A User-Friendly Highly Scalable AMG Solver

Yvan Notay*

Université Libre de Bruxelles
Service de Métrologie Nucléaire

Atlanta, February 3, 2017

* Supported by the Belgian FNRS
http://homepages.ulb.ac.be/∼ynotay

1. Motivation (1)

ULB

Many application software need an efficient solver for linear systems from (or related to) the discretization of PDEs like

$$-\text{div}(D\,\textbf{grad}(u)) + \mathbf{v}\,\textbf{grad}(u) + c\,u \;=\; f \qquad (+BC)$$

- De facto standard for a long time: direct solvers
  To substitute them one needs
  - a black box solver
    (we provide the matrix & rhs, it returns the solution)
  - that is robust
    (stable performance with respect to changes in the BC, PDE coeff., geometry & discretization grid)
- Efficient solver means:
  solve the system in near linear time: $\frac{\text{elapsed}}{n\times\#\text{cores}} \approx$ (small) cst

1. Motivation (2)

Multigrid methods:

often robust, always efficient $\rightarrow$ good candidates

AMG variants: designed to work black box

1. Motivation (2)

ULB

Multigrid methods:

often robust, always efficient $\rightarrow$ good candidates

AMG variants: designed to work black box – user friendly?

- In principle, yes: AMG stands for Algebraic Multigrid, where "algebraic" means that all algorithmic components are derived from the matrix itself

  $\rightarrow$ the user passes the matrix to the soft and that's it

- In practice, many people use an AMG software without knowing much about it

1. Motivation (2)

ULB

Multigrid methods:

often robust, always efficient $\rightarrow$ good candidates

AMG variants: designed to work black box – user friendly?

- In principle, yes: AMG stands for Algebraic Multigrid, where "algebraic" means that all algorithmic components are derived from the matrix itself

  $\rightarrow$ the user passes the matrix to the soft and that's it

- In practice, many people use an AMG software without knowing much about it

  . . .

  but many others experienced struggling with variant selection and parameters tuning before getting satisfactory results

2. Aggregation-based AMG (1)

Our approach for more user friendliness: make it simpler!

2. Aggregation-based AMG (1)

Our approach for more user friendliness: make it simpler!

Principle: Keep many features common to AMG methods

- Basic two-grid scheme which alternates smoothing iterations and coarse grid corrections
  - Gauss–Seidel for smoothing
  - Coarse grid correction based on a prolongation matrix P built from the system matrix A

    P is $n \times n_c$ (n_c : number of coarse variables)

    The coarse grid matrix is $A_c = P^T A P$

2. Aggregation-based AMG (1)

ULB

Our approach for more user friendliness: make it simpler!

Principle: Keep many features common to AMG methods

- Basic two-grid scheme which alternates smoothing iterations and coarse grid corrections
  - Gauss–Seidel for smoothing
  - Coarse grid correction based on a prolongation matrix P built from the system matrix A

    P is $n \times n_c$ (n_c: number of coarse variables)

    The coarse grid matrix is $A_c = P^T A P$
- Proceed recursively

  Set up phase: to define coarser and coarser levels

  Solve phase: approximate solution of coarse systems based on the two grid scheme at coarse level

2. Aggregation-based AMG (2)

ULB

. . . but

do not try to find a P which imitates geometric multigrid
(even for model problems)
nor focus on the accuracy of the coarse model

2. Aggregation-based AMG (2)

ULB

. . . but

do not try to find a P which imitates geometric multigrid (even for model problems)
nor focus on the accuracy of the coarse model

Instead, select P as simple & sparse as possible while keeping the two-grid convergence rate under control

2. Aggregation-based AMG (2)

ULB

. . . but

do not try to find a P which imitates geometric multigrid (even for model problems)
nor focus on the accuracy of the coarse model

Instead, select P as simple & sparse as possible while keeping the two-grid convergence rate under control

This is achieved with prolongation P based on plain aggregation

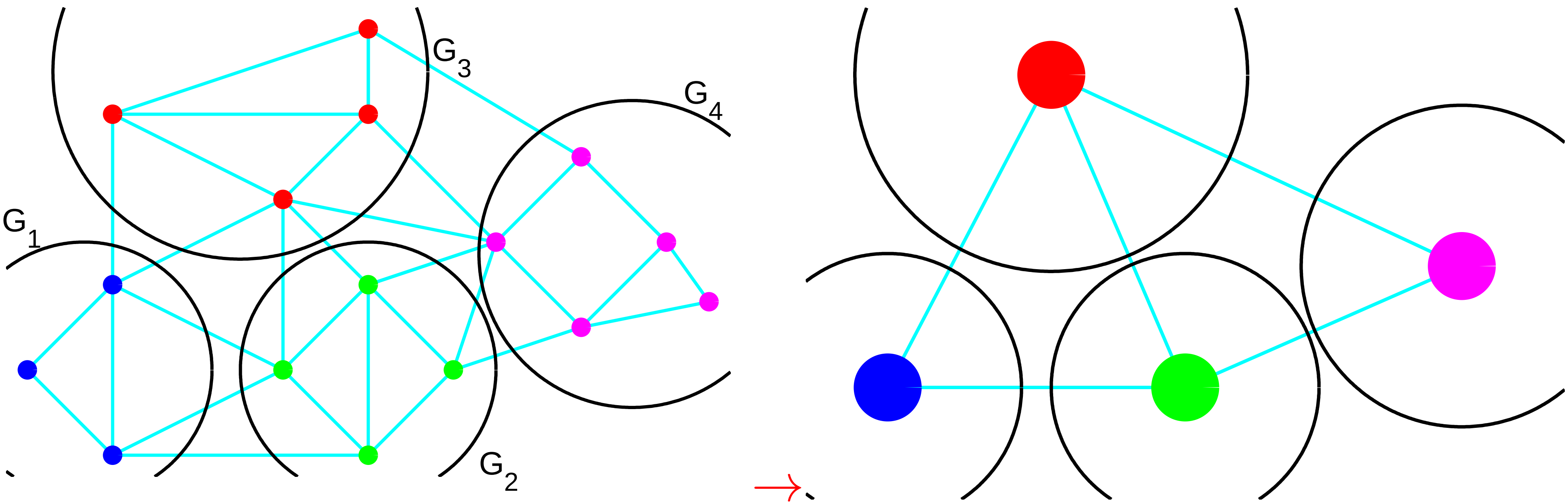
Coarsening by plain aggregation

Coarse unknowns: 1 per aggregate and vice-versa

Prolongation P : piecewise constant

Coarse grid matrix: obtained by a simple summation

$$P_{iJ} = \begin{cases} 1 & \text{if } i \in G_J \\ 0 & \text{otherwise} \end{cases} \quad , \qquad (A_c)_{ij} = \sum_{k \in G_i} \sum_{\ell \in G_j} a_{k\ell}$$



