CGMN and CARP-CG on clusters of multi-core CPUs

Jonas Thies

German Aerospace Center (DLR)
Simulation and Software Technology
High Performance Computing group
Jonas.Thies@DLR.de

DLR

Outline

Motivation

Overview of this talk

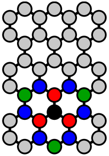
Have you met a linear system that didn't want to be solved iteratively?

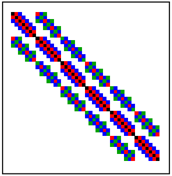
Aims of this talk:

- bring CGMN to your attention
- show two competing parallelization schemes
- discuss implementation aspects

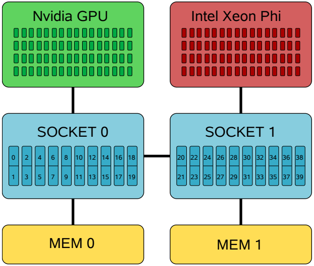
Example application that is 'hard to precondition'

- Graphene: Carbon atoms in a 2D hexagonal mesh
- Hamiltonian: random diagonal entries $|a_{ii}| < |a_{i \neq j}|$
- symmetric and completely indefinite
- Task: find 10-1000 innermost eigenpairs





Hardware challenges for solvers



- multi-level parallelism
- heterogenous hardware
- complex memory/cache hierarchy
- resilience: fast recovery if a node fails (using some flavor of checkpoint/restart)

Drawbacks of common parallel preconditioners

- Domain decomposition methods (FETI, Schwarz+ILU etc)
  - high memory demands (bandwidth bottleneck)
  - resilience: checkpointing the preconditioner not practical
  - load balancing: only static
- AMG
  - limited to certain problem classes (e.g. elliptic PDEs)
  - setup phase complex and hard to parallelize
  - needs parallel smoother
  - more communication on coarser grids

The CGMN algorithm

Kaczmarz iteration

- SOR on the minimum norm problem (MNP),

$$\mathbf{A}\mathbf{A}^T \mathrm{y} = \mathrm{b}, \mathrm{x} = \mathbf{A}^T \mathrm{y}.$$

- equivalent to Kaczmarz iteration for $\mathbf{A}\mathrm{x} = \mathrm{b}$ (KACZ)
- forward + backward KACZ $\Longrightarrow$ SSOR on the MNP

$$\begin{aligned} \mathrm{x}^{(k+1)} &= \mathbf{Q}_{SSOR}\mathrm{x}^{(k)} + \mathbf{R}_{SSOR}\,\mathrm{b}, \\ \text{with } \mathbf{Q}_{SSOR} &= Q_1 Q_2 \dots Q_n Q_{n-1} \dots Q_1, \\ Q_i &= \mathbf{I} - \frac{\omega}{||a_{i,:}^H||^2} a_{i,:}^H a_{i,:} \end{aligned}$$

- Q_i: projections onto i'th row $a_{i,:}$ of $\mathbf{A}$

CGMN (Björck & Elfving, 1979)

- CG for $(\mathbf{I} - \mathbf{Q}_{SSOR})x = \mathbf{R}_{SSOR}\, b$ converges even though the system matrix is only symmetric positive semi-definite.
- implicit SSOR preconditioning
- efficient row-wise formulation
- extremely robust: A may be non-symmetric, singular, non-square etc.
- row scaling aleviates issue of 'squared condition number'

Core operation: KACZ sweep (in CRS format)

spMVM, $y \leftarrow \mathbf{A}x$

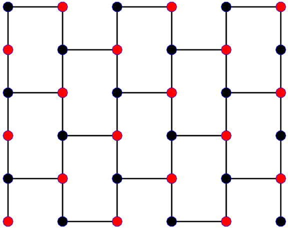
```
for (i=0; i<n; i++) do
    tmp=0
    for (j=rptr[i]; j<rptr[i+1]; j++) do
        tmp+=val[j]*x[col[j]];
    end for
// non-temporal store
    y[j]=tmp;
end for
```

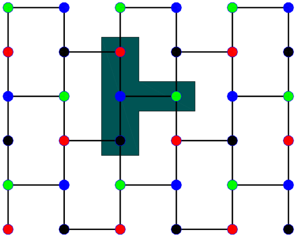
Kaczmarz update, $x \leftarrow KACZ(\mathbf{A}, x, b, \omega)$

```
for (i=0; i<n; i++) do
    tmp=-b[i]; // b!=0 only in 1st iteration
    nrm=0;
    for (j=rptr[i]; j<rptr[i+1]; j++) do
        tmp+=val[j]*x[col[j]];
        nrm+=val[j]*val[j];
    end for
// update x
    tmp*=omega/nrm;
    for (j=rptr[i]; j<rptr[i+1]; j++) do
        x[cols[j]]-=tmp*val[j];
    end for
end for
```

Parallelization

Multi-Coloring (MC)



