

GPU Linear solver for reservoir simulation on GPU, An industrial perspective

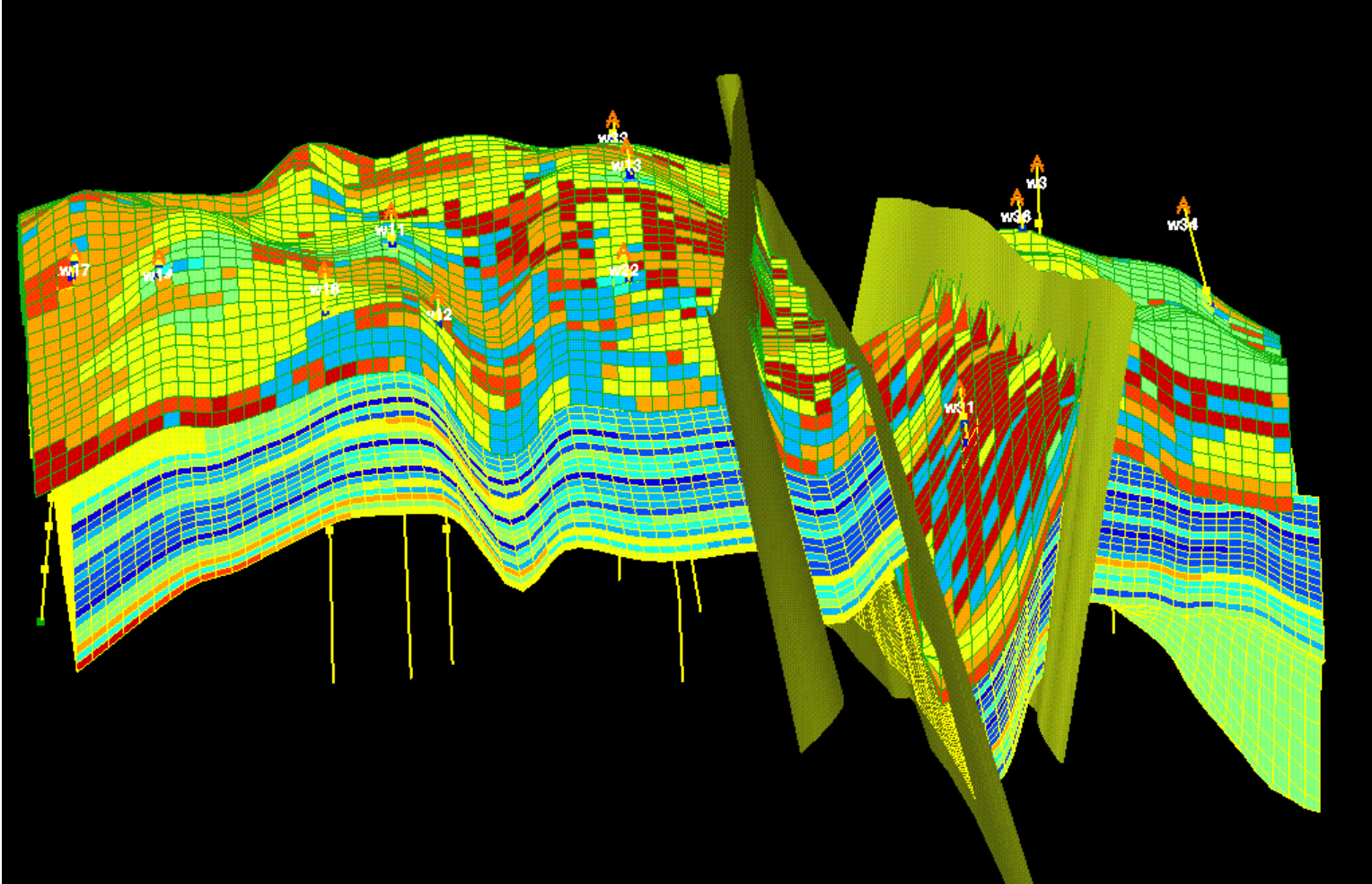
Mohamed HACENE
Thomas GUIGNON
Stéphane REQUENA
IFP



Reservoir Simulation (1)

- **Geological structure:**
 - Porous media with water oil and gaz
 - Wells (producer, injector)
- **Goal: simulate production phase behavior:**
 - Production forecast.
 - Improving oil recovery.
 - Plan for new drilling, injection ...
- **Multiphase flow in porous media:**
 - Darcy flow equation (flow speed, pressure porosity)
 - Conservation laws.
 - Finite volume discretization.
 - Structured, unstructured and CPG mesh.