ULB

Convergence analysis

Under some assumptions, it can be shown that the two-grid spectral radius (or condition number) is bounded as a function of

$$K = \max_{i=1,\dots,n_c} K_{G_i}$$

where K_{G_i} is a quantity associated to aggregate G_i that is easy (and relatively cheap) to assess

K_{G_i} characterize thus the quality of the aggregate G_i

The theory is rigorous for M-matrices (possibly nonsymmetric), with heuristic extensions to matrices with nonnegative row-sum (in practice, works as long as the negative offdiagonal connections “dominate” the positive ones)

2. Aggregation-based AMG (5)

Quality aware aggregation algorithm

Basic principle: build aggregates in a greedy fashion, trying to optimize the quality indicator K_{G_i} while keeping it in any case within prescribed bounds

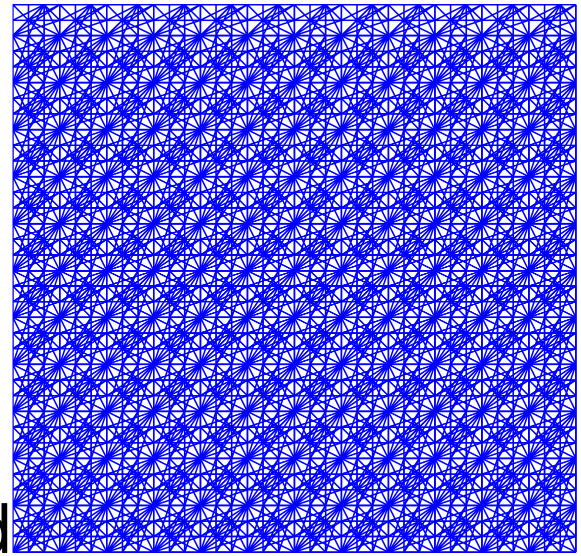
In that way, some minimal convergence properties are guaranteed, making the method particularly robust

(The two-grid convergence rate is under control)

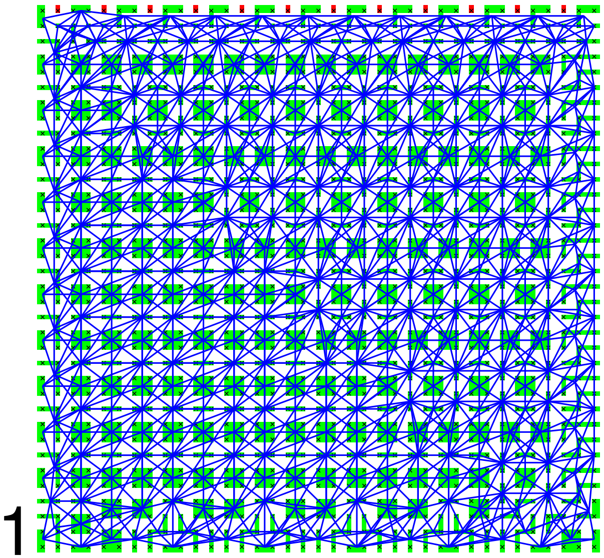
2. Aggregation-based AMG (6)

Illustration of how it may work

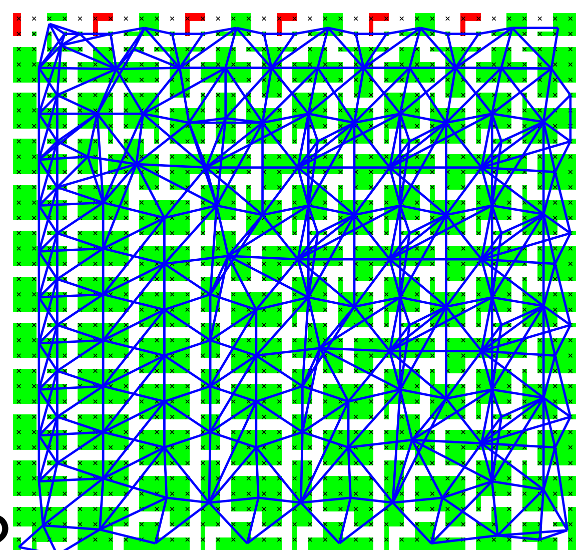
Regular grid, 3rd order finite elements (p3) for Poisson
($nnz(A) \approx 16\,n$)



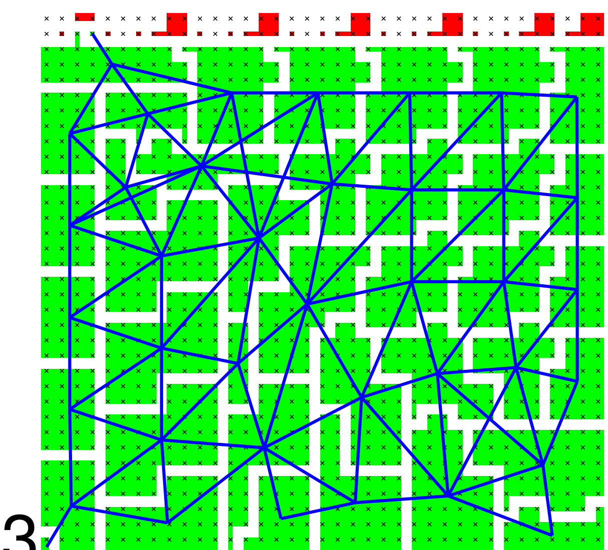
Fine grid



Level 1



Level 2



Level 3

2. Aggregation-based AMG (7)

No free lunch theorem:

We gained something, where are the downsides?

2. Aggregation-based AMG (7)

ULB

No free lunch theorem:

We gained something, where are the downsides?

(1) The two-grid convergence rate is not as good as can be with more classical AMG methods

2. Aggregation-based AMG (7)

ULB

No free lunch theorem:

We gained something, where are the downsides?

(1) The two-grid convergence rate is not as good as can be with more classical AMG methods

However, using the method as a preconditioner for CG or GMRES/GCR, the impact on the number of iterations is limited and can be offset by

- a cheaper setup
- a lower cost per iteration, thanks to
  - cheap smoothing
    (only 1 GS sweep for pre- and post-smoothing)
  - lighter coarse grid matrices
    (with less rows and less nonzero entries per row)

2. Aggregation-based AMG (8)

ULB

(2) Not optimal with standard multigrid V- or W-cycles (the number of iterations grows with the number of levels and hence with the matrix size)

2. Aggregation-based AMG (8)

(2) Not optimal with standard multigrid V- or W-cycles (the number of iterations grows with the number of levels and hence with the matrix size)

Solution: Enhanced multigrid cycle: the K-cycle

2. Aggregation-based AMG (8)

(2) Not optimal with standard multigrid V- or W-cycles
(the number of iterations grows with the number
of levels and hence with the matrix size)

Solution: Enhanced multigrid cycle: the K-cycle

In a multigrid algorithm, the coarse systems $A_c\mathbf{u}_c = \mathbf{r}_c$
are approximately solved with a few iterations of the
two-grid method at the considered (coarse) level
(recursivity: this way one moves to a further coarser level)

