High Performance Computing Numerical Linear Algebra

Kees Vuik

Delft University of Technology

7th International Workshop CoDesign 2017
October 20-21, 2017
HeFei City, China

TU Delft

Affiliation Kees Vuik

- Professor of Numerical Analysis
- Director of the TU Delft Institute of Computational Science and Engineering
- Scientific Director of 4TU.AMI Applied Mathematics Insitute

TU Delft

TU *D*elft Institute for *C*omputational *Science* and *Engineering*

TU Delft

DCSE

Enabling Technology for Industry

Content

- Introduction
- Problem and solution methods
- Building blocks
- Implementation on Modern Hardware
- Conclusions

1. Introduction: Large linear systems

- Applications

  Computational Fluid Dynamics

  Helmholtz problems (seismic, MRI)

  Power Networks (energy)

- Methods

  Multigrid Methods

  Krylov Subspace Methods

  Domain Decomposition Methods

Example Linear Systems

- Ruede et al, 2015

Nodes	Threads	DoFs	iter	time	time w.c.g.	time c.g. in %
5	80	$2.7 \cdot 10^{9}$	10	685.88	678.77	1.04
40	640	$2.1 \cdot 10^{10}$	10	703.69	686.24	2.48
320	5 120	$1.2 \cdot 10^{11}$	10	741.86	709.88	4.31
2 560	40 960	$1.7 \cdot 10^{12}$	9	720.24	671.63	6.75
20 480	327 680	$1.1 \cdot 10^{13}$	9	776.09	681.91	12.14

Table 2. Weak scaling results with and without coarse grid for the spherical shell geometry.

Introduction: Eigenvalue Problems

- Applications

  Stability of a structure

  Resonance

  Google PageRank

- Methods

  Power method (google)

  Arnoldi

  Jacobi Davidson

Example eigenvalues (Ipsen)

Statistics

- Google indexes 10s of billions of web pages
- “3 times more than any competitor”
- Google serves ≥ 200 million queries per day
- Each query processed by ≥ 1000 machines
- All search engines combined serve a total of ≥ 500 million queries per day

[Desikan, 26 October 2006]

Example eigenvalues

Computation of PageRank

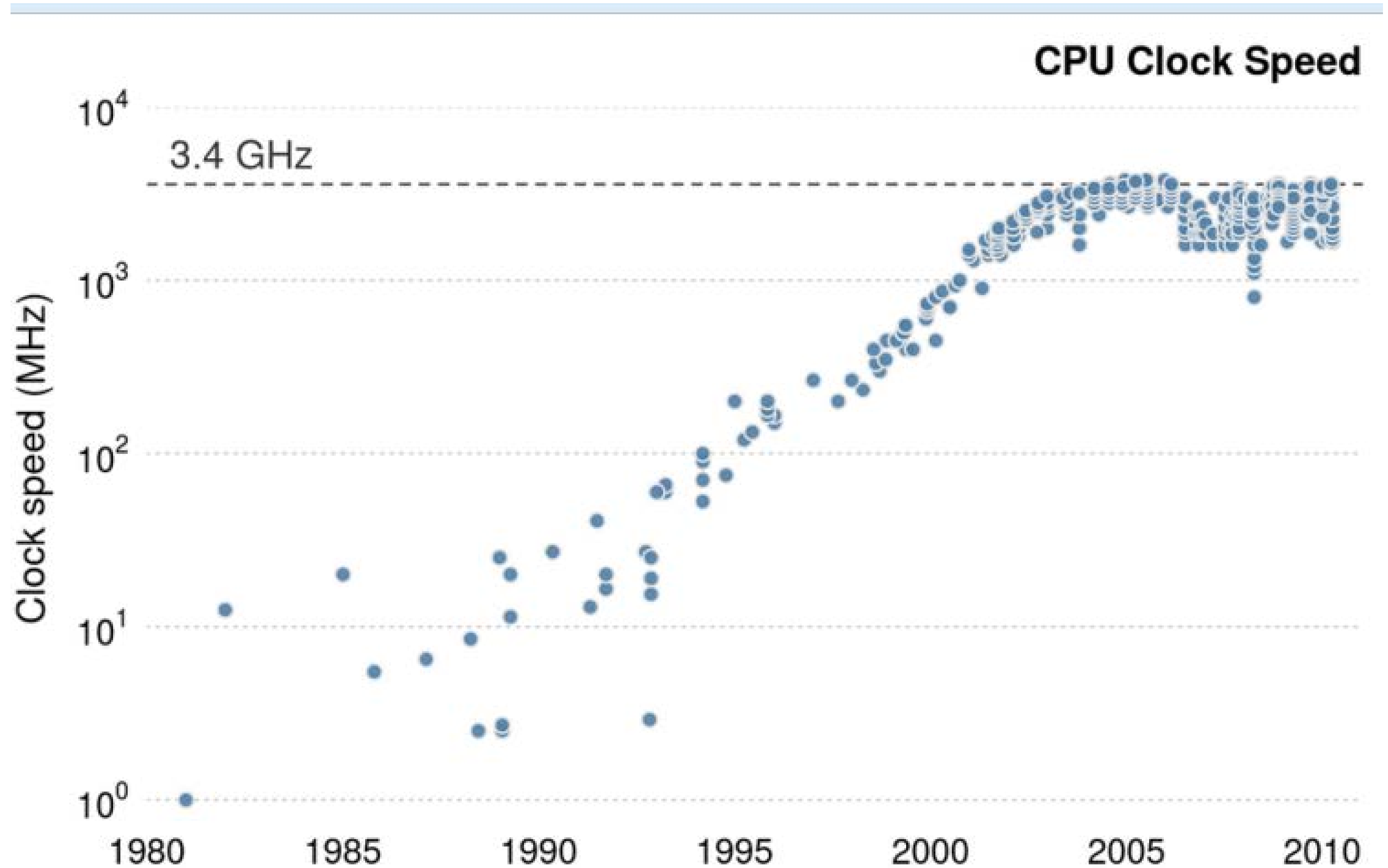
The world's largest matrix computation
[Moler 2002]

- Eigenvector
- Matrix dimension is 10s of billions
- The matrix changes often
  250,000 new domain names every day
- Fortunately: Matrix is sparse

