On Level Scheduling for Incomplete LU Factorization Preconditioners on Accelerators

Karl Rupp, Barry Smith
rupp@mcs.anl.gov

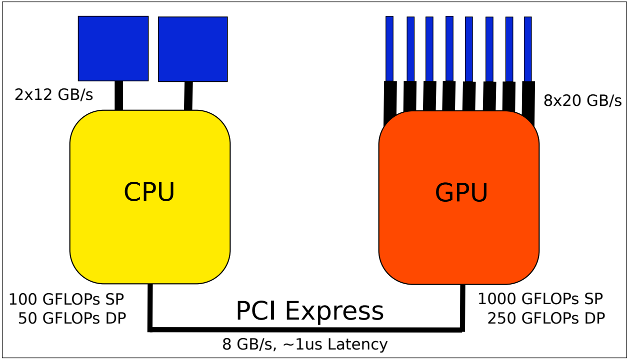
Mathematics and Computer Science Division
Argonne National Laboratory

FEMTEC 2013

May 20th, 2013

Parallel Hardware Constraints

Computing Architecture Schematic

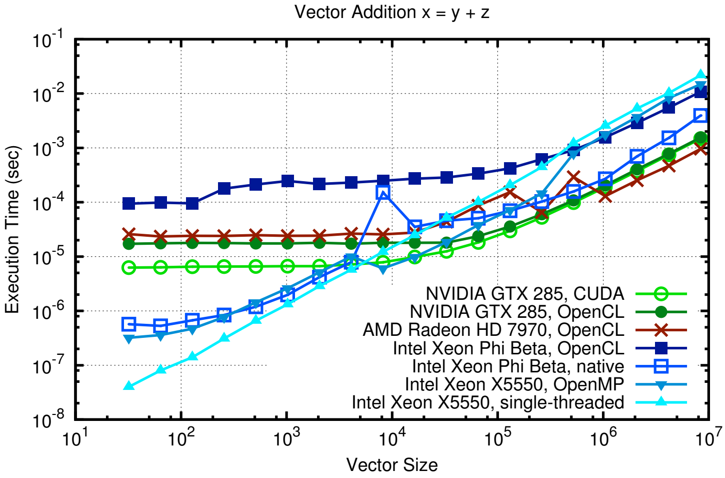


Good for large FLOP-intensive tasks, high memory bandwidth

PCI-Express can be a bottleneck

$\gg 10$-fold speedups (usually) not backed by hardware

Benchmarks


Vector Addition x = y + z
Execution Time (sec)
Vector Size
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}
NVIDIA GTX 285, CUDA
NVIDIA GTX 285, OpenCL
AMD Radeon HD 7970, OpenCL
Intel Xeon Phi Beta, OpenCL
Intel Xeon Phi Beta, native
Intel Xeon X5550, OpenMP
Intel Xeon X5550, single-threaded

Incomplete LU Factorization Preconditioners

Basic Idea

Factor sparse matrix $\boldsymbol{A} \approx \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$

$\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ sparse, triangular

ILU0: Pattern of $\tilde{\boldsymbol{L}}$, $\tilde{\boldsymbol{U}}$ equal to $\boldsymbol{A}$

ILUT: Keep k elements per row

Solver Cycle Phase

Residual correction $\tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}x = z$

Forward solve $\tilde{\boldsymbol{L}}y = z$

Backward solve $\tilde{\boldsymbol{U}}x = y$

Little parallelism in general

$$
\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}
$$

Level Scheduling

Build dependency graph

Substitute as many entries as possible simultaneously

Trade-off: Each step vs. multiple steps in a single kernel

$$\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}$$

Level Scheduling

- Build dependency graph
- Substitute as many entries as possible simultaneously
- Trade-off: Each step vs. multiple steps in a single kernel