- 1 stationary iteration: V-cycle
- 2 stationary iterations: W-cycle

2. Aggregation-based AMG (8)

ULB

(2) Not optimal with standard multigrid V- or W-cycles
(the number of iterations grows with the number
of levels and hence with the matrix size)

Solution: Enhanced multigrid cycle: the K-cycle

In a multigrid algorithm, the coarse systems $A_c\mathbf{u}_c = \mathbf{r}_c$
are approximately solved with a few iterations of the
two-grid method at the considered (coarse) level
(recursivity: this way one moves to a further coarser level)

- 1 stationary iteration: V-cycle
- 2 stationary iterations: W-cycle
- 2 iter. with Krylov (CG/GMRES) acceleration: K-cycle

  Same workflow and about the same cost as with the W-cycle,
  but major robustness enhancement

Iterative solution with AGgregation-based algebraic MultiGrid

Linear system solver software package

- Black box
- FORTRAN 90 (easy interface with C & C++)
- Matlab interface

```
>>  x=agmg(A,y);
>>  x=agmg(A,y,1);   % SPD case
```

- Free academic license
  Professional version available (with extra features)

3. AGMG (robustness study) (2)

ULB

Robustness

Assessed on a large test suite of discrete second order elliptic PDEs, comprising

- problems on 2D/3D regular grids and on 2D/3D unstructured grids, some with strong local refinement
- problems with (big) jumps and/or (large) anisotropy in the PDE coefficients
- symmetric (SPD) and nonsymmetric problems (2D/3D convection-diffusion with dominating convection)
- finite difference and finite element (up to $p4$) discretizations

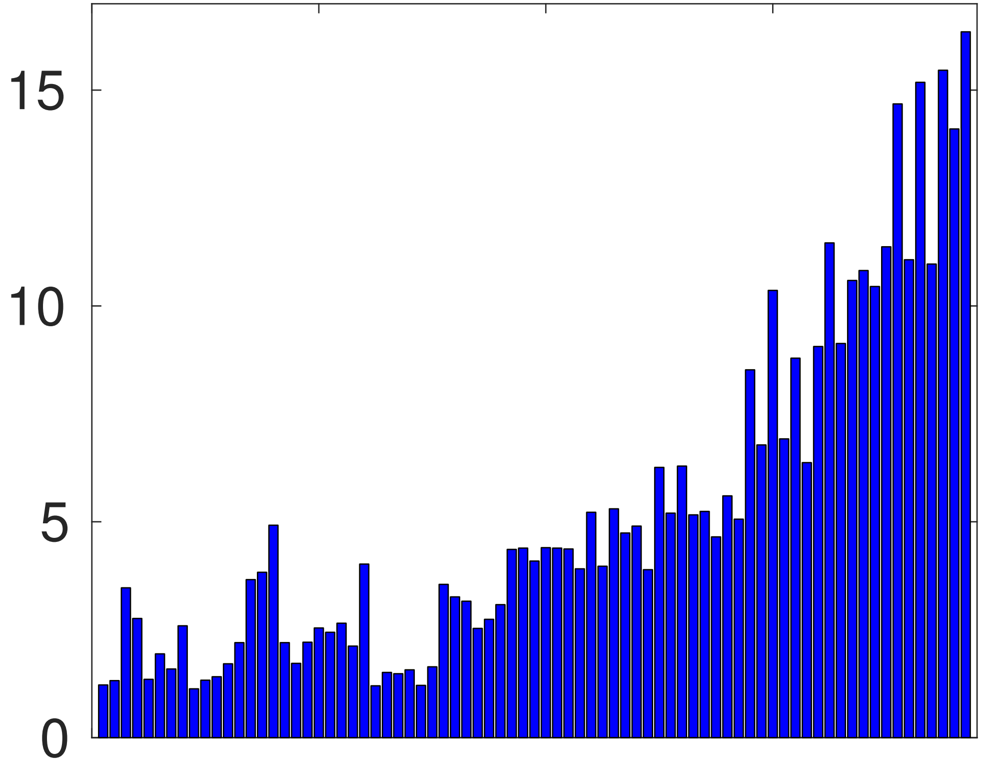
Size: Minimal: 5×10^5 – Maximal: $3. \times 10^7$

nnz per row: Minimal: $5.$ – Maximal: $74.$

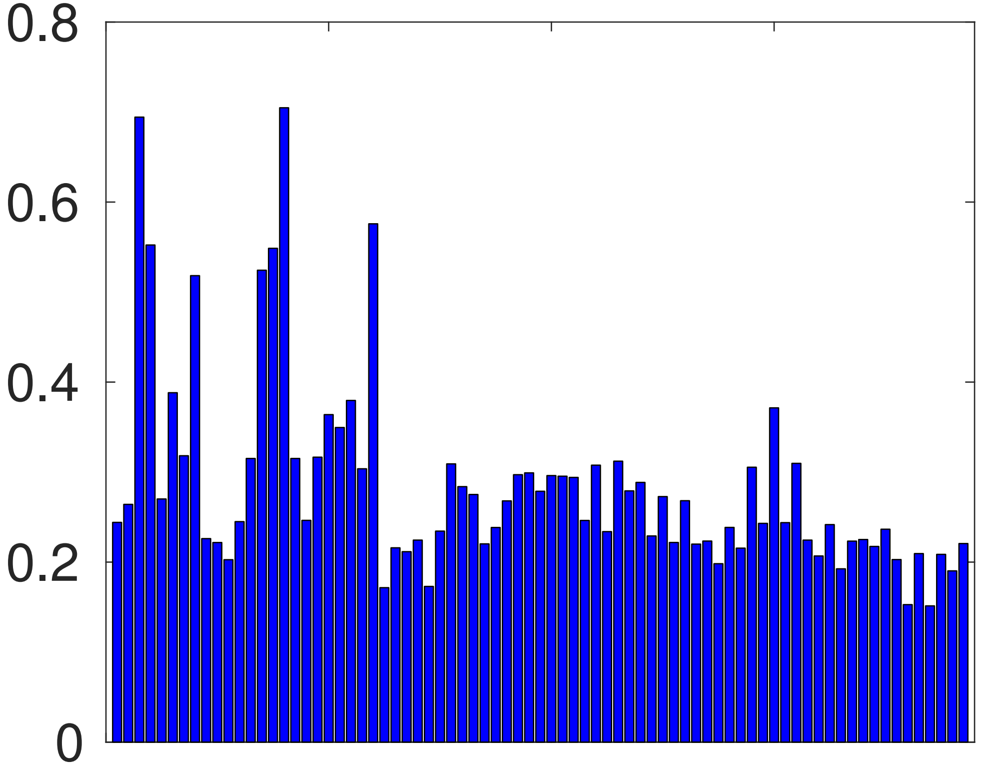
3. AGMG (robustness study) (3)

