Fast and robust ILU preconditioners on the GPU

Kees Vuik, Rohit Gupta, Martin van Gijzen

Martijn de Jong, Auke Ditzel (Marin), Auke van der Ploeg (Marin)

Delft University of Technology

Preconditioning 2013, Oxford, UK.

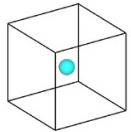
TU Delft
Delft University of Technology

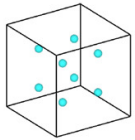
Contents

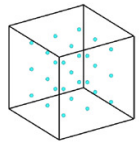
Main question

Can ILU preconditioners be combined with GPU's?

1. Problem Description (Bubbly Flow)







Mass-Conserving Level-Set method to solve the Navier Stokes equation. Marker function ϕ changes sign at interface.

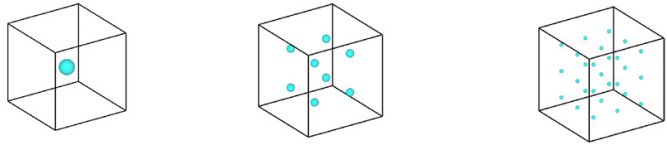
$$S(t) = \{x | \phi(x, t) = 0\}. \tag{1}$$

Interface is evolved using advection of Level-Set function

$$\frac{\partial \phi}{\partial t} + u. \bigtriangledown \phi = 0 \tag{2}$$

[1] A mass-conserving Level-Set method for modeling of multi-phase flows. S.P. van der Pijl, A. Segal and C. Vuik. International Journal for Numerical Methods in Fluids 2005; 47:339–361

1. Problem Description (Bubbly Flow)



$$-\nabla.(\frac{1}{\rho(x)}\nabla p(x)) = f(x),\ x \in \Omega \tag{1}$$

$$\frac{\partial}{\partial n}p(x) = 0,\ x \in \partial\Omega \tag{2}$$

- Pressure-Correction (above) equation is discretized to a linear system $Ax = b$.
- Most time consuming part is the solution of this linear system
- A is Symmetric Positive-Semi-Definite (SPSD) so Conjugate Gradient is the method of choice.

2. Neumannn series and deflation

Truncated Neumann Series Preconditioning[1],[2]

$$M^{-1} = K^T D^{-1} K, \text{ where } K = (I - LD^{-1} + (LD^{-1})^2 + \cdots). \tag{3}$$

(4)

L is the strictly lower triangular of A,
where D=diag(A).

1. More terms give better approximation.
2. In general the series converges if $\| LD^{-1} \|_\infty < 1$.
3. As much parallelism on offer as Sparse Matrix Vector Product.

[1] A vectorizable variant of some ICCG methods. Henk A. van der Vorst. SIAM Journal of Scientific Computing. Vol. 3 No. 3 September 1982.

[2] Approximating the Inverse of a Matrix for use in Iterative Algorithms on Vector Processors. P.F. Dubois. Computing (22) 1979.

Deflation

Background

Removes small eigenvalues from the eigenvalue spectrum of $M^{-1}A$.
The linear system $Ax = b$ can then be solved by employing the splitting,

$$x = (I - P^T)x + P^T x \text{ where } P = I - AQ. \tag{5}$$

$$\Leftrightarrow Pb = PA\hat{x}. \tag{6}$$

$Q = ZE^{-1}Z^T$, $E = Z^T AZ$.
E is the coarse system that is solved every iteration.
Z is the deflation sub-space matrix. It contains an approximation of the eigenvectors of $M^{-1}A$.
For our experiments Z consists of piecewise constant vectors.

Deflation

Deflated Preconditioned Conjugate Gradient Algorithm

1: Select x_0. Compute $r_0 := b - Ax_0$ and $\hat{r_0} = Pr_0$.
2: Solve $My_0 = \hat{r_0}$ and set $p_0 := y_0$.
3: **for** j:=0,..., until convergence **do**
4: $\quad \hat{w}_j := PAp_j$
5: $\quad \alpha_j := \frac{(\hat{r}_j, y_j)}{(p_j, \hat{w}_j)}$
6: $\quad \hat{x}_{j+1} := \hat{x}_j + \alpha_j p_j$
7: $\quad \hat{r}_{j+1} := \hat{r}_j - \alpha_j \hat{w}_j$
8: $\quad$ Solve $My_{j+1} = \hat{r}_{j+1}$
9: $\quad \beta_j := \frac{(\hat{r}_{j+1}, y_{j+1})}{(\hat{r}_j, y_j)}$
10: $\quad p_{j+1} := y_{j+1} + \beta_j p_j$
11: **end for**
12: $x_{it} := Qb + P^T x_{j+1}$

Deflation

Operations involved in deflation[1] [2].