Reservoir simulation (2)



Simulation de réservoir (3)

- **Discretization: non linear system / dt**
 - Newton
 - Iterative linear solver (BiCGStab preconditioned): $Ax=b$
 - ~ 80 % simulation time spent in linear solver.
- **A: Unstructured linear system (blocs 3x3)**
 - CSR format.
 - Matrix structure closely related to the mesh.
 - Wells system.
- **Non structured Mesh graph (CPG grid):**
 - faults,
 - dead cells (no porous media),
 - pinch out, local grid refining (LGR)

Linear solver

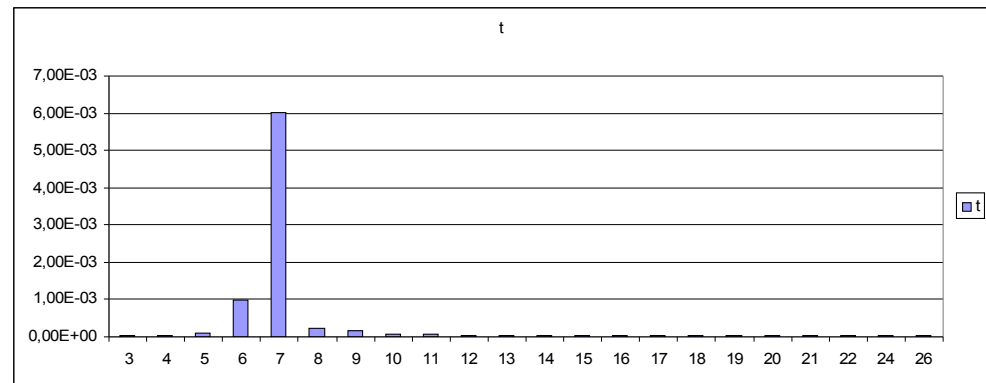
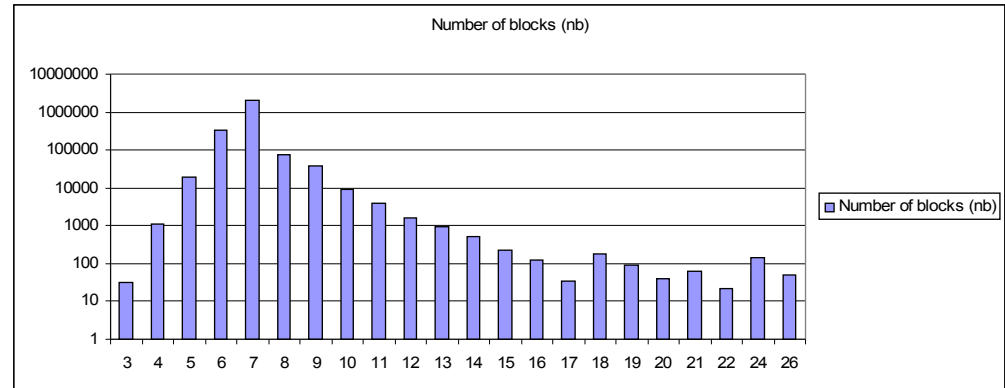
- **Preconditioned BiCGStab:**
 - Iterative Krylov.
 - Sparse Matrix vector product (+40%)
 - Preconditioner (+40% *)
 - Other BLAS 1 (-20%)
- **GPU Strategy:**
 - Iterative part done on GPU
 - Input: A et b, M
 - Output: x and residual
 - M computation is still done on CPU.
- **At best x5 acceleration on whole simulation**
 - But ask a reservoir engineer...