ULB

Time per unknown



Time per nnz



Total wall clock time in microseconds per unknown or nnz

– vs – problem index

(problems ordered by increasing number of nnz per row)

(Desktop workstation – Intel XEON E5-2620 at 2.10GHz – 2017)

3. AGMG (comparative study) (4)

ULB

Comparison with some other methods

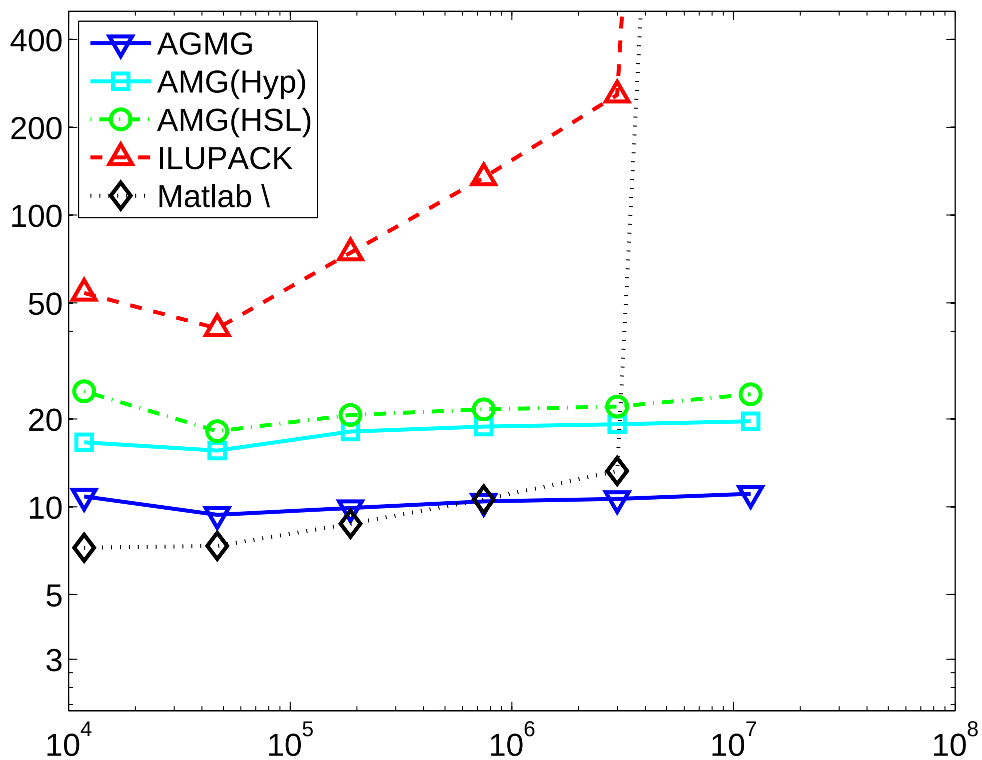
- AMG(Hyp): a classical AMG method (Hypre library)
- AMG(HSL): a classical AMG method (HSL library)
- ILUPACK: efficient threshold-based ILU preconditioner
- Matlab \: Matlab sparse direct solver (UMFPACK)

All methods but the last with Krylov subspace acceleration

Iterations stopped when $\frac{\|\mathbf{r}_k\|}{\|\mathbf{r}_0\|} < 10^{-6}$

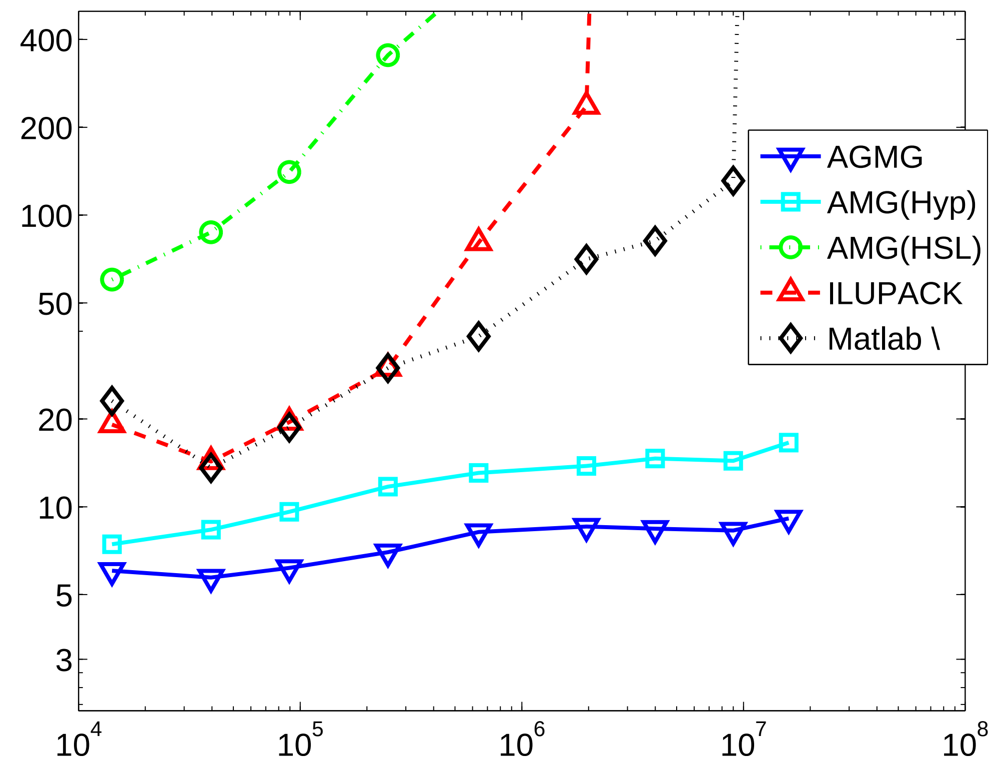
3. AGMG (comparative study) (5)

Poisson 2D, FE(P3)



(33% of nonzero offdiag > 0)

Convection-Diffusion 2D, FD



(circulating flow, $\nu = 10^{-6}$)

