Towards a Parallel Algebraic Multigrid Solver Using PGAS

Niclas Jansson and Erwin Laure

KTH Royal Institute of Technology

Acknowledgment

The main goal of the EC Horizon 2020 funded ExaFLOW project is to address key algorithmic challenges in CFD to enable simulation at exascale,

- Accurate **error control**, adaptive mesh refinement
- Solver **efficiency**, scalable numerical methods and preconditioners
- Strategies to ensure **fault tolerance** and resilience
- **Input/output** for extreme data, data reduction
- **Energy awareness** in solver design

http://www.exaflow-project.eu

Introduction

Why use PGAS languages?

- One-sided communication
- Allow for fine grained parallelism
- More productive languages

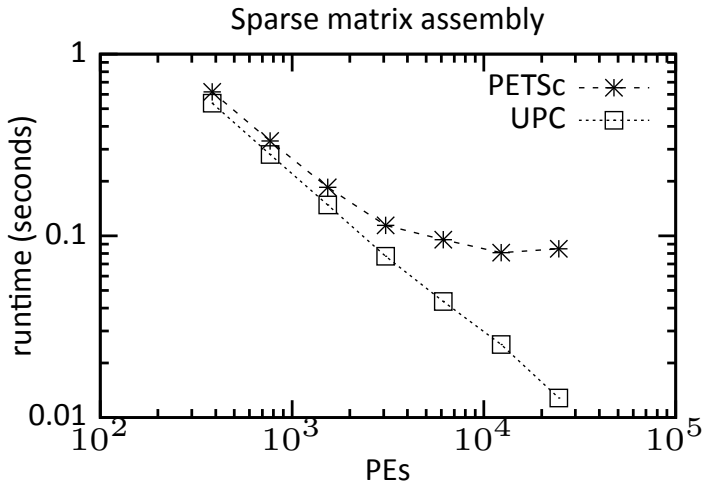
A typical two-sided distributed memory program

Data decomposition
while not done **do**
  Compute local part
  Send/Receive overlap
  Add contribution from overlap
end while

One sided comm. allows for novel parallelization algorithms

PGAS based Linear Algebra library

- Row wise distribution of matrices and column vectors
- Matrix/vector entries accessible by all
  - Easier to write solvers/preconditioners
  - Less synchronization points
- Implemented in Unified Parallel C
- Hybrid interface for use with MPI codes
- Low latency communication kernels
  - Reduce overhead cf. message passing
  - Improving fine grained parallelism
    - Sparse matrix assembly (FEM)
    - Allowing for less elements/core

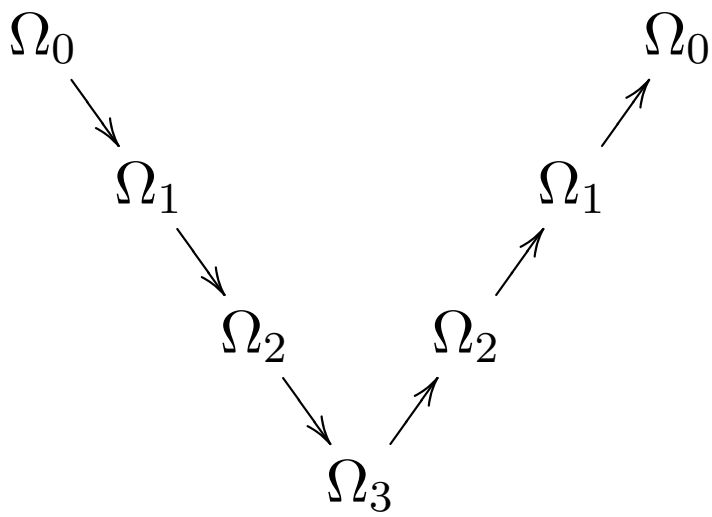


Algebraic Multigrid

A multilevel method for solving $Ax = b$, where A is an $n \times n$ matrix with entries a_{ij} and x, b are vectors of size n.

- Eliminated smooth errors by solving $Ae = r$, on a coarser problem
- Interpolate back, and correct the fine solution, $x = x + e$

if $k =$ coarsest level **then**
  Solve $A^k x^k = f^k$
else
  Relax μ_1 times on $A^k x^k = f^k$
  Set $x^{k+1} = 0$, $f^{k+1} = I_k^{k+1}(f^k - A^k x^k)$
  Apply V--cycle on level $k + 1$
  Correct the solution by
  $x^k = x^k + I_{k+1}^k x^{k+1}$
  Relax μ_2 times on $A^k x^k = f^k$
end if



- Asymptotically optimal complexity
- Notorious difficult to parallelize in an optimal way!

