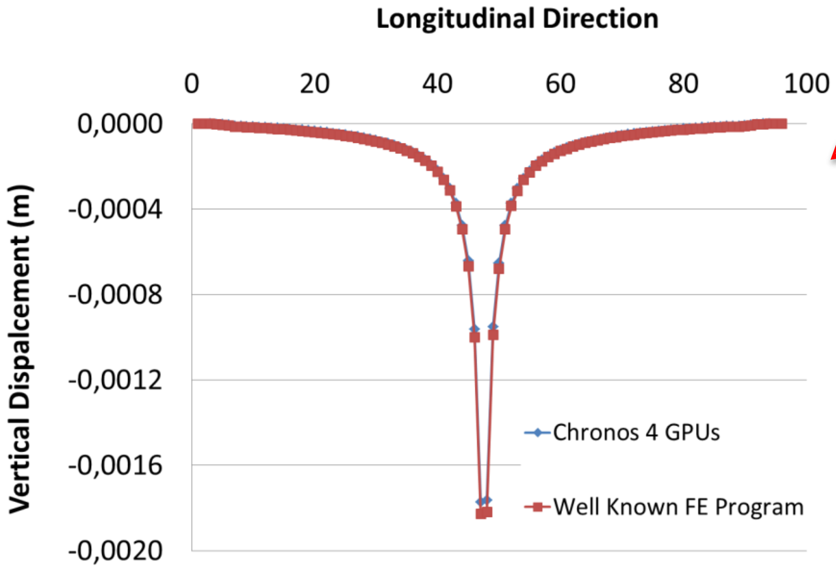
- The time spent in each operation in Chronos

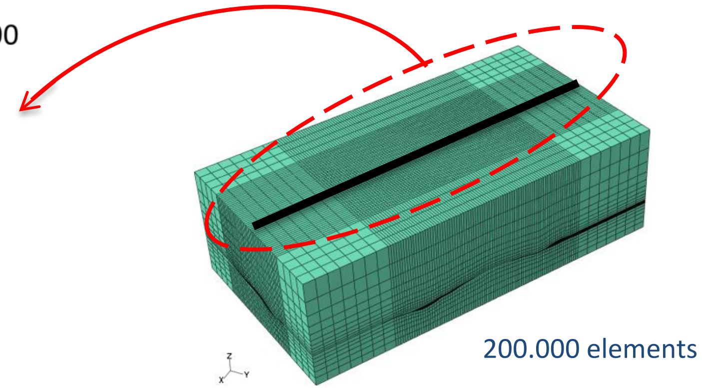




Current Results

- The accuracy verification: Chronos vs. Well known FE program





GPU TECHNOLOGY CONFERENCE

Current Results

- Time Comparison: Chronos vs. Well known FE program

TABLE 6: Hardware Configuration

Device	Type	Number of cores	Memory size
4 x GPU	GeForce GTX Titan 0.876 GHz	2688	6 GB Global Memory
CPU	Intel Core i7-4770 3.40 GHz	4	32 GB

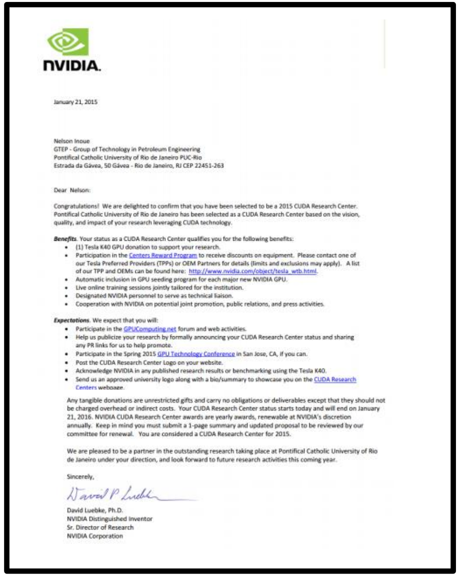
TABLE 7: Results

Number of Elements	Simulation Time (s)		Performance Improvement
	Chronos 4 GPUs	Well Known FE Program	
200.000	21	516 (8.6 min)	24,57 x
500.000	43	3407 (56.78 min)	79,23 x
1.000.000	83	Insufficient Memory	x
2.000.000	168	Insufficient Memory	x

NVIDIA CUDA Research Center

- Pontifical Catholic University of Rio de Janeiro is a **NVIDIA CUDA Research Center**

CUDA Research Center Logo



January 21, 2015

Nelson Inoue
GTEP - Group of Technology in Petroleum Engineering
Pontifical Catholic University of Rio de Janeiro PUC-Rio
Estrada da Gávea, 50 Gávea - Rio de Janeiro, RJ CEP 22451-263

Dear Nelson:

Congratulations! We are delighted to confirm that you have been selected to be a 2015 CUDA Research Center. Pontifical Catholic University of Rio de Janeiro has been selected as a CUDA Research Center based on the vision, quality, and impact of your research leveraging CUDA technology.

Benefits. Your status as a CUDA Research Center qualifies you for the following benefits:

- (1) Tesla K40 GPU donation to support your research.
- Participation in the Centers Reward Program to receive discounts on equipment. Please contact one of our Tesla Preferred Providers (TPPs) or OEM Partners for details (limits and exclusions may apply). A list of our TPP and OEMs can be found here: http://www.nvidia.com/object/tesla_wtb.html.
- Automatic inclusion in GPU seeding program for each major new NVIDIA GPU.
- Live online training sessions jointly tailored for the institution.
- Designated NVIDIA personnel to serve as technical liaison.
- Cooperation with NVIDIA on potential joint promotion, public relations, and press activities.

Expectations. We expect that you will:

- Participate in the GPUComputing.net forum and web activities.
- Help us publicize your research by formally announcing your CUDA Research Center status and sharing any PR links for us to help promote.
- Participate in the Spring 2015 GPU Technology Conference in San Jose, CA, if you can.
- Post the CUDA Research Center Logo on your website.
- Acknowledge NVIDIA in any published research results or benchmarking using the Tesla K40.
- Send us an approved university logo along with a bio/summary to showcase you on the CUDA Research Centers webpage.

Any tangible donations are unrestricted gifts and carry no obligations or deliverables except that they should not be charged overhead or indirect costs. Your CUDA Research Center status starts today and will end on January 21, 2016. NVIDIA CUDA Research Center awards are yearly awards, renewable at NVIDIA's discretion annually. Keep in mind you must submit a 1-page summary and updated proposal to be reviewed by our committee for renewal. You are considered a CUDA Research Center for 2015.

We are pleased to be a partner in the outstanding research taking place at Pontifical Catholic University of Rio de Janeiro under your direction, and look forward to future research activities this coming year.

Sincerely,

David Luebke, Ph.D.
NVIDIA Distinguished Inventor
Sr. Director of Research
NVIDIA Corporation

CUDA Research Center award letter

PUC-Rio Homepage

Conclusions

- GPUs has showed great potential to speed up numerical analyses
- However, the speed-up may only be reached, in general, if new programs or algorithms are implemented and optimized in a parallel way for GPUs

Acknowledgements

- The authors would like to thank Petrobras for the financial support and SIMULIA and CMG for providing the academic licenses for the programs Abaqus and Imex, respectively
- And NVIDIA for the opportunity to show our work in this Conference

GPU TECHNOLOGY CONFERENCE

Pontifícia Universidade Católica do Rio de Janeiro

Thank You