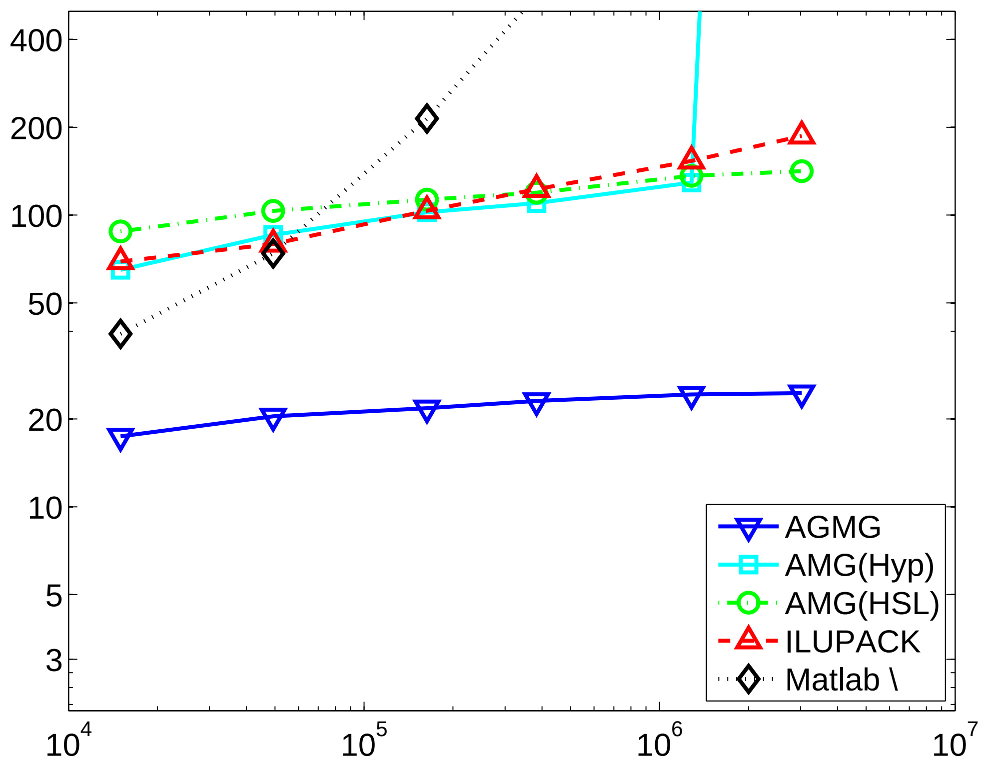
Wall clock time in microseconds per unknown – vs – # unknowns

(Computing node – Intel XEON L5420 processors at 2.50GHz – 2012)

3. AGMG (comparative study) (6)

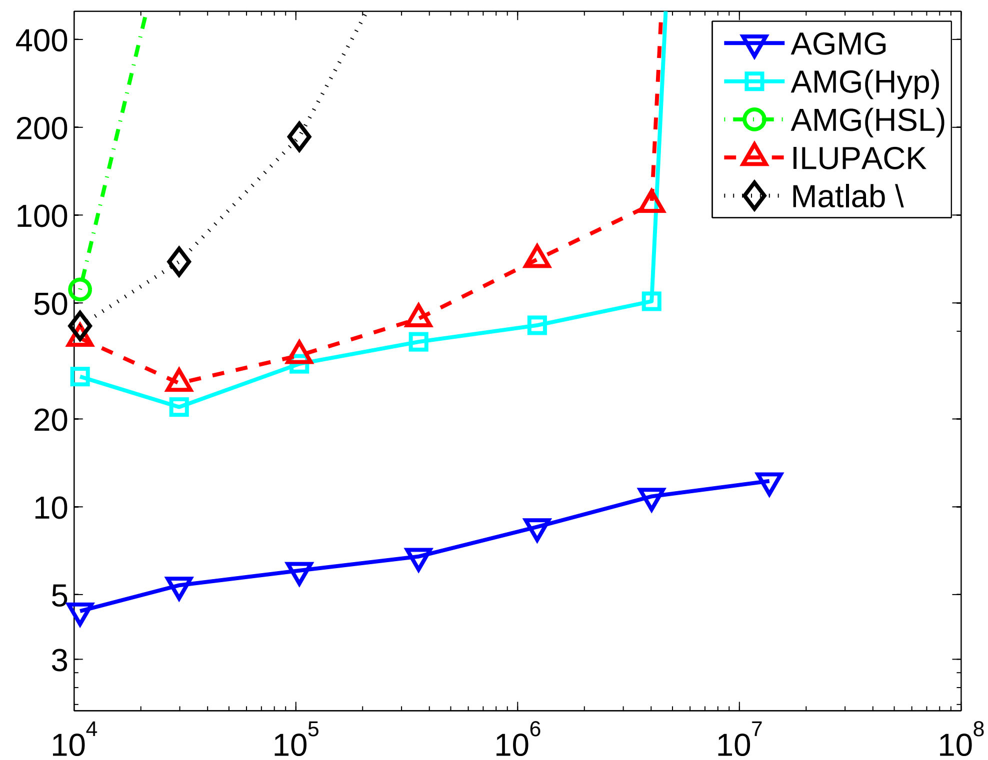
ULB

POISSON 3D, FE(P3)



(51% of nonzero offdiag > 0)

CONVECTION-DIFFUSION 3D, FD



(circulating flow, $\nu = 10^{-6}$)

Wall clock time in microseconds per unknown – vs – # unknowns

(Computing node – Intel XEON L5420 processors at 2.50GHz – 2012)

4. Parallelization (1)

ULB

General strategy

Partitioning of the unknowns $\rightarrow$ distribution of matrix rows

- Setup phase
  Aggregation algorithm: unchanged but aggregates are only formed with unknowns in a same partition

  $\rightarrow$ inherently parallel

  Only few communications needed to form the next coarse grid matrix

4. Parallelization (1)

ULB

General strategy

Partitioning of the unknowns $\rightarrow$ distribution of matrix rows

- **Setup phase**
  Aggregation algorithm: unchanged but aggregates are only formed with unknowns in a same partition

  $\rightarrow$ inherently parallel

  Only few communications needed to form the next coarse grid matrix

- **Solve phase**
  - Smoothing: Truncated Gauss-Seidel, ignoring connections between different partitions $\rightarrow$ inherently parallel
  - Grid transfer operations: inherently parallel

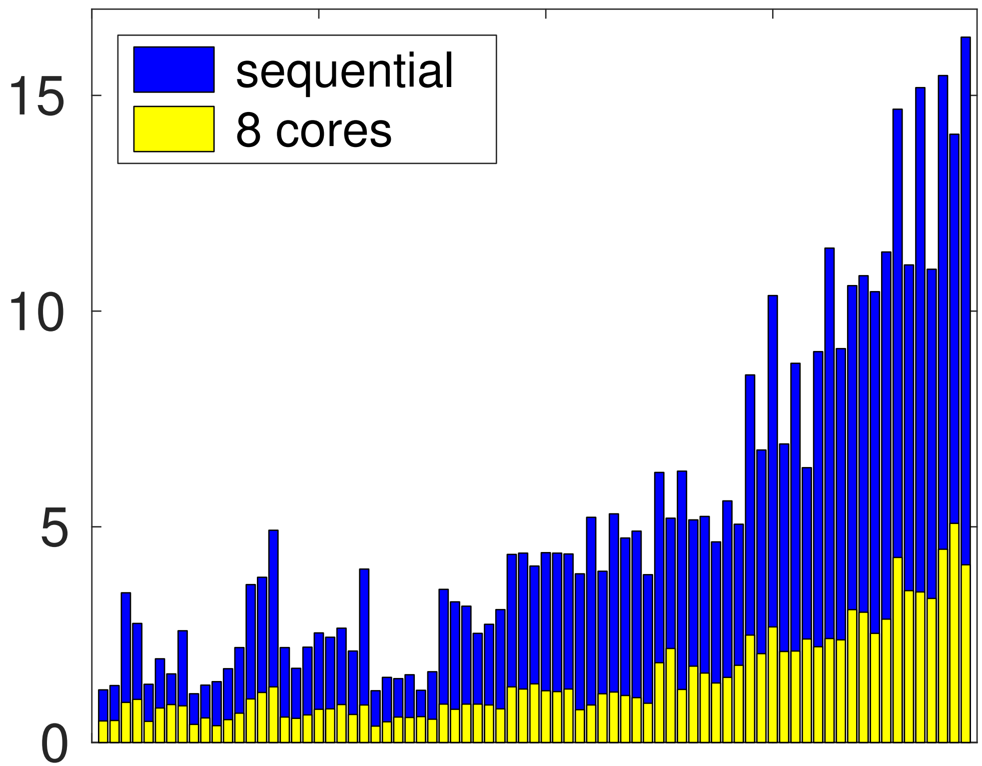
4. Parallelization (2)