TU Delft

2. Problem and solution methods

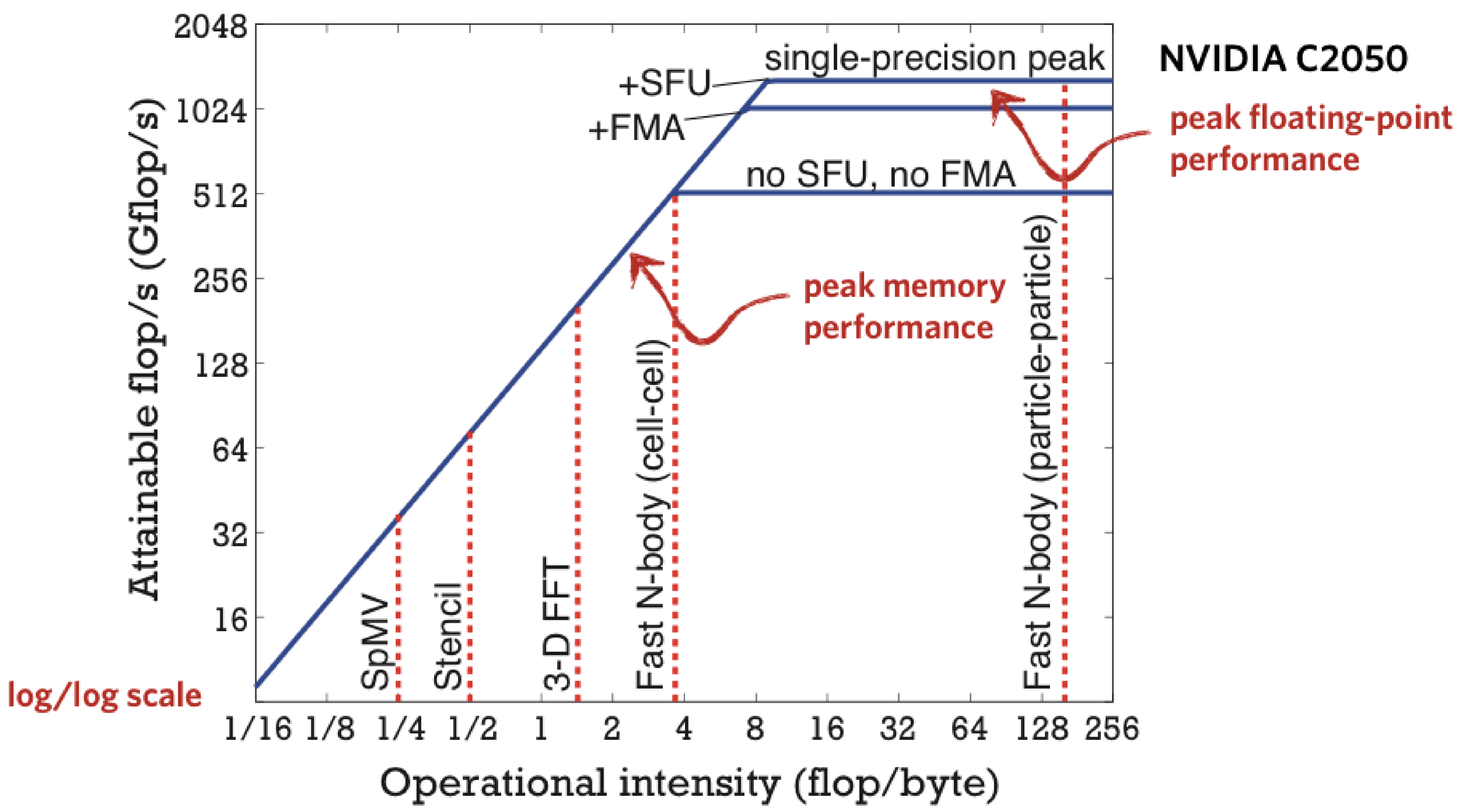
- Computational speed stagnates
- Parallel computing
- Fine grain
- Coarse grain
- Balance between flops and memory



TU Delft

DCSE
Enabling Technology for Industry

Roofline model



DCSE
Enabling Technology for Industry

New Hardware

- Multi-core CPU
- Clusters of PC’s
- GPU *graphics processing unit*
- Xeon Phi
- FPGA field-programmable gate array
- Quantum Computing

  (TU Delft: QuTech https://qutech.nl/)

Data distribution

- Domain decomposition
- Data distribution
- Parallel computation (load balancing)
- Computation
- Communication (local, global)
- Scalability