- $a1 = Z^T p$.
- $m = E^{-1} a1$.
- $a2 = AZm$.
- $\hat{w} = p - a2$.

where, $E = Z^T AZ$ is the Galerkin Matrix and Z is the matrix of deflation vectors.

[1] Efficient deflation methods applied to 3-D bubbly flow problems. J.M. Tang, C. Vuik Elec. Trans. Numer. Anal. 2007.

[2] An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients. C. Vuik, A. Segal, J.A. Meijerink J. Comput. Phys. 1999.

Experiments and Results

Stopping Criteria $\rightarrow \frac{\|b-Ax_k\|_2}{\|r_0\|} \leq \epsilon$

1. ϵ is the tolerance we set for the solution.
2. x_k is the solution vector after k iterations of (P)CG.
3. r_0 is the initial residual.

Experiments and Results

Hardware

1. CPU - single core of E8500-3.16 GHz.
2. GPU - Tesla C2070.

Software

1. Inner System solve on CPU is with CG. On GPU it is an Explicit inverse based solution.
2. First Level Preconditioning on CPU is Incomplete Cholesky (IC). On GPU it is Truncated Neumann Series based.
3. Deflation operation is highly-optimized on the CPU.
4. All deflation vectors are piece-wise constant.

Experiments and Results

Timing and Speedup Definition

Speedup is measured as the ratio of the time taken(T) to complete k iterations (of the DPCG method) on the two different architectures,

$$Speedup = \frac{T_{CPU}}{T_{GPU}} \tag{5}$$

- Number of Unknowns = 128^3.
- Tolerance set to 10^{-6}.
- Density Contrast is 10^{-3}

Naming deflation vectors

- SD-i -> Sub-domain deflation with i vectors.
- LS-i -> Level-Set deflation with i vectors.
- LSSD-i -> Level-Set Sub-domain deflation with i vectors.

Experiments and Results

9 bubbles - 64 Sub-domains

	CPU	GPU-CUSP	
	DICCG(0)	DPCG(neu2)	
	SD-64	SD-63	LSSD-135
Number of Iterations	472	603	136
Total Time	81.39	13.61	5.58
Iteration Time	81.1	10.61	2.48
Speedup	-	7.64	32.7

Table : 9 bubbles. Two deflation variants. GPU and CPU Execution Times and Speedup.64 sub-domains.

Experiments and Results

9 bubbles - 512 Sub-domains

	CPU	GPU-CUSP	
	DICCG(0)	DPCG(neu2)	
	SD-512	SD-511	LSSD-583
Number of Iterations	67	81	81
Total Time	12.51	4.56	4.62
Iteration Time	12.18	1.56	1.62
Speedup	-	7.81	7.52

Table : 9 bubbles. Two deflation variants. GPU and CPU Execution Times and Speedup.512 sub-domains.

3. MRILU

- Interactive waves in ship simulator
- Properties of the system matrix
- The RRB solver
- Special ordering
- CUDA implementation
- Results (solver speedup)

Interactive waves in ship simulator

Linearized Variational Boussinesq equations:

$$\frac{\partial \zeta}{\partial t} + \nabla \cdot (\zeta \mathbf{U} + h\nabla\varphi - h\mathcal{D}\nabla\psi) = 0, \tag{1a}$$

$$\frac{\partial \varphi}{\partial t} + \mathbf{U} \cdot \nabla\varphi + g\zeta = -P_s, \tag{1b}$$

$$\mathcal{M}\psi + \nabla \cdot (h\mathcal{D}\nabla\varphi - \mathcal{N}\nabla\psi) = 0. \tag{1c}$$

After discretization (FVM for space, Leapfrog for time):

$$S\vec{\psi} = \mathbf{b}, \tag{2}$$

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = L\mathbf{q} + \mathbf{f}. \tag{3}$$