$$
\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}
$$

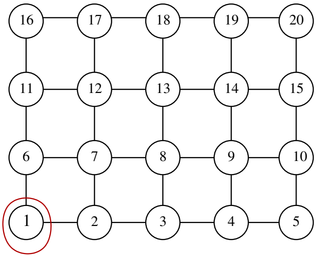
Level Scheduling

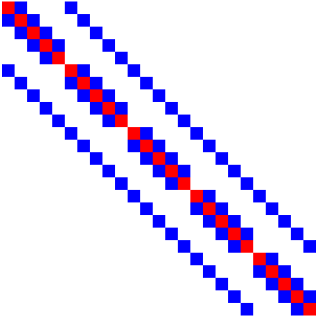
- Build dependency graph
- Substitute as many entries as possible simultaneously
- Trade-off: Each step vs. multiple steps in a single kernel

$$
\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}
$$

Interpretation on Structured Grids

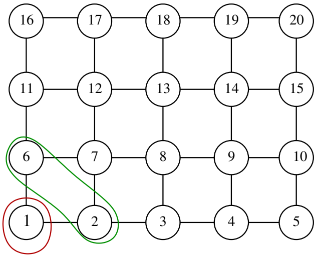
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

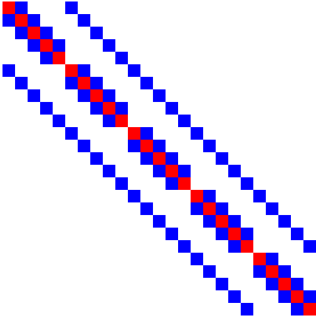




Interpretation on Structured Grids

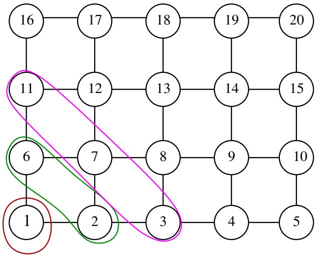
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

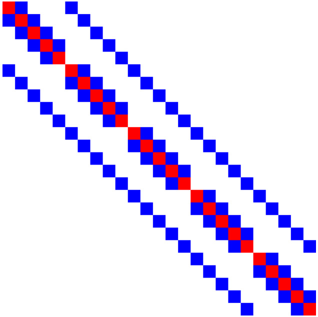




Interpretation on Structured Grids

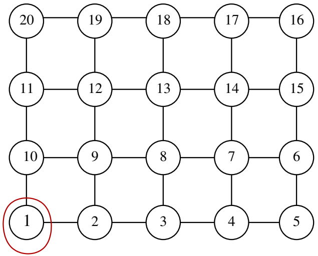
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

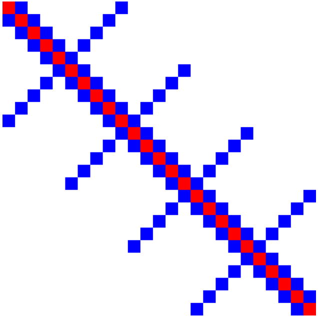




Interpretation on Structured Grids

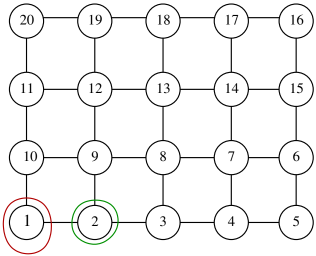
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

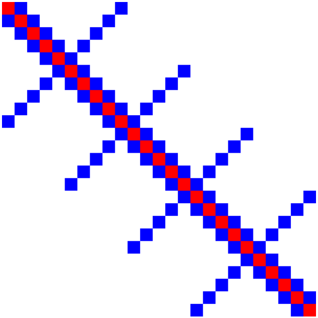




Interpretation on Structured Grids

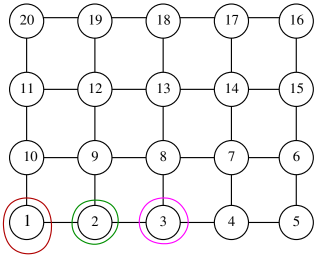
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