Domain decomposition

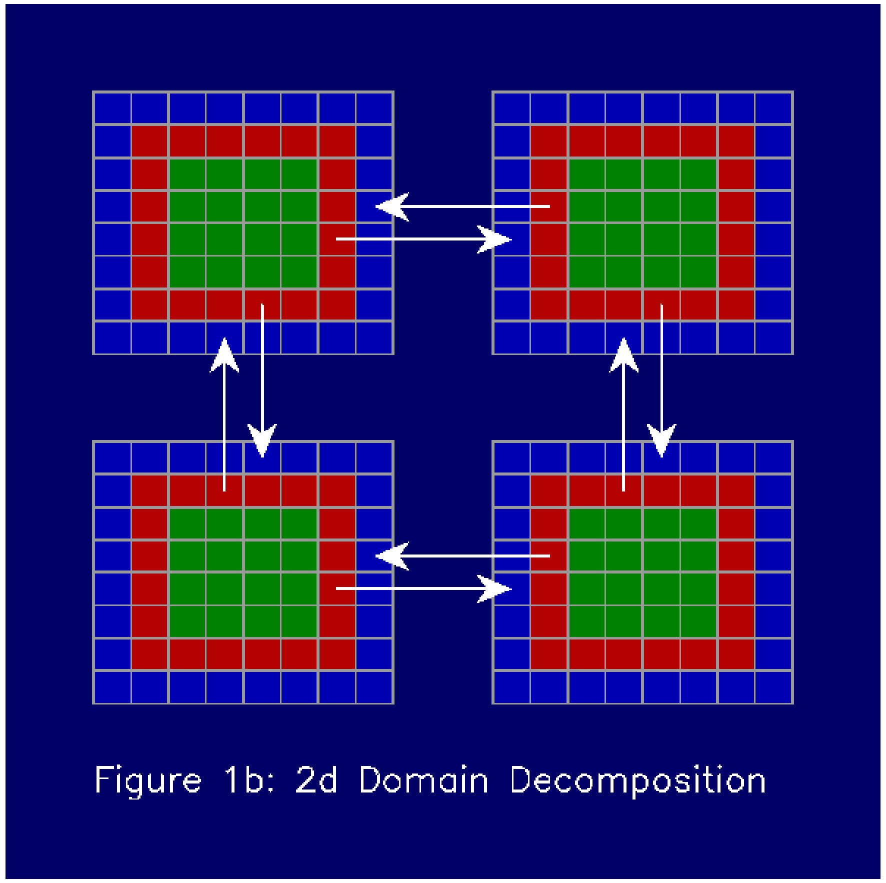
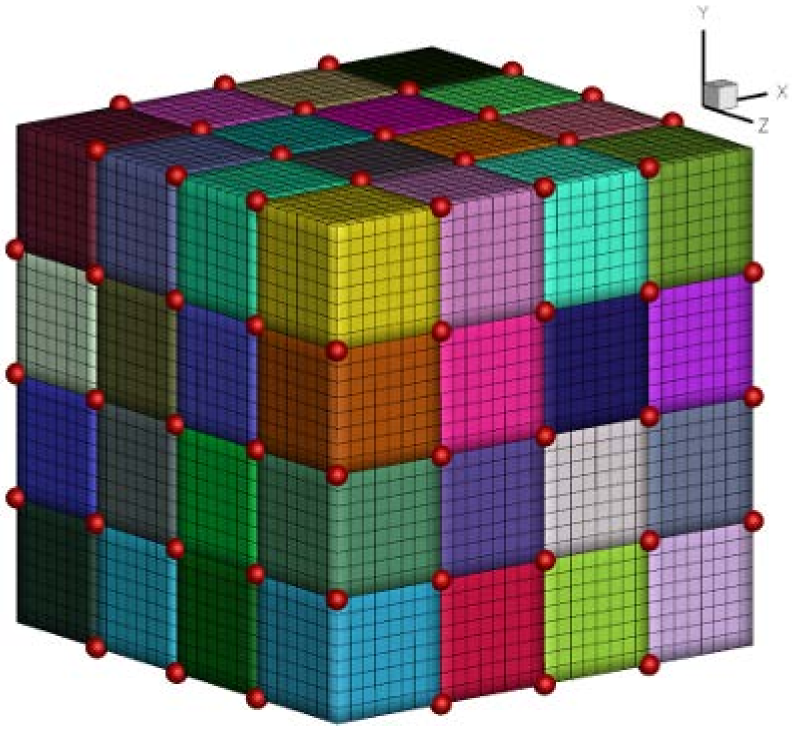
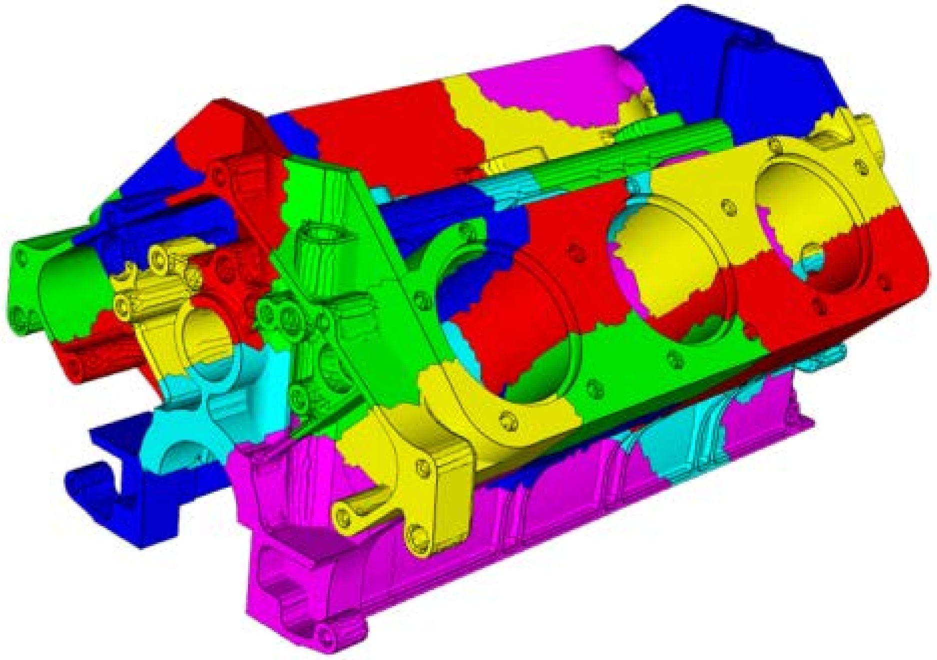


Figure 1b: 2d Domain Decomposition





3. Building blocks

Iterative methods and eigenvalues

- Vector update
- Inner product
- Matrix vector product
- Preconditioner construction
- Preconditioner vector product

Vector update

- Embarrassingly parallel

for i= 1 : nblocks

x(1:n) = x(1:n) + $\alpha * $ y(1:n)