Sparse matrix vector product (spmv) on GPU with CUDA

- **GPU constraint: SIMD (SIMT)**
 - « All threads » do the same things at the same time.
 - Very fine grain parallelism.
- **Simple CSR spmv Impossible:**
 - Variable matrix line length.
 - Threads idles on « short » lines.
 - Alignement constraints.
- **Find regularity: reorder matrix to group lines by width.**
 - $y = P.A.x$ or $y = P.A.P^{-1}.x$
- **1 kernel per width.**

Performance analysis: GCS2K

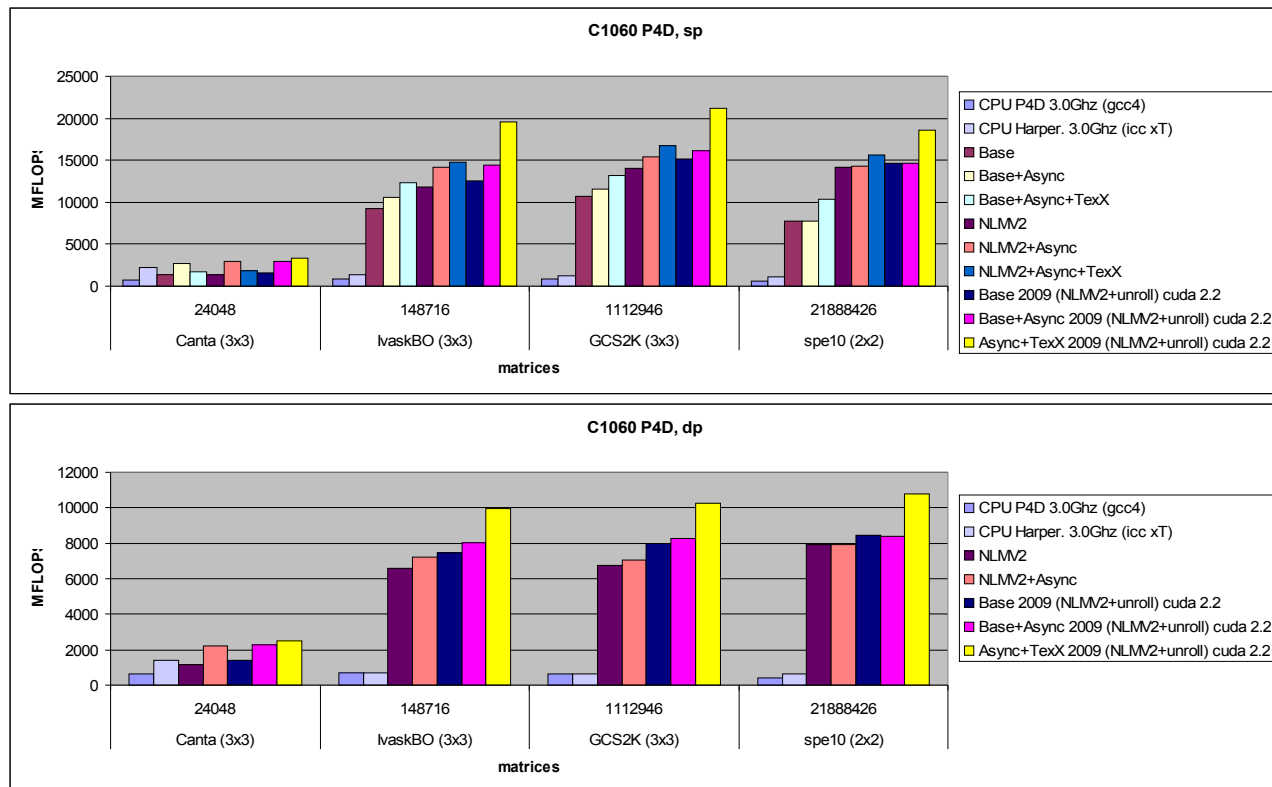
- Majority of blocks and work for width 6 and 7.