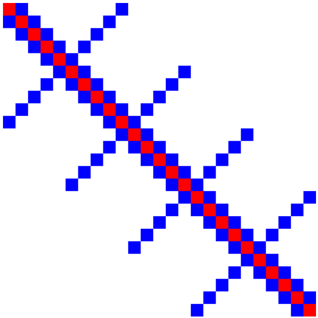




Interpretation on Structured Grids

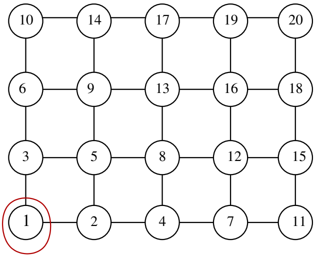
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

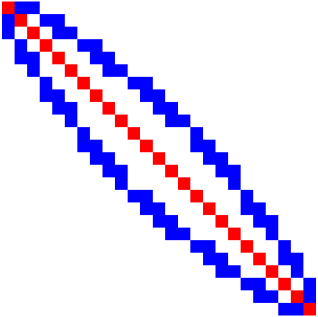




Interpretation on Structured Grids

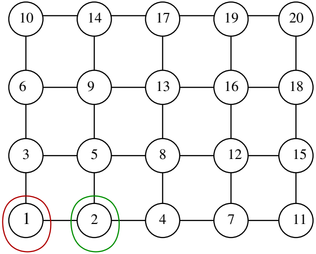
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

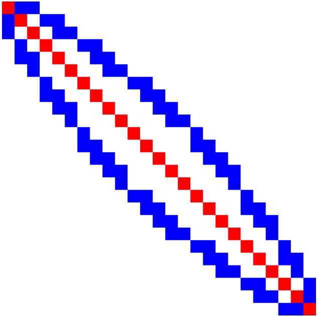




Interpretation on Structured Grids

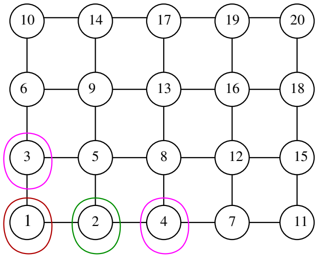
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

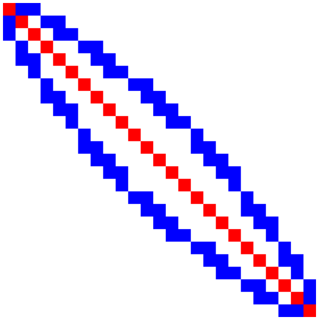




Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

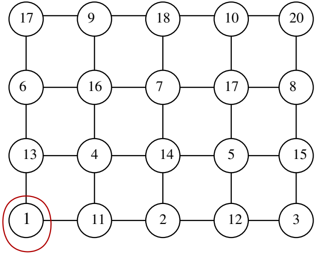


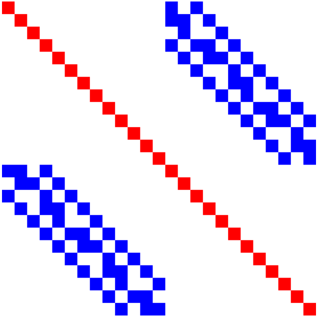


Incomplete LU Factorization Preconditioners

Interpretation on Structured Grids

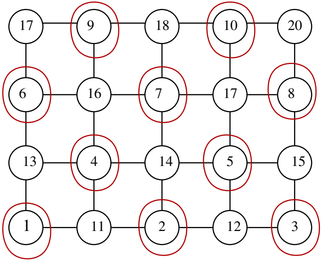
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

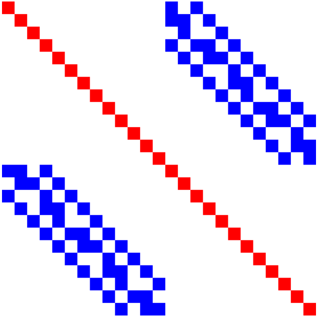




Interpretation on Structured Grids

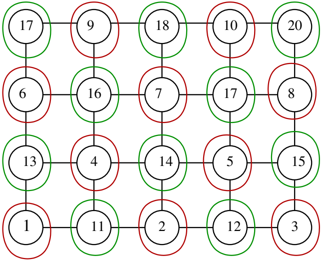
- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d