end

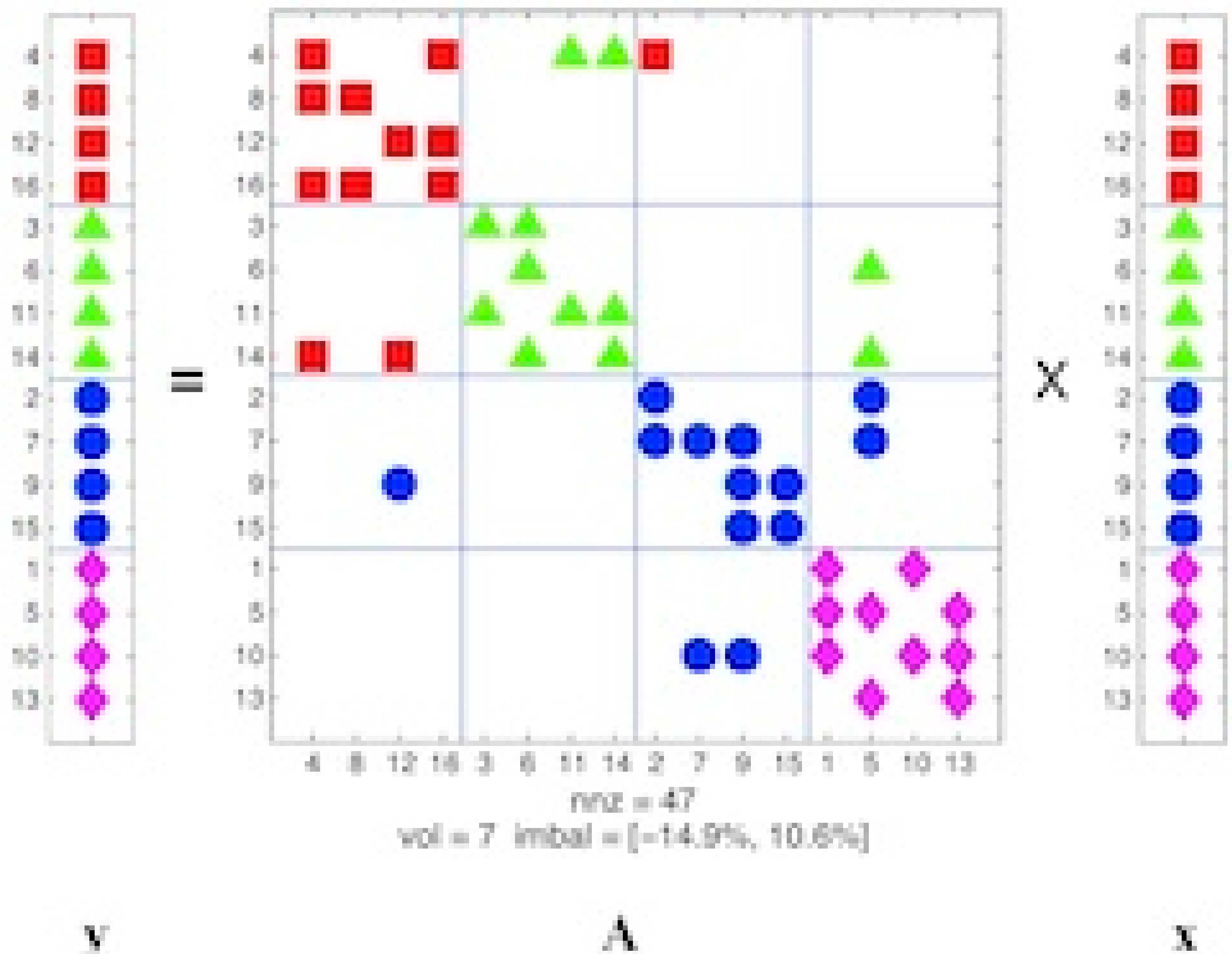
Inner product

- Local inner products (fully parallel)
- Global sum
- Global communication

this can be a serious bottleneck

Matrix vector product

- Structured
- Unstructured
- Nearest neighbour communication



Preconditioning

- Diagonal preconditioning
- Basic Iterative Method
- ILU-type preconditioning
- Multi-grid preconditioning
- Block preconditioning
- Operator preconditioning

Second level preconditioning

Due to parallel preconditioning the connection between the domains is lost.

Second level preconditioning repairs this.

- Coarse Grid Correction
- Multilevel preconditioning
- Deflation
- Balanced Neumann-Neumann

4. Implementation on Modern Hardware

Overview of techniques

Some additional remarks

Outlook to the future

Multi-core CPU

- Staircase Incomplete Choleski Preconditioner

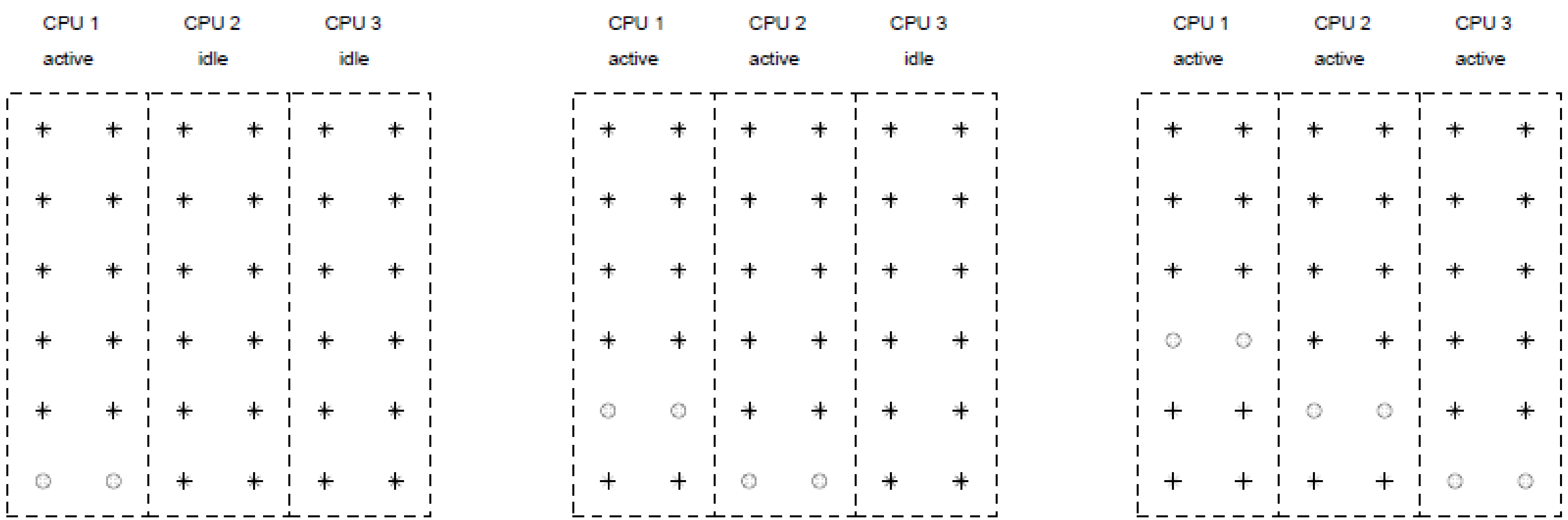


Figure 29: The first stages of the staircase parallel solution of the lower triangular system $Lx = b$. The symbols denote the following: $*$ nodes to be calculated, o nodes being calculated, and + nodes that have been calculated.

Multi-core CPU

- Mechanical problem
- Domain decomposition using physical properties into account
- Good speedup on 8-16 cores

F.J. Lingen, P.G. Bonnier, R.B.J. Brinkgreve, M.B. van Gijzen, and C. Vuik
A parallel linear solver exploiting the physical properties of the underlying mechanical problem
Computational Geosciences, 18, pp. 913-926, 2014

Clusters of PC's

T.B. Jonsthovel, M.B. van Gijzen, C.Vuik, and A. Scarpas
On the Use of Rigid Body Modes in the Deflated Preconditioned Conjugate Gradient Method
SIAM Journal on Scientific Computing, 35, p.B207-B225, 2013