Properties of the system matrix

The system matrix S is given by a 5-point stencil:

$$\begin{bmatrix} 0 & -\frac{\Delta x}{\Delta y}\overline{\mathcal{N}_N} & 0 \\ -\frac{\Delta y}{\Delta x}\overline{\mathcal{N}_W} & \frac{\Delta x}{\Delta y}\overline{\mathcal{N}_N} + \frac{\Delta y}{\Delta x}\overline{\mathcal{N}_E} + \Delta x \Delta y \mathcal{M}_C + \frac{\Delta x}{\Delta y}\overline{\mathcal{N}_S} + \frac{\Delta y}{\Delta x}\overline{\mathcal{N}_W} & -\frac{\Delta y}{\Delta x}\overline{\mathcal{N}_E} \\ 0 & -\frac{\Delta x}{\Delta y}\overline{\mathcal{N}_S} & 0 \end{bmatrix}.$$

Matrix S is:

- real-valued, sparse (5-point, pentadiagonal)
- diagonally dominant (not very strong for small mesh sizes)
- symmetric positive definite (SPD)
- quite large (in the order of millions by millions)

The RRB-solver (1)

The RRB-solver:

- is a PCG-type solver (preconditioned conjugated gradients)
- uses as preconditioner: the RRB-method

RRB stands for “Repeated Red-Black”.

The RRB-method determines an incomplete factorization:

$$S = LDL^T + R \quad \Longrightarrow \quad M = LDL^T \approx S$$

The RRB-solver (2)

As the name RRB reveals: multiple levels

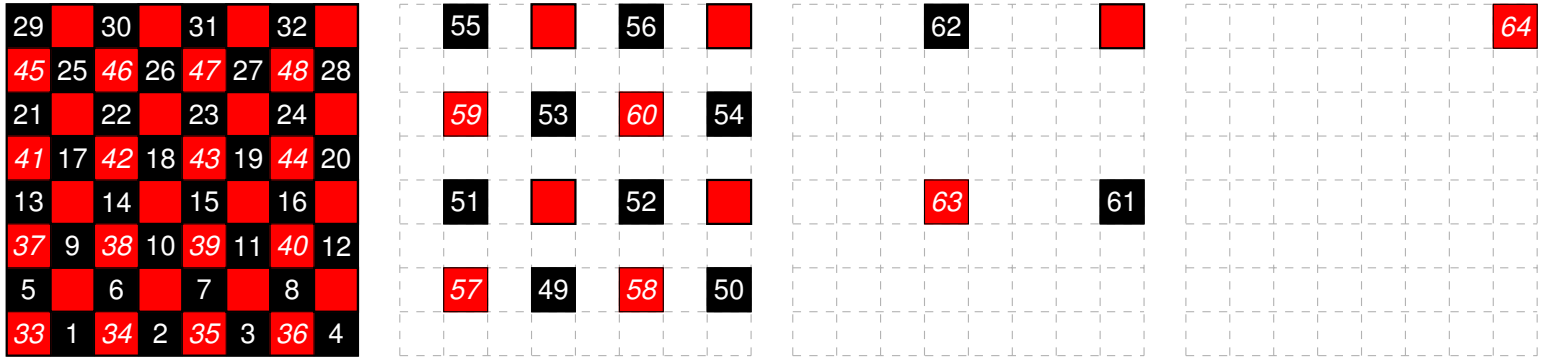
Therefore the RRB-solver has good scaling behaviour (Multigrid)

Method of choice because:

- shown to be robust for all of MARIN's test problems
- solved all test problems up to 1.5 million nodes within 7 iterations(!)