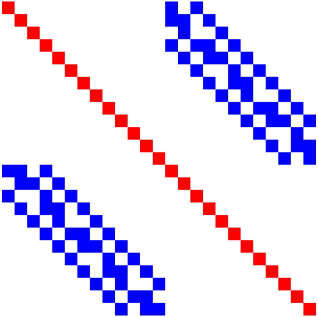




Interpretation on Structured Grids

- 2d finite-difference discretization
- Substitution whenever all neighbors with smaller index computed
- Works particularly well in 3d





Block-ILU

Apply ILU to diagonal blocks

Higher parallelism

Usually more iterations required (problem-dependent)

$$
\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}
$$

Block-ILU

- Apply ILU to diagonal blocks
- Higher parallelism
- Usually more iterations required (problem-dependent)

$$\begin{pmatrix}
5 & \times & \times & \times & & \times & \times & & \\
\times & 3 & \times & & & & & & \\
\times & \times & 4 & \times & & & & & \\
\times & & \times & 5 & \times & \times & & & \times \\
 & & & \times & 5 & \times & & \times & \times \\
\times & & & \times & \times & 6 & \times & \times & \\
\times & & & & & \times & 3 & & \\
 & & & & \times & \times & & 4 & \times \\
 & & & \times & \times & & & \times & 4
\end{pmatrix}$$

Benchmark - Setup

Benchmark Setup

Hardware

- NVIDIA GTX 580 (default)
- AMD HD 7970 (only for final benchmark)
- Intel Core2Quad 9550

Numbering

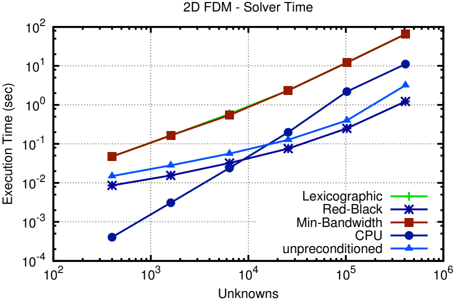
- Lexicographic
- Red-Black
- Minimum Degree

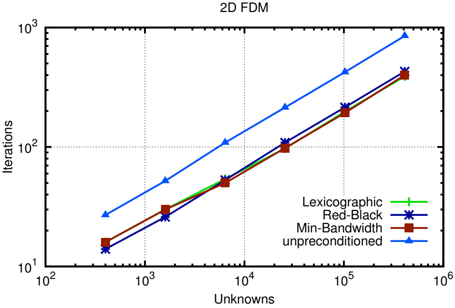
Remarks

- Setup purely on CPU, not included
- Data transfer costs not included
- OpenCL for both GPUs

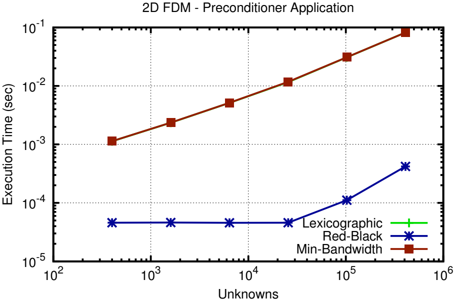
Case Study 1: 2D Poisson, Structured Grid

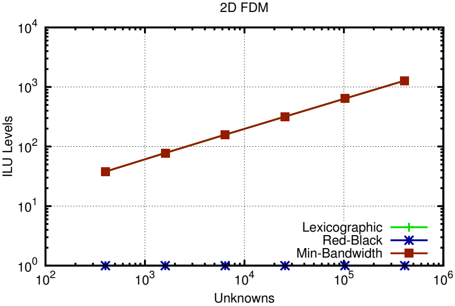
Benchmarks


2D FDM - Solver Time
Execution Time (sec)
Unknowns
Lexicographic
Red-Black
Min-Bandwidth
CPU
unpreconditioned