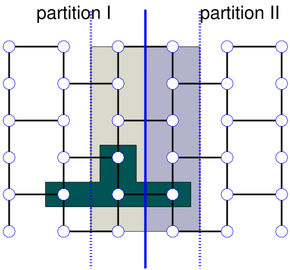


- requires "distance 2" coloring
- software: ColPack
  `http://cscapes.cs.purdue.edu/coloringpage/software.htm`

Component-Averaged Row Projection (CARP)

- Gordon & Gordon, 2005
- Kaczmarz locally
- write to halo
- exchange and average

equiv. to KACZ on a superspace of $\mathbb{R}^n$



Hybrid method: MC_CARP-CG

- global MC would require...
  - an extremely scalable coloring method
  - very well-balanced colors
  - many global sync-points (ca. 15 colors in our examples)
- global CARP requires ...
  - large number of MPI procs
  - increasing amount of 'interior halo elements'
  - non-trivial implemention on GPU and Xeon Phi

Idea: node-local MC with MPI-based CARP between the nodes

Convergence behavior

Experimental setup

- Machine: Intel Xeon "Ivy Bridge"
- 10 cores/socket, 2 sockets/node
- InfiniBand between nodes

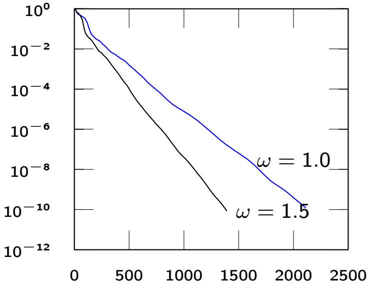
Test cases: conv. dominated PDE, Anderson localization

- 3D 7-point stencil
- octree ordering
- suitable boundary conditions

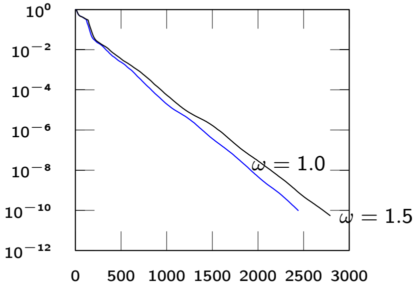
Application 1: convection dominated flow

$$-(e^{-xyz}U_x)_x - (e^{xyz}U_y)_y - (e^{(1-x)(1-y)(1-z)}U_z)_z$$
$$+\frac{40}{\delta x}\sin(\pi y)U_x + \frac{2}{\delta y}\sin(\pi z)U_y + \frac{2}{\delta z}\sin(\pi x)U_z = F$$

CARP-CG



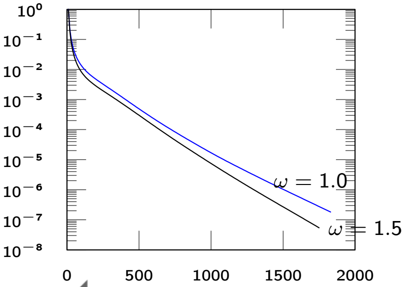
MC_CARP-CG



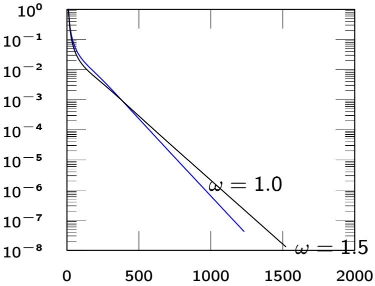
Application 2: Anderson localization

- 7-point stencil
- diag: random numbers from $[-\frac{L}{2}, \frac{L}{2}]$
- off-diagonal elements: -1
- small complex diagonal shift ($10^{-2}i$)

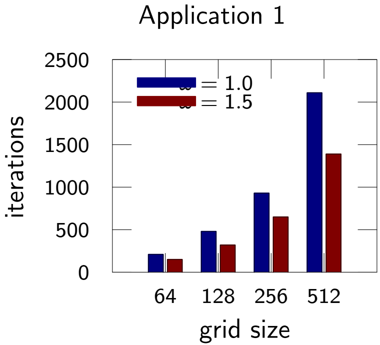
$L = 16.5$

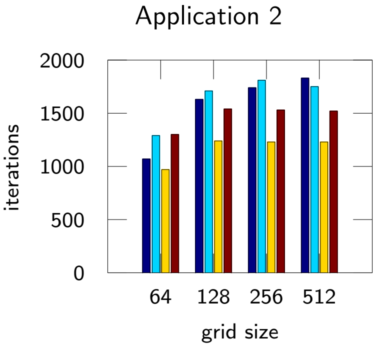


$L = 1.0$



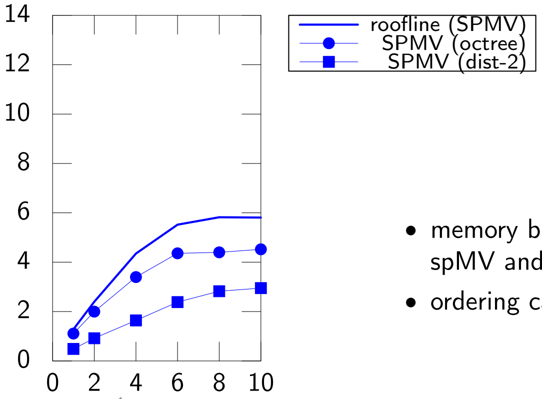
CARP-CG Convergence for increasing problem size