Special ordering (1)

An 8×8 example of the RRB-numbering process



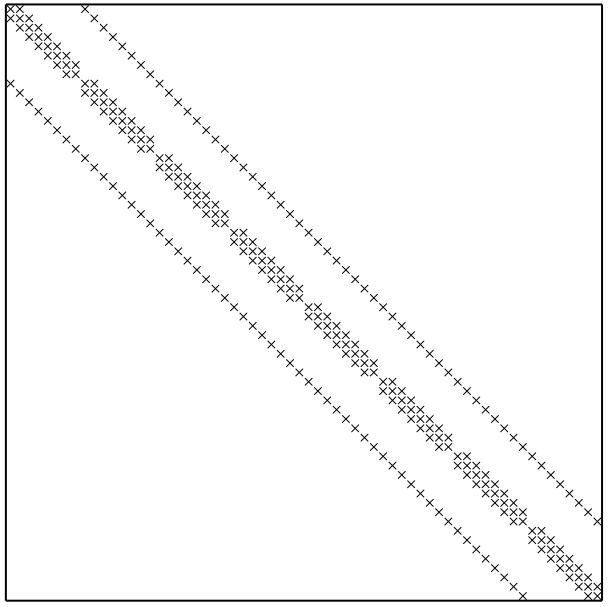
(1) (2) (3) (4)

All levels combined:

29	55	30	62	31	56	32	64
45	25	46	26	47	27	48	28
21	59	22	53	23	60	24	54
41	17	42	18	43	19	44	20
13	51	14	63	15	52	16	61
37	9	38	10	39	11	40	12
5	57	6	49	7	58	8	50
33	1	34	2	35	3	36	4

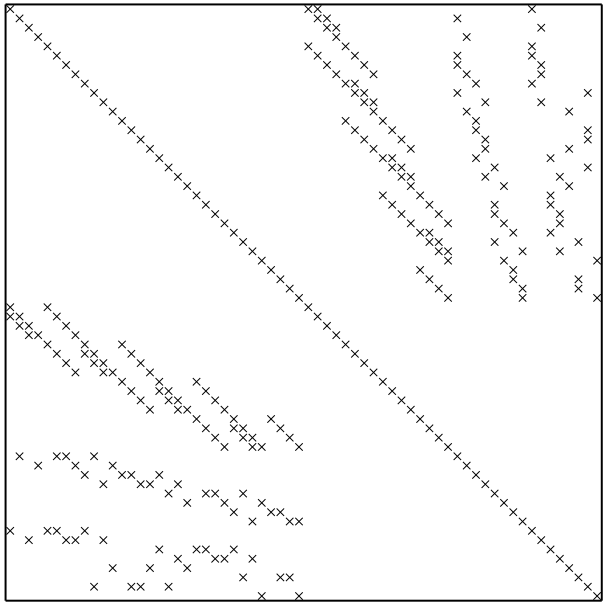
Special ordering (2)

Effect on sparsity pattern of matrix S:



Lexicographic

becomes

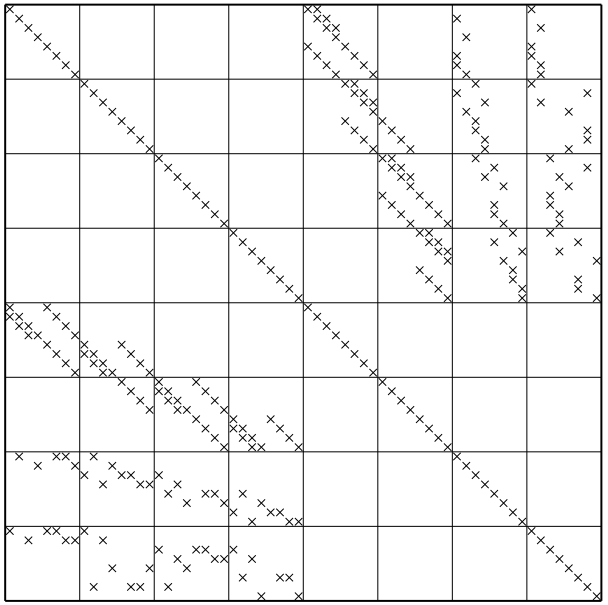


RRB-numbering

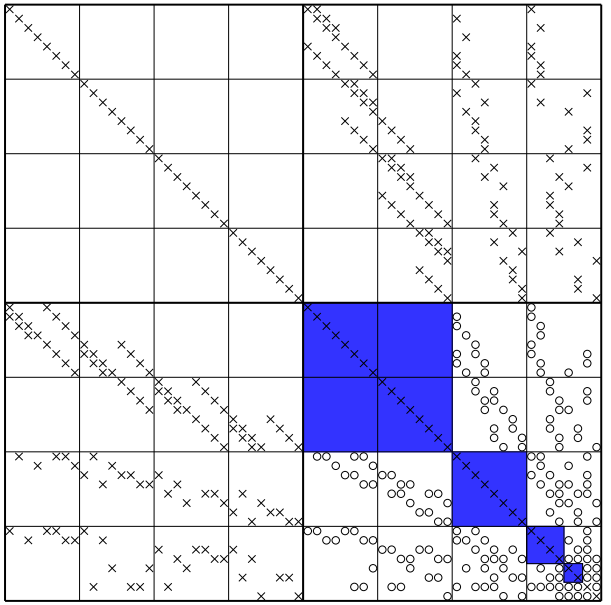
Special ordering (3)

Sparsity pattern of matrix S versus $L + D + L^T$

(recall preconditioner $M = LDL^T$)



becomes



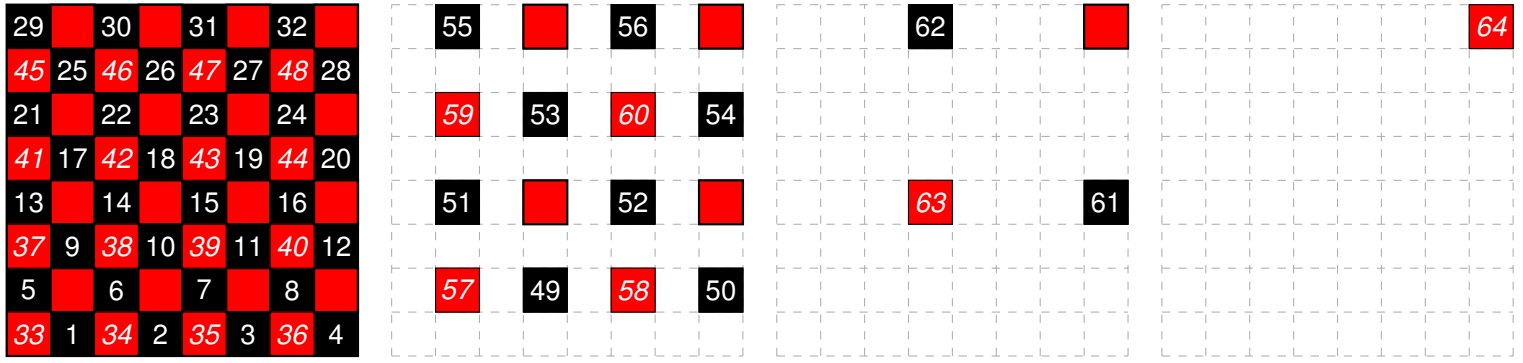
In the blue shaded areas fill-in has been dropped (lumping)

CUDA implementation (1)

Besides the typical Multigrid issues such as idle cores on the coarsest levels, in CUDA the main problem was getting “coalesced memory transfers”.

Why is that?