2D FDM
Iterations
Unknowns
Lexicographic
Red-Black
Min-Bandwidth
unpreconditioned

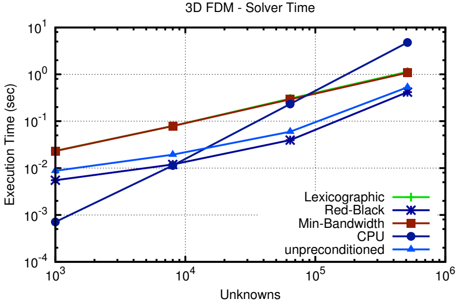
Benchmarks


2D FDM - Preconditioner Application
Execution Time (sec)
Unknowns
Lexicographic
Red-Black
Min-Bandwidth

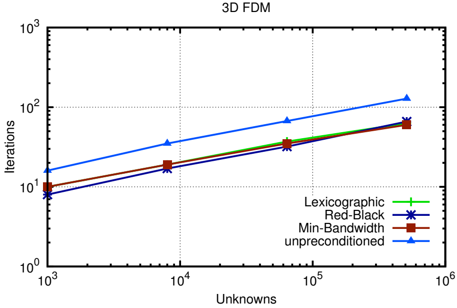

2D FDM
ILU Levels
Unknowns
Lexicographic
Red-Black
Min-Bandwidth

Case Study 2: 3D Poisson, Structured Grid

Benchmarks

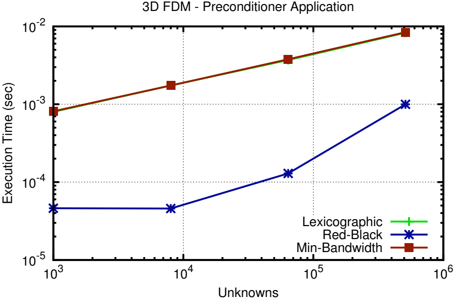


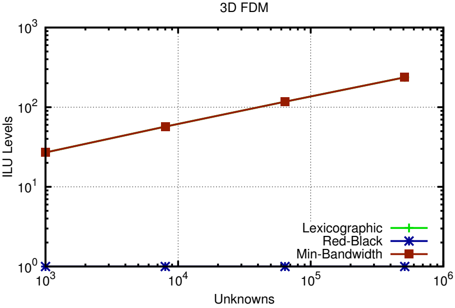
3D FDM - Solver Time
Execution Time (sec)
Unknowns
Lexicographic
Red-Black
Min-Bandwidth
CPU
unpreconditioned



3D FDM
Iterations
Unknowns
Lexicographic
Red-Black
Min-Bandwidth
unpreconditioned

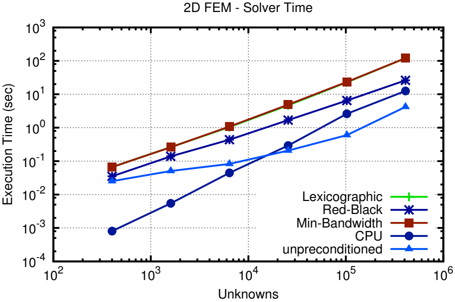
Benchmarks

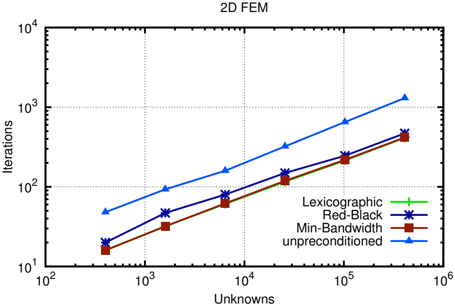

3D FDM - Preconditioner Application
Execution Time (sec)
Unknowns
Lexicographic
Red-Black
Min-Bandwidth