ULB

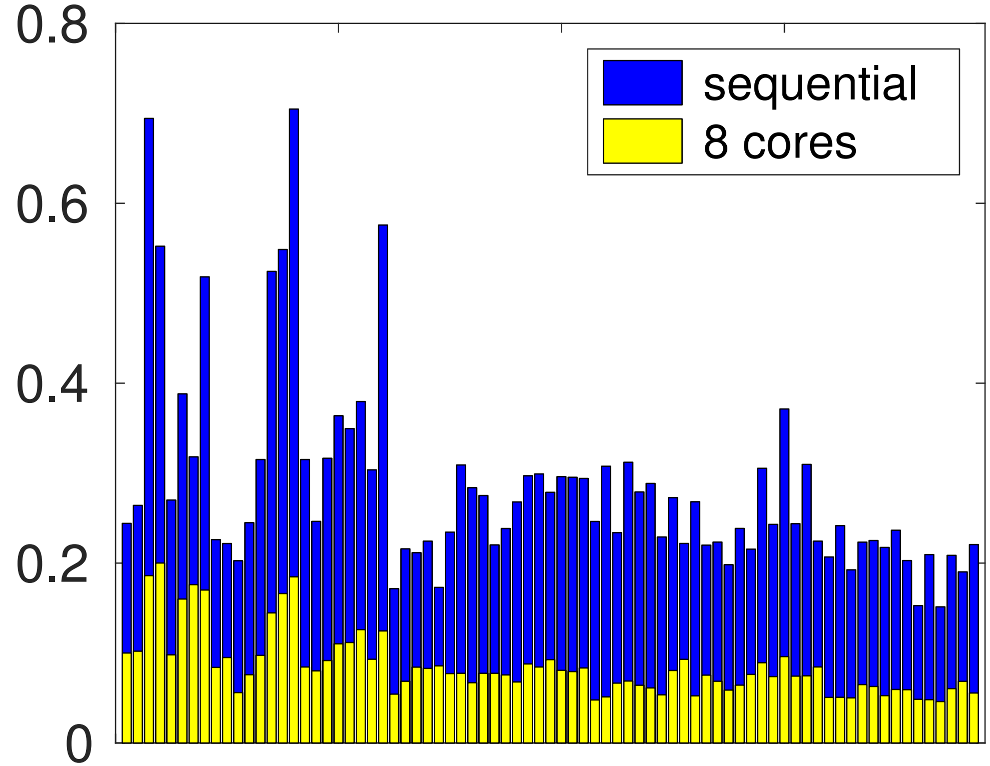
Nothing more needed for efficient multithreading (OpenMP)

Global results for the test suite:

Time per unknown



Time per nnz



Total wall clock time in microseconds per unknown or nnz

– vs – problem index

(Desktop workstation – Intel XEON E5-2620 at 2.10GHz – 2017)

4. Parallelization (3)

ULB

MPI: the Bottom level solver can be a bottleneck

- More frequently called than with the V-cycle
  $\rightarrow$ more critical

4. Parallelization (3)

ULB

MPI: the Bottom level solver can be a bottleneck

- More frequently called than with the V-cycle
  $\rightarrow$ more critical
- Sequential AGMG uses a sparse direct solver, but parallel versions of these do not scale well enough

4. Parallelization (3)

ULB

MPI: the Bottom level solver can be a bottleneck

- More frequently called than with the V-cycle
  $\rightarrow$ more critical
- Sequential AGMG uses a sparse direct solver, but parallel versions of these do not scale well enough
- Thus: dedicated Iterative bottom level solver
  Rationale:
  - only a small % of total flops
    $\rightarrow$ some suboptimality is harmless
  - few unknowns involved
    $\rightarrow$ needs a method that scales well despite this

  Our choice:
  a simplified two-level domain decomposition method

4. Parallelization (4)

ULB

Iterative bottom level solver

- Aggregation-based two-grid method
  (one further level: very coarse grid)
- All unknowns on a same process form 1 aggregate
  (very coarse grid: size = number of processes (cores))
- Better smoother:
  Block Jacobi
  (sparse direct solver for the local part of the matrix)
- Solution of very coarse grid systems:
  Sparse direct solver

4. Parallelization (4)

ULB

Iterative bottom level solver for massive parallelism

- Aggregation-based two-grid method
  (one further level: very coarse grid)
- All unknowns on a same process form 1 aggregate
  (very coarse grid: size = number of processes (cores))
- Better smoother:
  ~~Block Jacobi~~
  ~~(sparse direct solver for the local part of the matrix)~~
  Apply sequential AGMG to the local part of the matrix
  Allows us to use only 4 levels whatever the matrix size
- Solution of very coarse grid systems:
  ~~Sparse direct solver~~
  AGMG again, sequential or parallel within subgroups of processes (depending on the size of the systems)