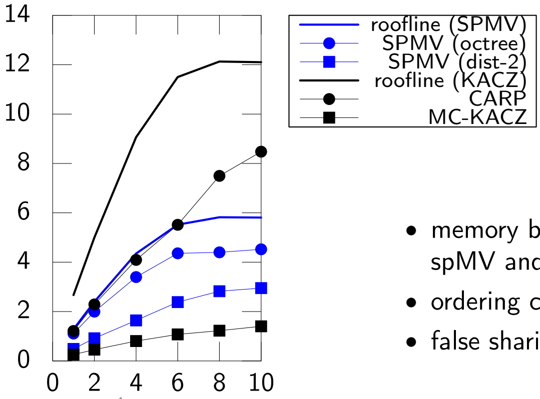
Performance aspects

Performance on a multi-core CPU



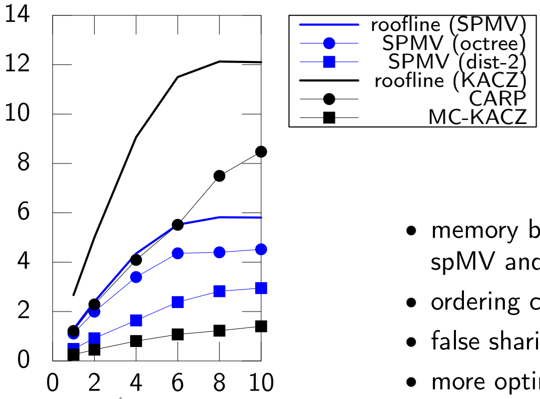
- memory bandwidth limits performance of spMV and KACZ
- ordering causes scattered access to x

Performance on a multi-core CPU



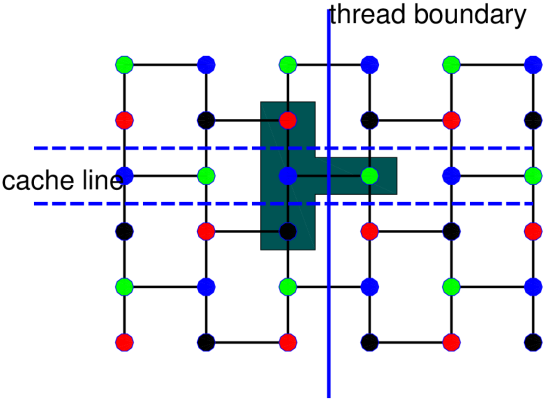
- memory bandwidth limits performance of spMV and KACZ
- ordering causes scattered access to x
- false sharing prevents socket scaling

Performance on a multi-core CPU

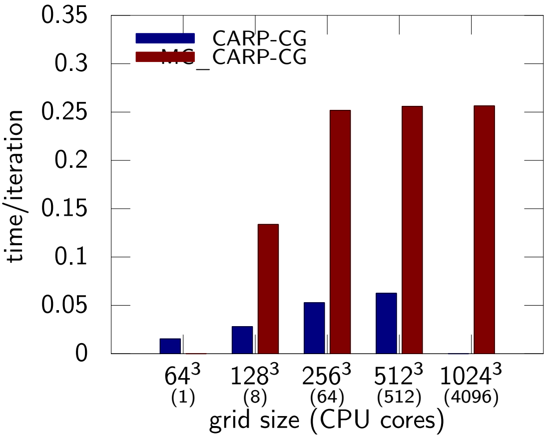


- memory bandwidth limits performance of spMV and KACZ
- ordering causes scattered access to x
- false sharing prevents socket scaling
- more optimizations possible but non-trivial

Performance hazards of multi-coloring approach



Weak scaling (8 cores/socket, 64^3 unknowns/core)



Summary

- CGMN is a useful method for matrices with small diagonal entries
- also useful for e.g. Helmholtz equations
- runs as fast as unpreconditioned CG on one CPU core
- parallelization schemes
  - distance-2 coloring bad for performance and overrelaxation
  - CARP gives very effective domain decomposition
  - but with quite some memory overhead
  - hybrid MC_CARP may be a good choice, e.g. for block methods

Acknowledgement and references

DFG project ESSEX
(Equipping Sparse Solvers for the EXa-scale)

- Björck & Elfving: Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT* 19, pages 145–163, 1979.
- Gordon & Gordon: Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems. *SISC* 27 (3), 2005, pages 1092–1117.