3D FDM
ILU Levels
Unknowns
Lexicographic
Red-Black
Min-Bandwidth

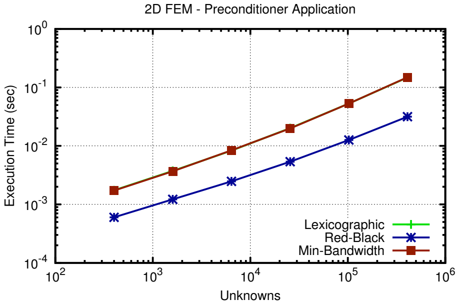
Case Study 3: 2D Poisson, Unstructured Grid

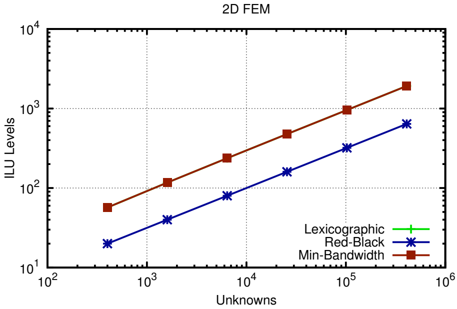
Benchmarks


2D FEM - Solver Time
Execution Time (sec)
Unknowns
Lexicographic
Red-Black
Min-Bandwidth
CPU
unpreconditioned


2D FEM
Iterations
Unknowns
Lexicographic
Red-Black
Min-Bandwidth
unpreconditioned

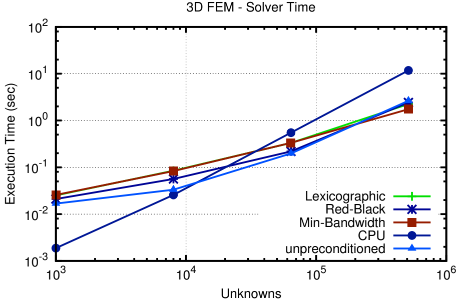
Benchmarks

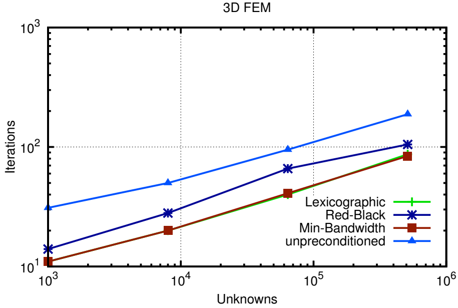

2D FEM - Preconditioner Application
Execution Time (sec)
Unknowns
Lexicographic
Red-Black
Min-Bandwidth


2D FEM
ILU Levels
Unknowns
Lexicographic
Red-Black
Min-Bandwidth

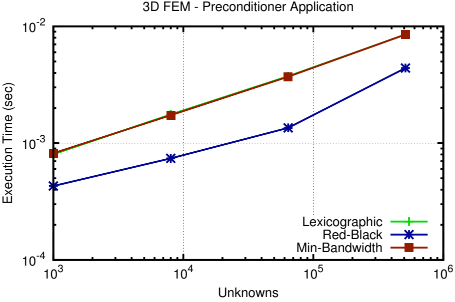
Case Study 4: 3D Poisson, Unstructured Grid

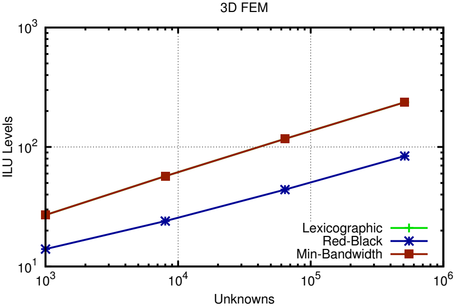
Benchmarks


3D FEM - Solver Time
Execution Time (sec)
Unknowns
Lexicographic
Red-Black
Min-Bandwidth
CPU
unpreconditioned


3D FEM
Iterations
Unknowns
Lexicographic
Red-Black
Min-Bandwidth
unpreconditioned