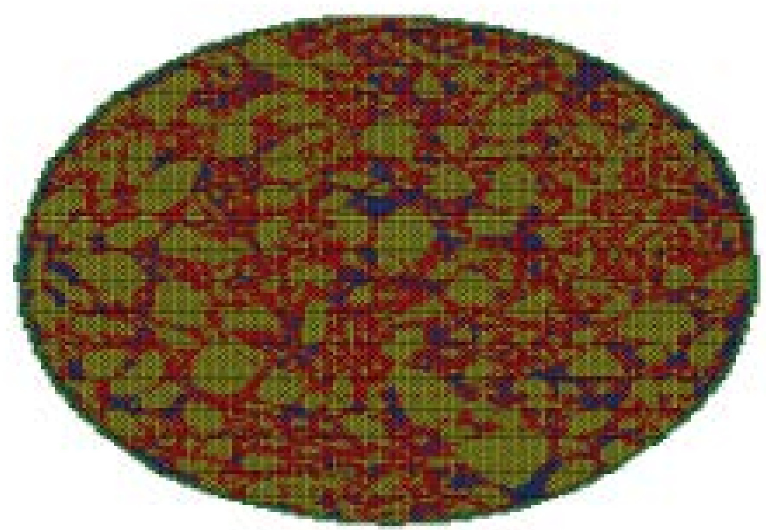
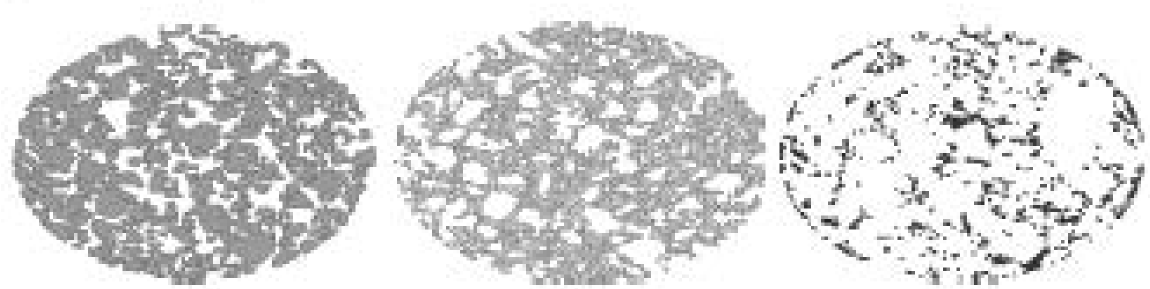


FIG. 7. *FE mesh representing core of asphaltic material containing aggregates (yellow), bitumen (red), and air voids (blue).*



(a) (b) (c)

FIG. 8. *Deflation strategy, identify sets of elements corresponding to material: (a) aggregates, (b) bitumen, and (c) air voids.*

TU Delft

DCSE
Enabling Technology for Industry

Clusters of PC's

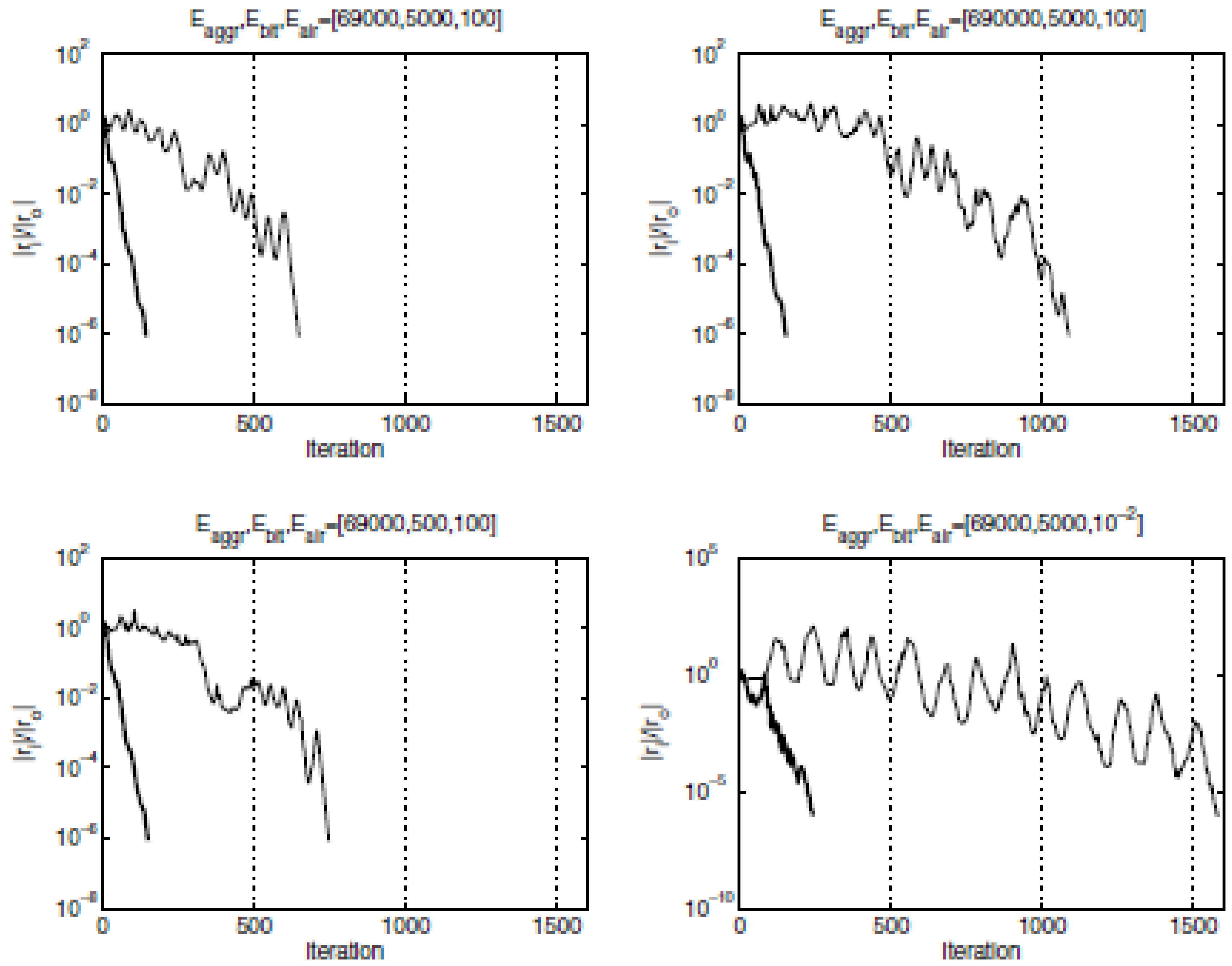


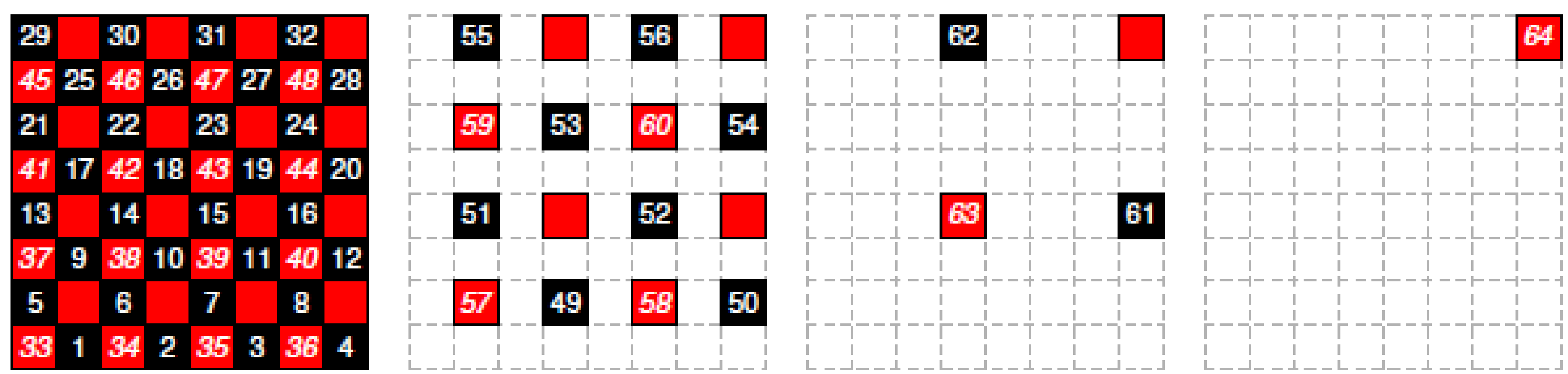
FIG. 6. *Convergence of PCG and DPCG (bold line) for cylinder containing three aggregates.*