Sparse matrix vector product on GPU

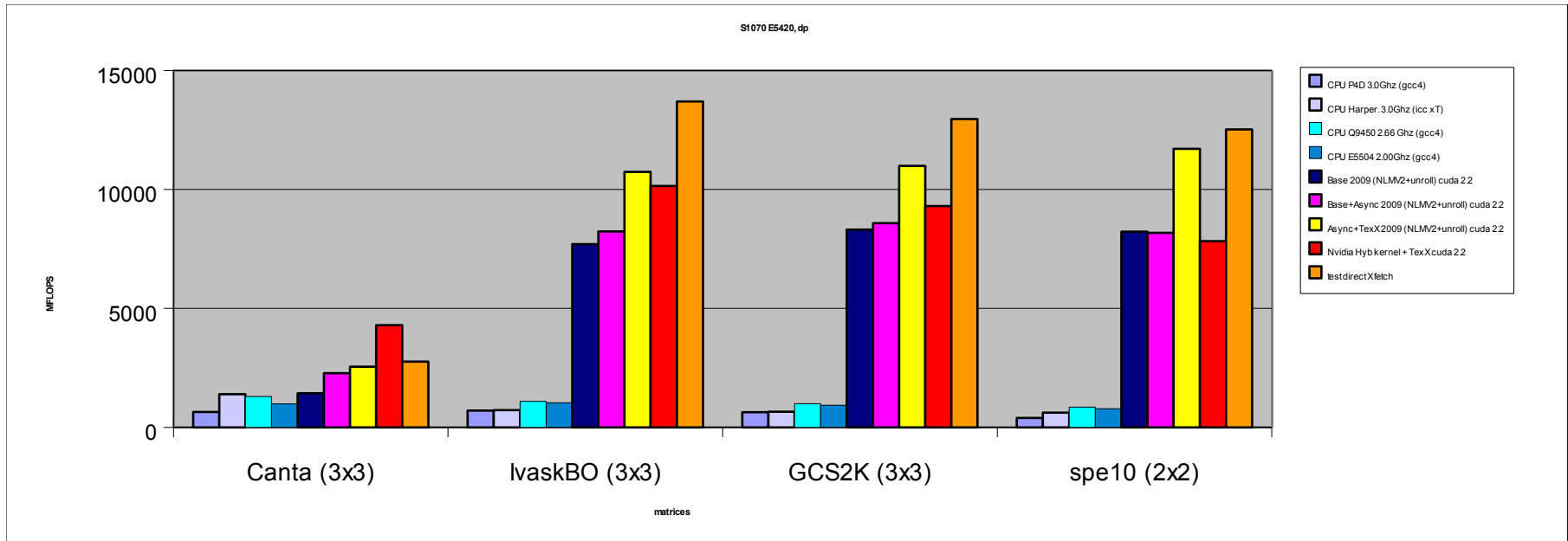
Key points for optimizing spmv:

- Texture cache.
- Non blocking kernel launch.



Concurrent work: Nvidia spmv

- Compared with Nvidia Hybrid + Texture cache spmv:
 - Nvidia spmv (red) really efficient for "small" matrices (Canta)
 - IFP spmv (yellow/orange) better for "big" matrices



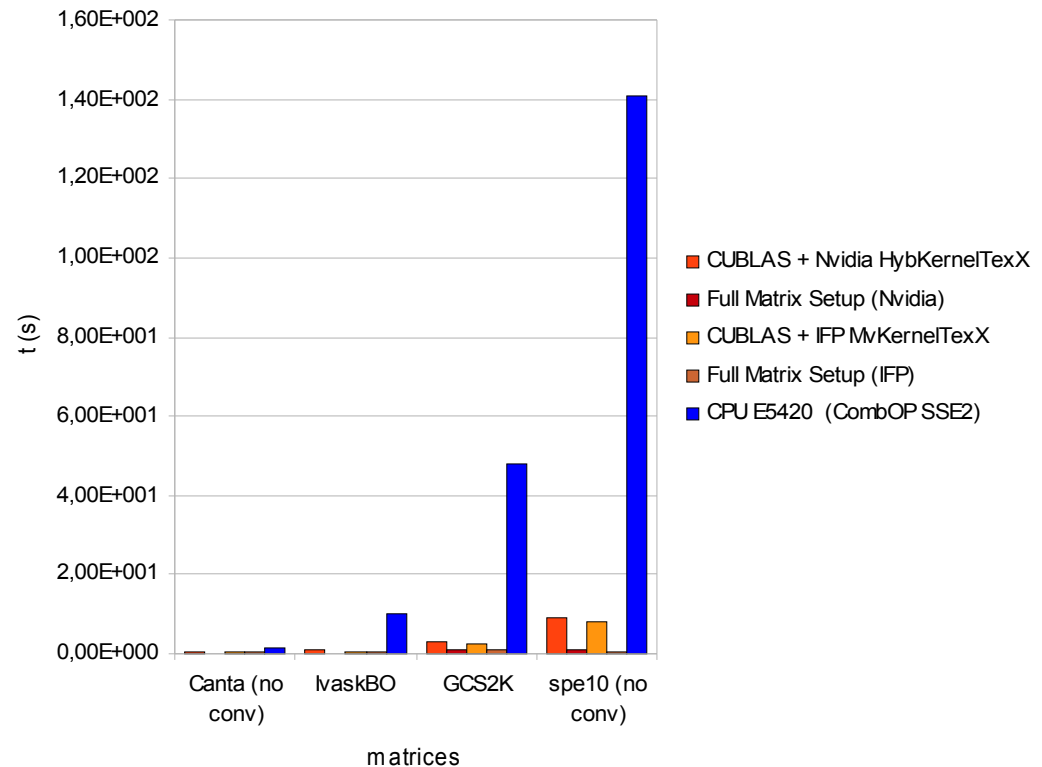
- **LORIA (Inria Lorraine): CNC (Concurrent Number cruncher (CNC)):**
 - <http://alice.loria.fr/index.php/software/4-library/34-concurrent-number-cruncher.html>
 - Paper + code
 - Spmv: point to blocks
 - Linear solver: cg but no preconditioner
- **IBM research: *Sparse Matrix-Vector Multiplication Toolkit for Graphics Processing Units***
 - <http://www.alphaworks.ibm.com/tech/spmv4gpu>
 - No results,
 - Code