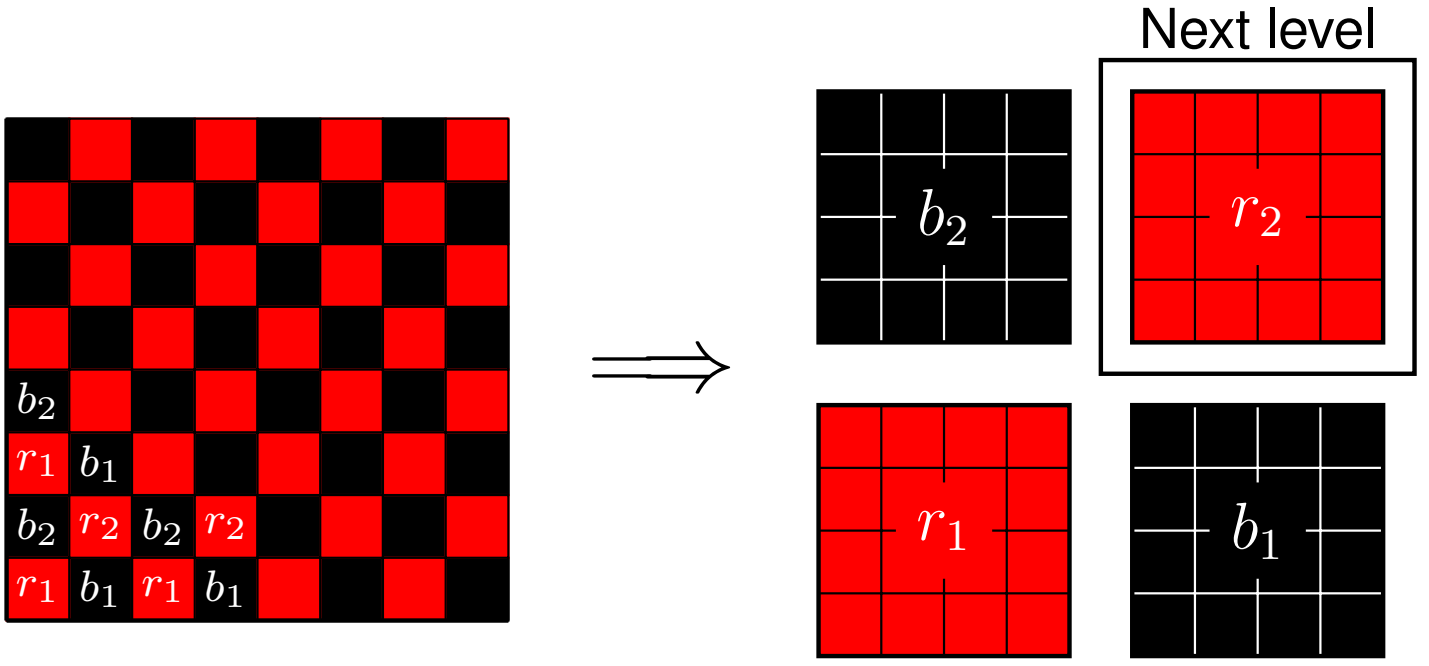
Recall the RRB-numbering: the number of nodes becomes 4$\times$ smaller on every next level:



CUDA implementation (2)

New storage scheme: $r_1/r_2/b_1/b_2$

Nodes are divided into four groups:



CUDA implementation (3)

The $r_1/r_2/b_1/b_2$-storage scheme

- is applied on every next coarser level till the point that the remaining level is smaller than 32×32 elements; the last levels are solved in one go on 1 streaming multiprocessor (SM) exploiting the benefits of cache
- almost comes for free (only at the beginning and ending of CG we have some overhead due to reordering of the data)
- allows for coalesced memory read and write operations throughout the entire CG algorithm which yields optimal throughput

Test problems



- Including: 2D Poisson, Gelderse IJssel (NL), Plymouth Sound (UK)
- Realistic domains up to 1.5 million nodes