GPU graphics processing unit

M. de Jong and C. Vuik
GPU Implementation of the RRB-solver
TU Delft Report 16-06

Special ordering

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

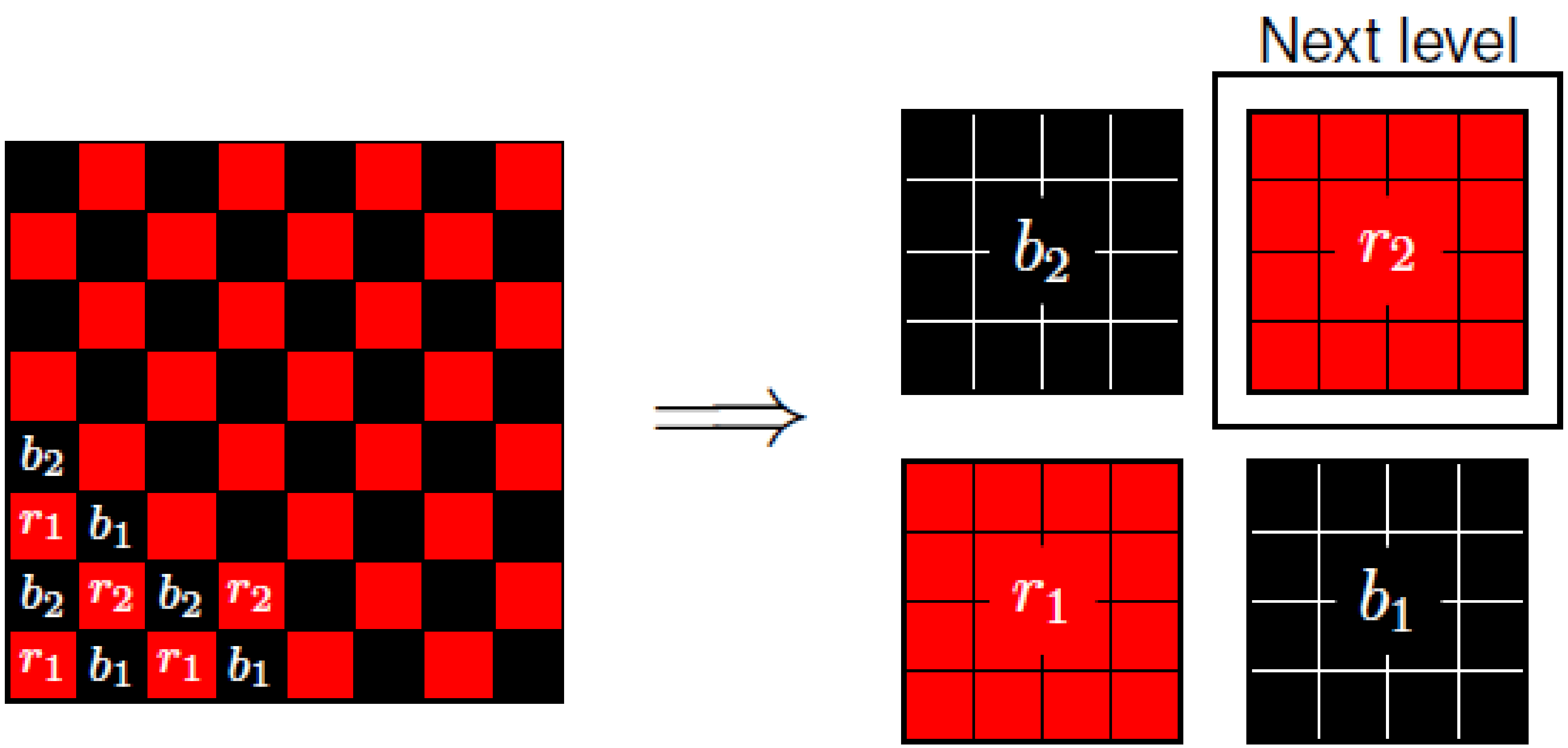
TU Delft

ıdustry

CUDA implementation (2)