BiCGStab (double precision)

■ S1070 tests:

- no preconditioner
- Numerically OK:
 - solutions closed to cpu results.
 - same convergence rates.

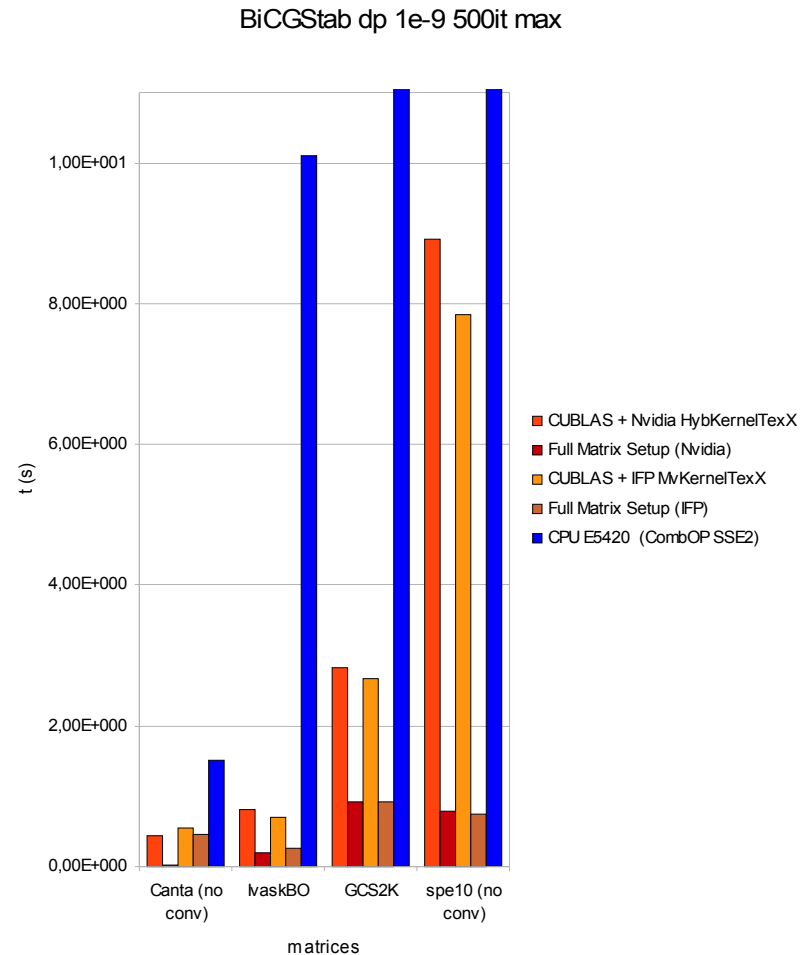
BiCGStab dp 1e-9 500it max



BiCGStab (double precision) ZOOM

■ Setup time important/ relative to iteration time but:

- setup + iteration still better than CPU for Ivak, GCS2K and SP10
- IFP spmv setup too important for small system (Canta)
 - Need to be improved.
- NVidia spmv setup really fast on Canta
 - BiCGStab GPU still interesting.