4. Parallelization (5)

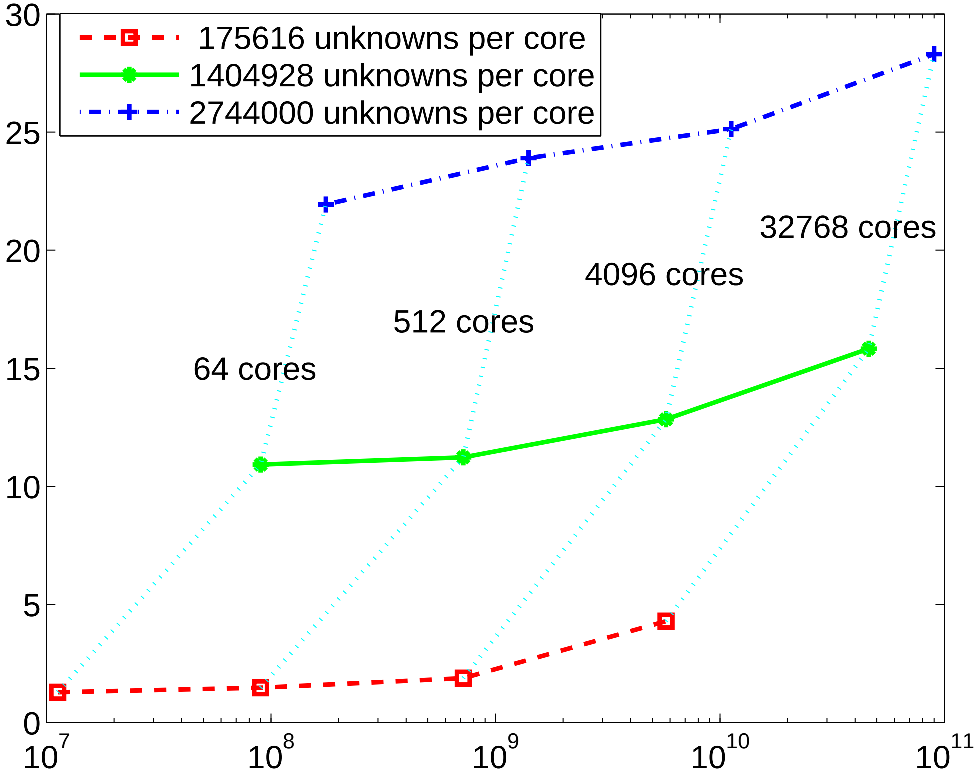
ULB

Performance of AGMG on supercomputers

3D Poisson (FD) on HERMIT (HPC – Cray XE6 – 2014)

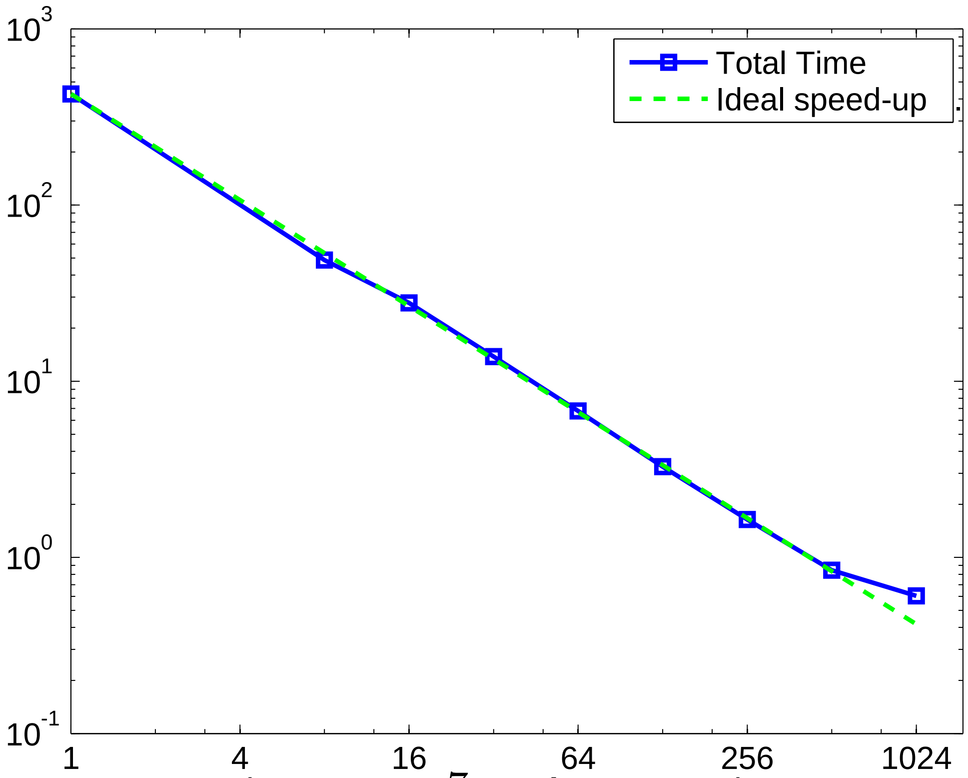
Weak scalability

Time – vs – # unknowns



Strong scalability

Time – vs – number of cores



Times reported are total wall clock times in seconds

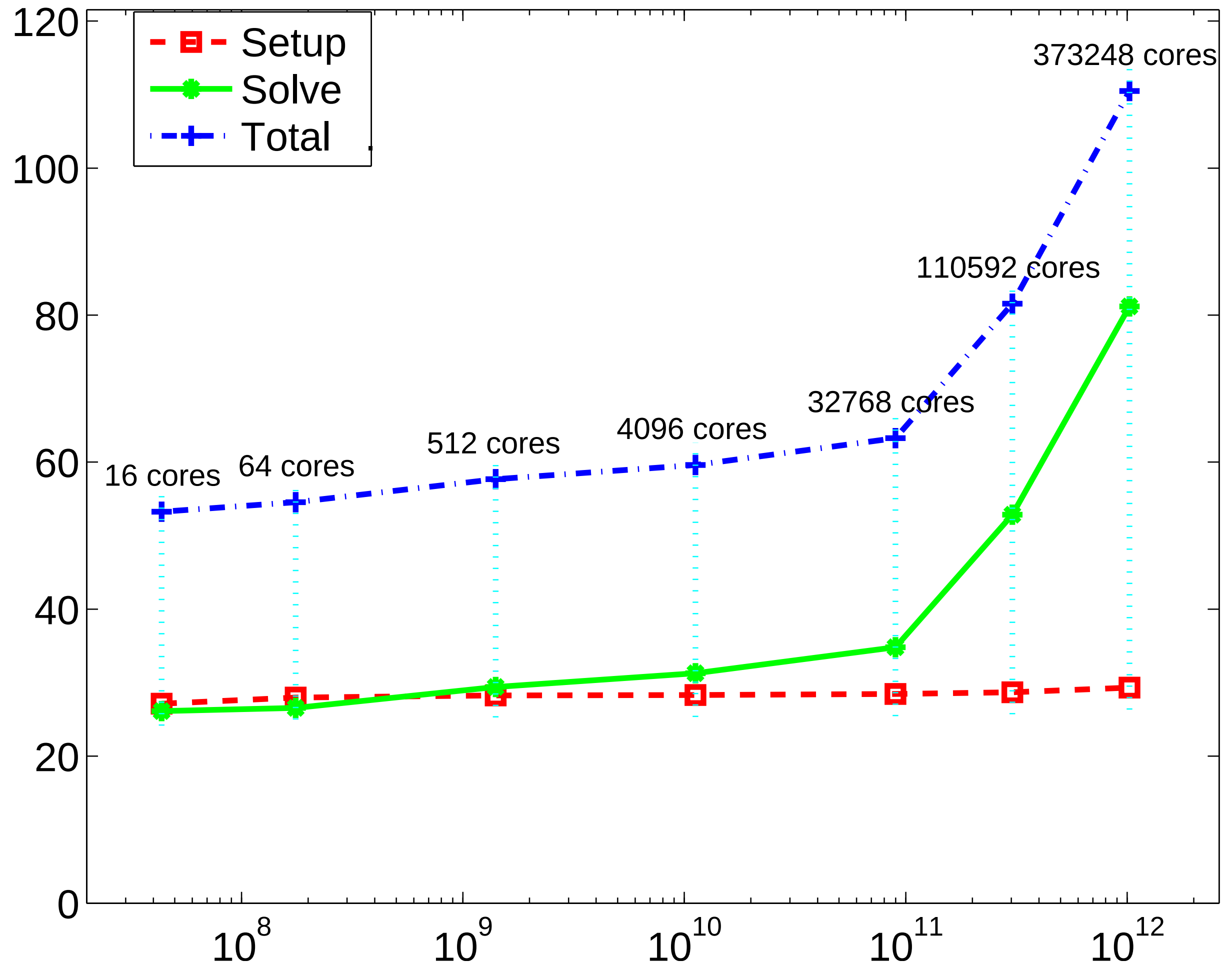
4. Parallelization (6)