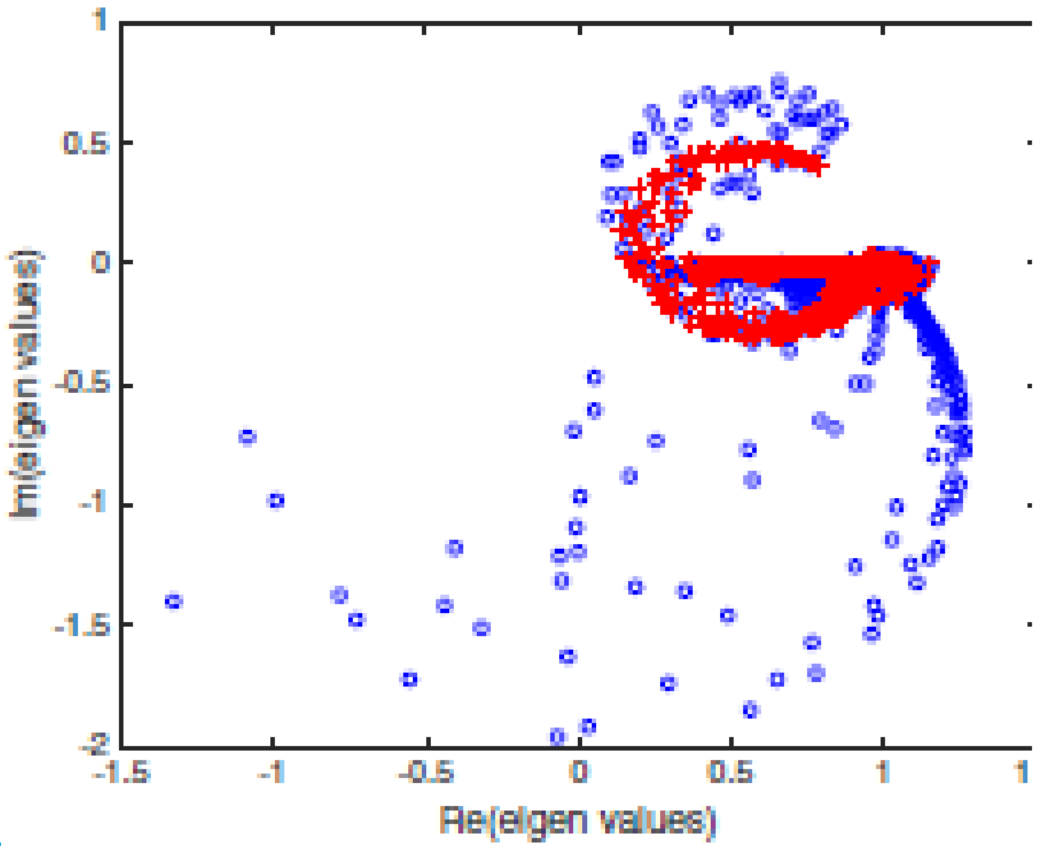
New storage scheme: $r_1/r_2/b_1/b_2$

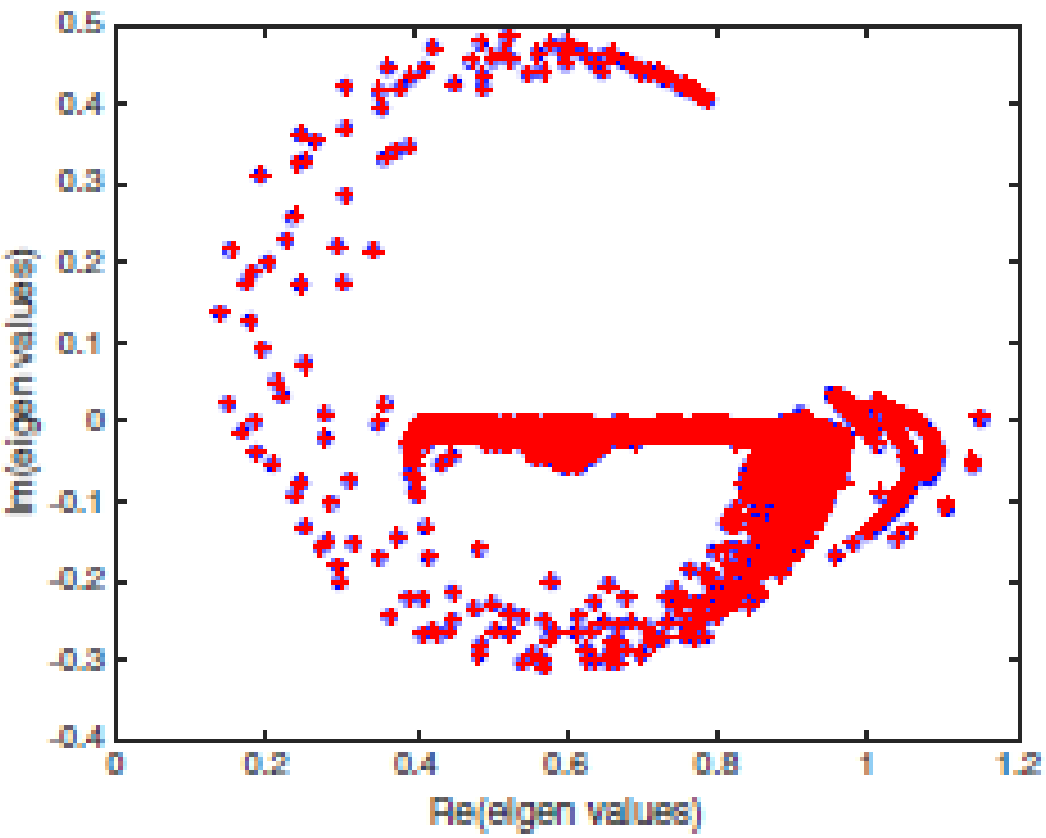
Nodes are divided into four groups:



GPU graphics processing unit

H. Knibbe, C. Vuik, and C.W. Oosterlee
Reduction of computing time for least-squares migration based on the Helmholtz equation by graphics processing units
Computational Geosciences, 20, pp. 297-315, 2016





TU Delft

DCSE
Enabling Technology for Industry

FPGA field-programmable gate array

- Data flow machine
- Hard and software programming
- Maxeler
- Altera
- Xilinx

FPGA field-programmable gate array

Implementation of the BiCGSTAB Method for the Helmholtz Equation on a Maxeler Data Flow Machine
Onno Meijers
MSc Thesis, TU Delft, 2017

http://ta.twi.tudelft.nl/nw/users/vuik/numanal/meijers_afst.pdf

Figure 3: A MaxWorkstation.

TU Delft

DCSE
Enabling Technology for Industry

FPGA field-programmable gate array

Algorithm 4.1 BiCGSTAB for implementation.

$\mathbf{u} = 0$; $\bar{\mathbf{r}} = \mathbf{r} = \mathbf{g} = \delta_{x_s,y_s,z_s}$; $\rho_{old} = \alpha = \omega = 1$; $\mathbf{v} = \mathbf{p} = 0$;
for $i = 0, 1, 2, \dots, maxit$ **do**
  $\rho_{new} = \mathbf{r}(x_s, y_s, z_s)$; $\beta = \frac{\rho_{new}}{\rho_{old}} \frac{\alpha}{\omega}$; $\rho_{old} = \rho_{new}$;
  $\mathbf{p} = \mathbf{r} + \beta(\mathbf{p} - \omega \mathbf{v})$;
  Solve $M\hat{\mathbf{p}} = \mathbf{p}$;
  $\mathbf{v} = A\hat{\mathbf{p}}$;
  $\alpha = \frac{\rho_{old}}{\mathbf{v}(x_s,y_s,z_s)}$;
  $\mathbf{s} = \mathbf{r} - \alpha \mathbf{v}$;
  if $||\mathbf{s}||$ small enough **then**
    $\mathbf{u} = \mathbf{u} + \alpha\hat{\mathbf{p}}$; quit;
  end if
  Solve $M\mathbf{z} = \mathbf{s}$;
  $\mathbf{t} = A\mathbf{z}$;
  $\omega = \frac{(\mathbf{t},\mathbf{s})}{(\mathbf{t},\mathbf{t})}$;
  $\mathbf{u} = \mathbf{u} + \alpha\hat{\mathbf{p}} + \omega\mathbf{z}$;
  $\mathbf{r} = \mathbf{s} - \omega\mathbf{t}$;
  if $||\mathbf{r}||$ is small enough **then**
    quit;
  end if
end for