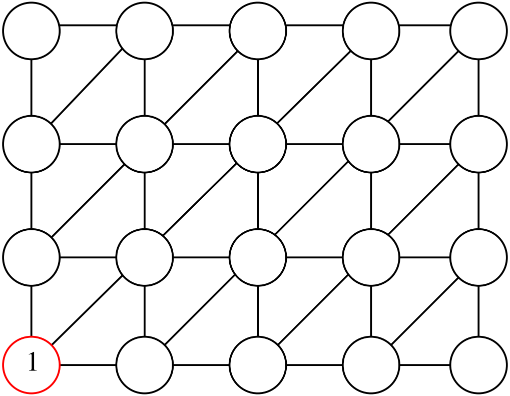
Benchmarks


3D FEM - Preconditioner Application
Execution Time (sec)
Unknowns
Lexicographic
Red-Black
Min-Bandwidth


3D FEM
ILU Levels
Unknowns
Lexicographic
Red-Black
Min-Bandwidth

Coloring

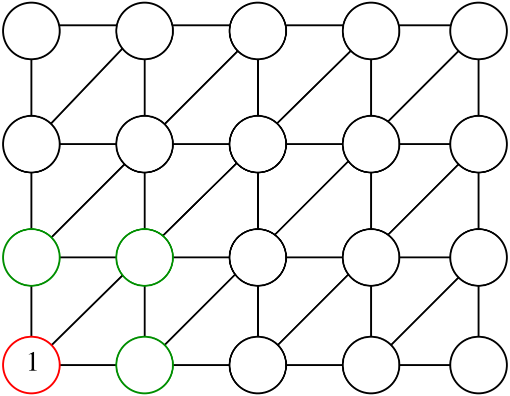
- Color dependency graph
- Purely algebraic



Coloring

Color dependency graph

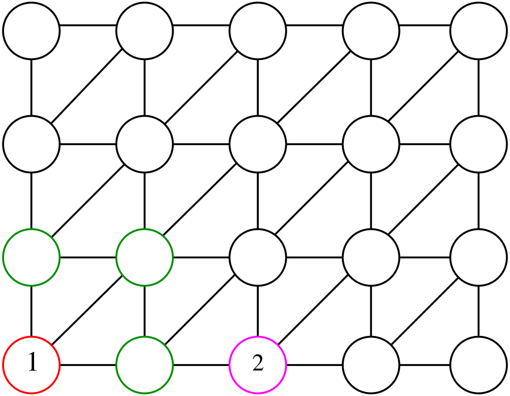
Purely algebraic



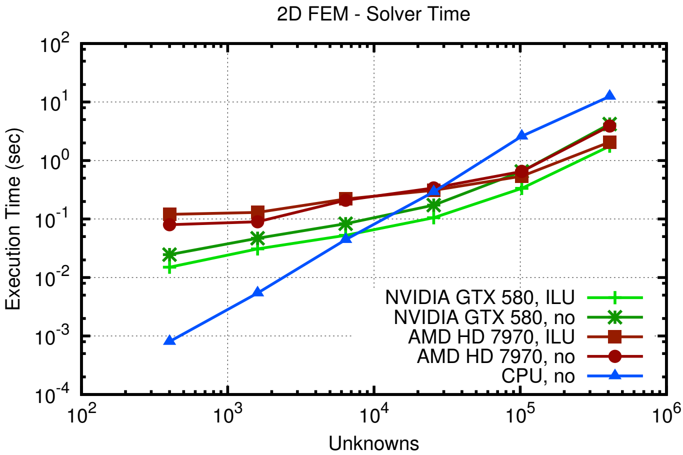
Coloring

Color dependency graph

Purely algebraic



Benchmarks



2D FEM - Solver Time
Execution Time (sec)
Unknowns
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
NVIDIA GTX 580, ILU
NVIDIA GTX 580, no
AMD HD 7970, ILU
AMD HD 7970, no
CPU, no

Conclusion

ILU Preconditioners

Fine-grained parallelism exploitable (if done right)

Higher-order discretizations less parallel

Matrix Pattern

CPU: banded for cache reuse

GPU: colored for parallelism

Availability

ViennaCL: `http://viennacl.sourceforge.net/`

(PETSc: `http://www.mcs.anl.gov/petsc/`)

ViennaCL + PETSc tutorial on Thursday afternoon!