ULB

3D Poisson (FD) on JUQUEEN (HPC – IBM BG/Q – 2014)

Weak scalability:

Time – vs – # unknowns



Times reported are total wall clock times in seconds

5. Extensions

ULB

Stokes problems

- Simple algebraic transformation to reinforce the weight of the diagonal blocks (literally: pre-conditioning)
- Then, use the block version of AGMG, that constraints the aggregate to be formed with a single type of unknown at a time (velocity component, pressure)

  $\rightarrow$ monolithic AMG, faster than block preconditioning

5. Extensions

ULB

Stokes problems

- Simple algebraic transformation to reinforce the weight of the diagonal blocks (literally: pre-conditioning)
- Then, use the block version of AGMG, that constraints the aggregate to be formed with a single type of unknown at a time (velocity component, pressure)

  $\rightarrow$ monolithic AMG, faster than block preconditioning

Graph Laplacians

- Standard aggregation based on multiple pairwise matching is inefficient for some exotic sparsity patterns
- Robustness recovered with Degree aware Rooted Aggregation (DRA)

  . . . can be also combined with quality control
- The resulting method is significantly faster than LAMG

6. Conclusions

- AGMG: AMG method with “simplified” coarsening
  Benefits:

6. Conclusions

- AGMG: AMG method with “simplified” coarsening
  Benefits:
  - Explicit control of the two-grid convergence rate
    $\longrightarrow$ Robustness

6. Conclusions

ULB

- AGMG: AMG method with “simplified” coarsening
  Benefits:
  - Explicit control of the two-grid convergence rate
    $\longrightarrow$ Robustness
  - Connectivity pattern on the coarse grids:
    similar or sparser than that on the fine grid
    ($\longrightarrow$ no complexity issue with the recursive use)

6. Conclusions

ULB

- AGMG: AMG method with “simplified” coarsening
  Benefits:
  - Explicit control of the two-grid convergence rate
    $\rightarrow$ Robustness
  - Connectivity pattern on the coarse grids:
    similar or sparser than that on the fine grid
    ($\rightarrow$ no complexity issue with the recursive use)
  - Easy parallelization
    (At large scale, need clever bottom level solver)

6. Conclusions

- AGMG: AMG method with “simplified” coarsening
  Benefits:
  - Explicit control of the two-grid convergence rate
    $\rightarrow$ Robustness
  - Connectivity pattern on the coarse grids:
    similar or sparser than that on the fine grid
    ($\rightarrow$ no complexity issue with the recursive use)
  - Easy parallelization
    (At large scale, need clever bottom level solver)
  - Only one variant, no parameter tuning needed
    $\rightarrow$ userfriendliness

6. Conclusions

ULB

- AGMG: AMG method with “simplified” coarsening
  Benefits:
  - Explicit control of the two-grid convergence rate
    $\rightarrow$ Robustness
  - Connectivity pattern on the coarse grids:
    similar or sparser than that on the fine grid
    ($\rightarrow$ no complexity issue with the recursive use)
  - Easy parallelization
    (At large scale, need clever bottom level solver)
  - Only one variant, no parameter tuning needed
    $\rightarrow$ userfriendliness
- Can be faster than other state-of-the-art solvers

6. Conclusions

ULB

- AGMG: AMG method with “simplified” coarsening
  Benefits:
  - Explicit control of the two-grid convergence rate
    $\rightarrow$ Robustness
  - Connectivity pattern on the coarse grids:
    similar or sparser than that on the fine grid
    ($\rightarrow$ no complexity issue with the recursive use)
  - Easy parallelization
    (At large scale, need clever bottom level solver)
  - Only one variant, no parameter tuning needed
    $\rightarrow$ userfriendliness
- Can be faster than other state-of-the-art solvers
- Fairly small setup time: especially well suited when only a modest accuracy is needed

Selected references

http://homepages.ulb.ac.be/~ynotay/AGMG

The K-cycle

- Recursive Krylov-based multigrid cycles (with P. S. Vassilevski), NLAA, 2008

Two-grid analysis of aggregation-based methods

- Algebraic analysis of aggregation-based multigrid (with A. Napov), NLAA, 2011

AGMG and quality aware aggregation

- An aggregation-based algebraic multigrid method, ETNA, 2010.
- An algebraic multigrid method with guaranteed convergence rate (with A. Napov), SISC, 2012
- Aggregation-based algebraic multigrid for convection-diffusion equations, SISC, 2012
- Algebraic multigrid for moderate order finite elements (with A. Napov), SISC, 2014

Parallelization

- A massively parallel solver for discrete Poisson-like problems, J. Comput. Physics, 2015

Graph Laplacians

- An efficient multigrid method for graph Laplacian systems II: robust aggregation (with A. Napov), SISC, to appear

AMG for Stokes

- Algebraic multigrid for Stokes equations, SISC, to appear

Selected references !! Thank you !!

ULB

http://homepages.ulb.ac.be/~ynotay/AGMG

The K-cycle

- Recursive Krylov-based multigrid cycles (with P. S. Vassilevski), NLAA, 2008

Two-grid analysis of aggregation-based methods

- Algebraic analysis of aggregation-based multigrid (with A. Napov), NLAA, 2011

AGMG and quality aware aggregation

- An aggregation-based algebraic multigrid method, ETNA, 2010.
- An algebraic multigrid method with guaranteed convergence rate (with A. Napov), SISC, 2012
- Aggregation-based algebraic multigrid for convection-diffusion equations, SISC, 2012
- Algebraic multigrid for moderate order finite elements (with A. Napov), SISC, 2014

Parallelization

- A massively parallel solver for discrete Poisson-like problems, J. Comput. Physics, 2015

Graph Laplacians

- An efficient multigrid method for graph Laplacian systems II: robust aggregation (with A. Napov), SISC, to appear

AMG for Stokes

- Algebraic multigrid for Stokes equations, SISC, to appear