FPGA field-programmable gate array

Algorithm 4.2 BiCGSTAB after implementation

$\mathbf{u} = \mathbf{0};\ \ \mathbf{r}_0 = \mathbf{r} = \mathbf{g} = \delta_{x_s,y_s,z_s};\ \rho_{old} = \alpha = \omega = \rho_{new} = 1;\ \mathbf{v} = \mathbf{p} = \mathbf{0};$
for $i = 0, 1, 2, \ldots, maxit$ **do**
  Part 0: $\beta = \frac{\rho_{new}}{\rho_{old}} \frac{\alpha}{\omega};\ \ \rho_{old} = \rho_{new};$
  Part 1: $\mathbf{p} = \mathbf{r} + \beta(\mathbf{p} - \omega\mathbf{v});$
  Part 1: $\mathbf{v} = A\mathbf{p};$
  Part 1: $\alpha = \frac{\rho_{old}}{(\mathbf{v},\mathbf{r}_0)};$
  Part 2: $\mathbf{s} = \mathbf{r} - \alpha\mathbf{v};$
  Part 2: $\mathbf{t} = A\mathbf{s};$
  Part 2: $\omega = \frac{(\mathbf{t},\mathbf{s})}{(\mathbf{t},\mathbf{t})};$
  Part 3: $\mathbf{u} = \mathbf{u} + \alpha\mathbf{p} + \omega\mathbf{s};$
  Part 3: $\mathbf{r} = \mathbf{s} - \omega\mathbf{t};$
  Part 3: $\rho_{new} = (\mathbf{r}, \mathbf{r}_0);$
  if $||\mathbf{r}||$ is small enough **then**
    quit;
  end if
end for

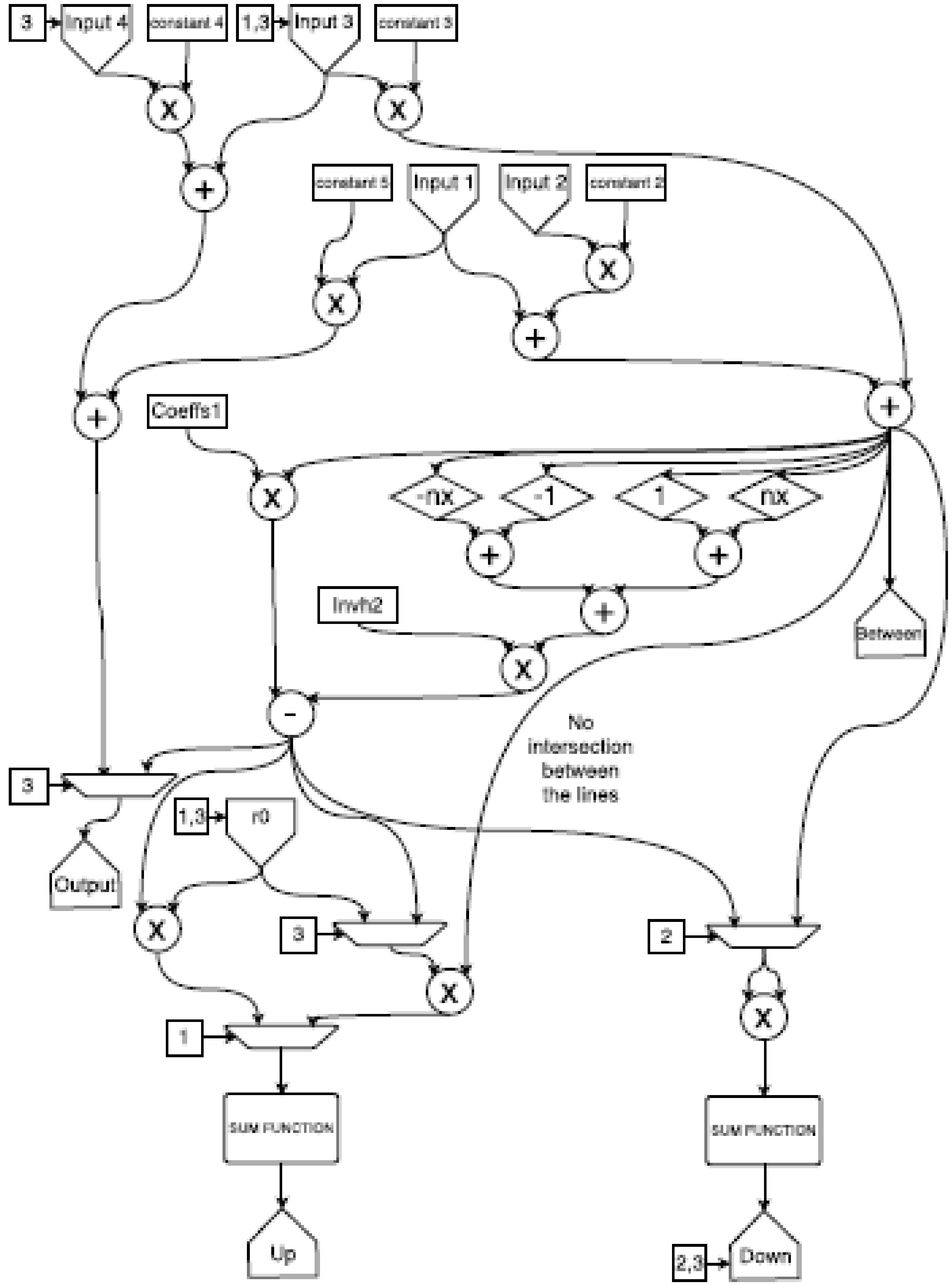


Figure 6: Simple representation of the data flow graph of the Kernel.

TU Delft

DCSE
Enabling Technology for Industry

Quantum Computing

arXiv.org > cs > arXiv:1705.07413

Search or Article ID inside arXiv | All papers | Broaden you

(Help | Advanced search)

Computer Science > Computational Engineering, Finance, and Science

On the impact of quantum computing technology on future developments in high-performance scientific computing

Matthias Möller, Cornelis Vuik

(Submitted on 21 May 2017 (v1), last revised 19 Jun 2017 (this version, v2))

Quantum computing technologies have become a hot topic in academia and industry receiving much attention and financial support from all sides. Building a quantum computer that can be used practically is in itself an outstanding challenge that has become the 'new race to the moon'. Next to researchers and vendors of future computing technologies, national authorities are showing strong interest in maturing this technology due to its known potential to break many of today's encryption techniques, which would have significant impact on our society. It is however quite likely that quantum computing has beneficial impact on many computational disciplines.

TU Delft

DCSE
Enabling Technology for Industry

Conclusions

- CPU’s will not be faster
- Heterogeneous computing is the future
- More parallelism
- Various precisions used
- Special algorithms needed
- Optimization of code is important
- Decrease memory access