Testing method

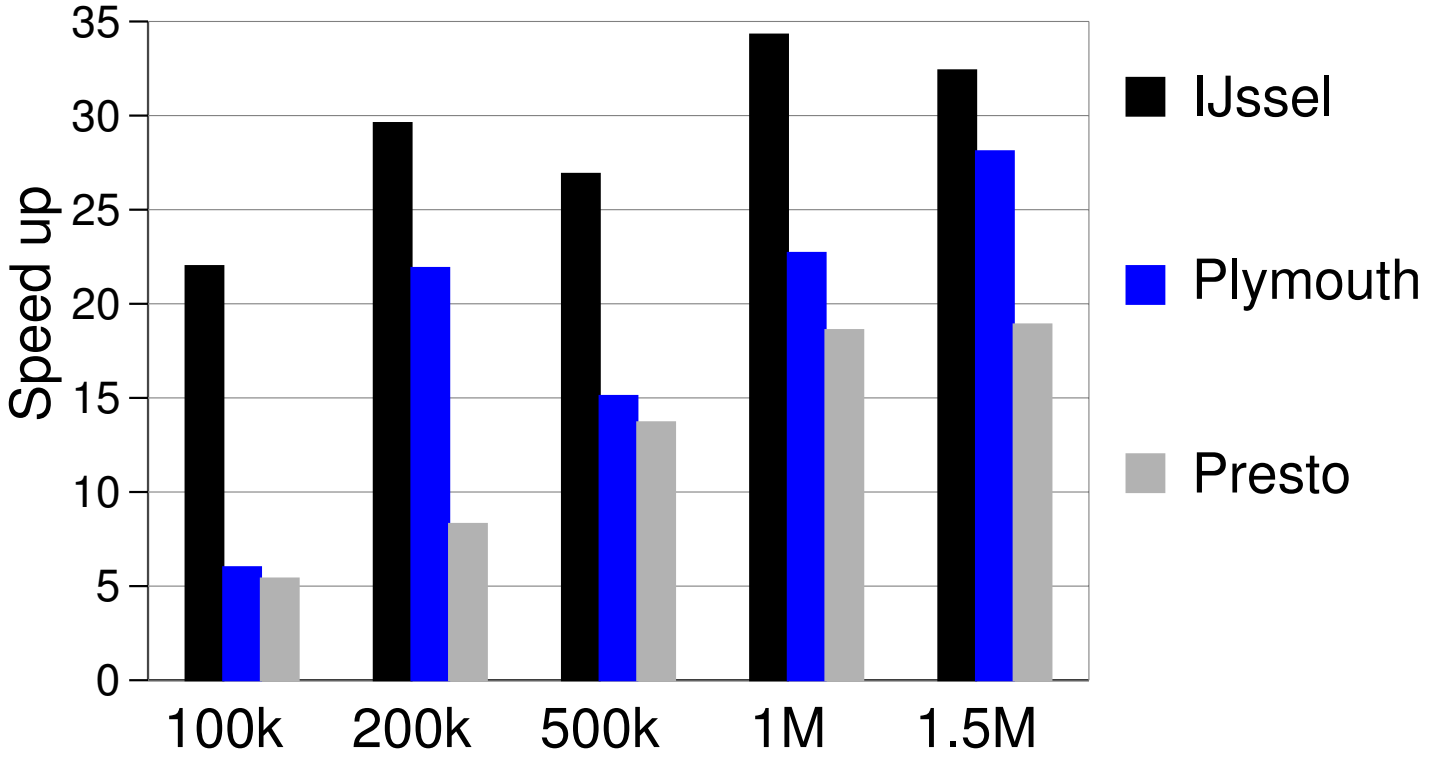
- The 2D Poisson problem was used for throughput analysis of the CUDA RRB-solver
- The realistic test problems were used for detailed time measurements, number of CG-iterations, etc.
- Optimized C++ RRB-solver on 1 core of CPU (Xeon W3520 @ 2.67 GHz)
- CUDA RRB-solver on all cores of GPU (GeForce GTX 580)

Results (1)

	IJssel		Plymouth		Presto	
Size	CUDA	C++	CUDA	C++	CUDA	C++
100k	2.1	47.3	1.9	11.5	1.9	10.3
200k	2.8	83.8	2.6	57.7	2.7	22.3
500k	4.8	130.4	4.7	71.5	4.7	64.6
1M	7.7	266.3	7.9	178.3	7.9	148.0
1.5M	10.7	347.3	10.6	298.5	11.6	219.0

Average time in ms (1000 runs).

Results (2)



Speed up numbers for the realistic test problems.

4. Conclusions

- ILU type preconditioners can be used on GPU's by a Neumann series approach
- Deflation type preconditioners are very suitable for GPU's
- The combination of Neumann series and Deflation preconditioners leads to robust and fast solvers on the GPU
- A special ordering of a red black reordering can lead to speedup of a factor 30-40 on the GPU.

Main question

Can ILU preconditioners be combined with GPU's?

Main question

Can ILU preconditioners be combined with GPU's?

YES

References

- H. Knibbe and C.W. Oosterlee and C. Vuik GPU implementation of a Helmholtz Krylov solver preconditioned by a shifted Laplace multigrid method Journal of Computational and Applied Mathematics, 236, pp. 281-293, 2011
- R. Gupta, M.B. van Gijzen and C. Vuik 3D Bubbly Flow Simulation on the GPU - Iterative Solution of a Linear System Using Sub-domain and Level-Set Deflation, 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2013, ISBN 978-1-4673-5321-2, pp. 359-366, 2013
  http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6498576
- M. de Jong Developing a CUDA solver for large sparse matrices for MARIN, MSc Thesis, Delft University of Technology, 2012
  http://ta.twi.tudelft.nl/nw/users/vuik/numanal/jong_afst.pdf