BiCGStab + ILU0 GPU

■ Stage 2009 M. Hacene:

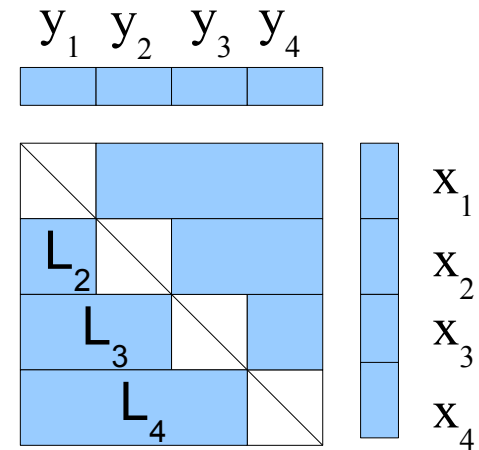
- Solve $L.U.x = y$ on GPU
- GPU massively parallel L.U $x=y$ solve requires A to be permuted.

■ Coloring adjacency matrix graph:

- Matrix reordered by color
- Linear System with large diagonal blocks.

■ Resolution $L.x = y$:

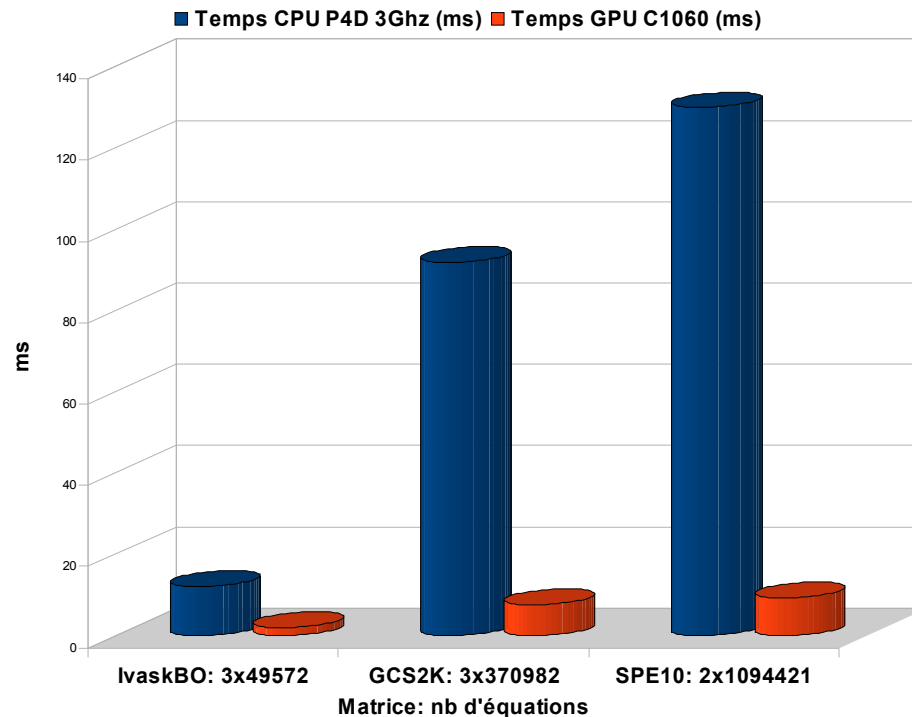
- $x_1 = D_1^{-1} \cdot y_1$
- $x_2 = D_2^{-1} \cdot y_2 - L_2 \cdot x_1$
- ...



GPU LU solve

- Solve $L.U.x = y$ (Greedy coloring) on C1060
 - x6 to x14 acceleration over CPU (P4D 3.Ghz)

Résolution $L.U.x = y$, creux non structuré: Temps CPU/GPU double précision



Reordering: numerical effect

■ Reordering Slow GPU convergence / CPU:

	CPU: natural ordering	GPU: Greedy coloring
Ivak	9	216
GCS2 K	44	80
spe10	393	3237

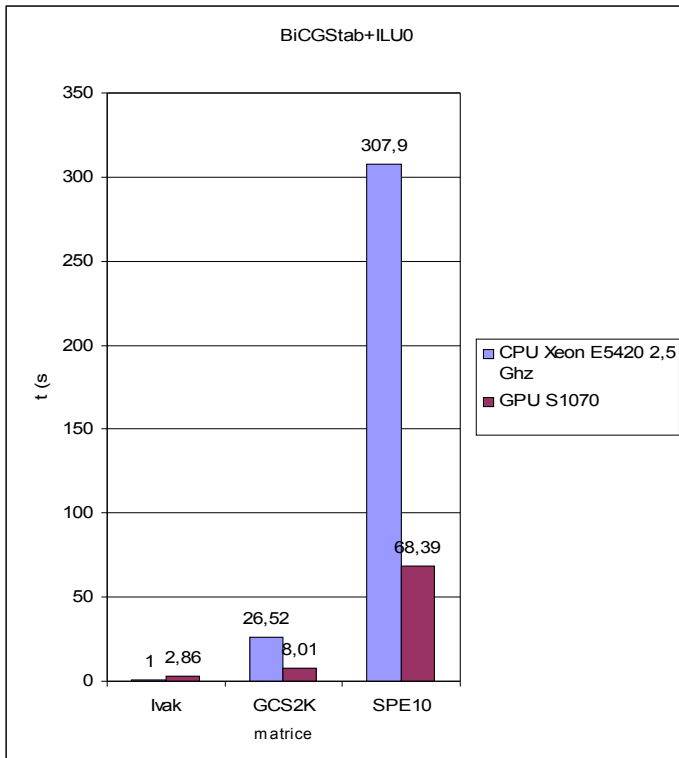
■ Changing coloring improve convergence rate:

- Mainly on spe10
- More color decrease L.U $x = y$ efficiency

BiCGStab + ILU0 GPU

■ M. Hacene 2009 work (first results):

- Matrix needs permutation for massively parallel L.U.x=y solve operation: decrease convergence rates but.



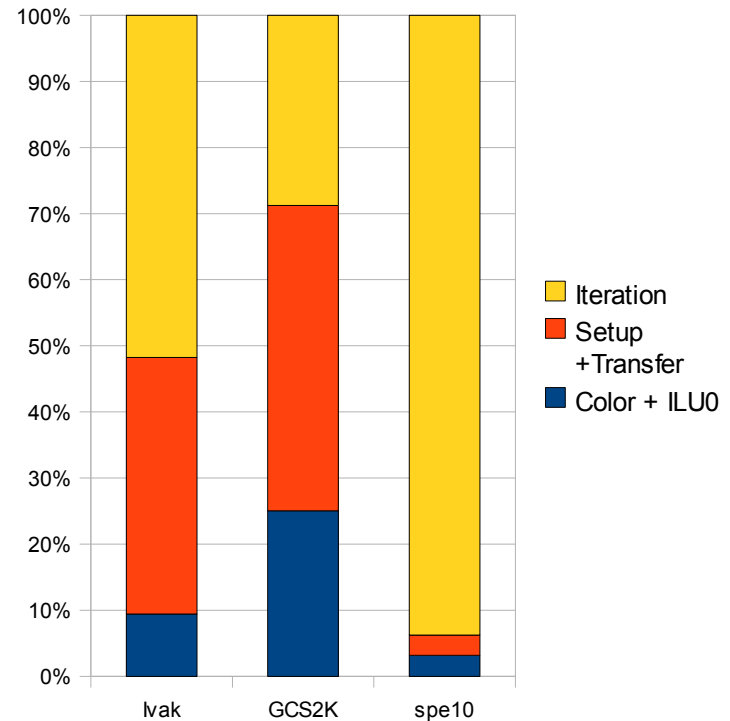
- GPU/CPU: x3 to x4,5 (GCS2K, SPE10)
- Most of time spent in building LU factors for GPU:
 - CPU faster for IvaskBO

BiCGStab + ILU0: the costs

■ Improvements:

- Improved coloring:
 - Faster convergence rates
- Faster coloring
- No matrix structure change
 - Color only one time
 - Part of setup only one time
- Use Nvidia spmv for $L.U.x = y$
 - Faster setup for small matrices
 - Spmv more efficient for small matrices

■ GOAL: x10 on GCS2K



Conclusion / Future work

- **Be more fair: use more cores on CPU.**
- **Improve ILU.**
- **Go Hybrid: MPI + GPU.**
- **Improve IFP spmv (there still way to do)**
- **Go real life: branch a GPU solver in IFP reservoir simulator (and see what happen)**
- **Go OpenCL ?**

Thanks

- **Part of this work was funded by ANR project PARA**
- **NVIDIA for C1060 and GT200 proto cards**
- **CAPS Enterprise for help in profiling, double precision and texture cache.**