AMG coarsening

Let Ω^k be the set of components of x at level k

- Split Ω^k into two disjoint sets C and F
- No underlying geometry
- Need to pick coefficients in A^{k+1} related to a error freq.
- Classify if unknowns are strongly coupled to each other

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\}$$

- Measure λ_i, number of points strongly influenced by i

Ruge-Stüben Coarsening

Related to interpolation quality

> ***C1:*** *For each point j that strongly influences a F-point i, j is either a C-point or it strongly depends on a C-point l that also strongly influences i.*

Size of the coarser level

> ***C2:*** *C should be a maximal subset of all points with the property that no two C points are strongly connected to each other.*

Ruge-Stüben Coarsening

Let $U = \Omega^k$
/* First phase */
while $U \neq \emptyset$ **do**
    Pick an $i \in U$ with maximal λ_i
    Set $U = U - \{i\}$, $C = C + \{i\}$
    Add all points j which strongly depends on i to F
    Increase the measure λ_l for all points l that are strongly dependent on j
    Decrease the measure λ_m for all points m that are strongly dependent on i
end while

/* Second phase */
for all $i \in F$ **do**
    if i violates criteria **C1 then**
        $F = F - \{i\}$, $C = C + \{i\}$
    end if
end for

AMG Coarsening

A challenge for message passing is the restricted local view of the data

- A thread must be able to determine if a neighboring point is *C* or *F*
- Points will change between *F* and *C* during coarsening
- Explicit communication with neighbors
- Less optimal coarsening algorithms
  - Easier parallelization
  - Pay with more multigrid cycles
  - Set/Graph based (CLJP, PMIS HMIS)
  - Heuristics (RS3)

Restrictions comes from a two-sided message passing perspective

- Put *C*/*F* data in global memory
- Accessible by all threads
- Solves most of the parallelization issues

AMG Coarsening

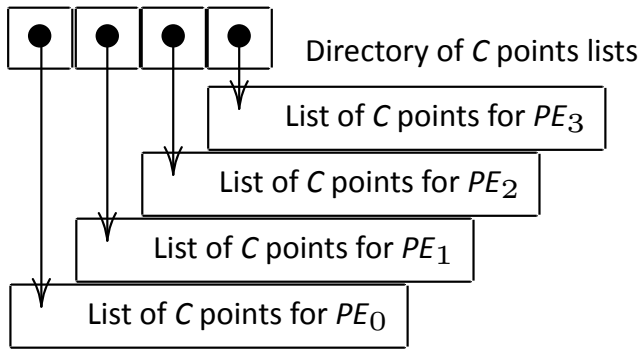
PGAS based RS

Most of the data structures are local

- Unordered set of *C* and *F* points
- Measure λ_i stored in red-black trees

Keep a list of *C* variables in global memory

- PE dependent block size
- Directory approach (arbitrary size)
- Protect the list with a set of variables
  - Declared as UPC `strict`
  - Less expensive than using locks



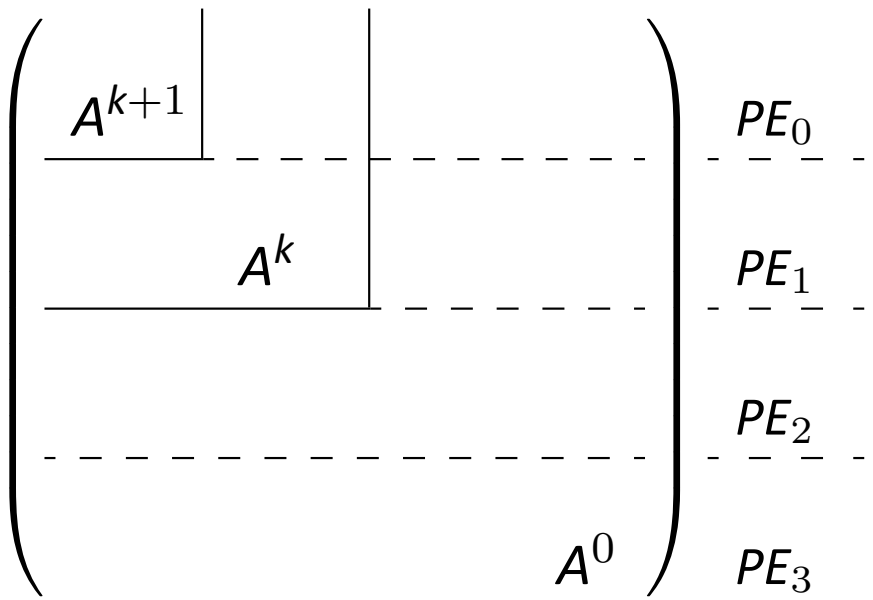
PGAS based RS

Let $U = \Omega^k$ and $Cg(:) = 0$
/* First phase */
while $U \neq \emptyset$ **do**
    Pick an $i \in U$ with maximal λ_i
    Set $U = U - \{i\}$, $C = C + \{i\}$ and $Cg(i) = 1$
    Add all points j which strongly depends on i to F
    Increase the measure λ_l for all points l that are strongly dependent on j
    Decrease the measure λ_m for all points m that are strongly dependent on i
end while
Barrier
/* Second phase */
for all $i \in F$ **do**
    if i violates criteria **C1 then**
        Wait **while** $S(i) \neq 0$
        $S(i) = 1$ /* Protect variable i */
        $F = F - \{i\}$, $C = C + \{i\}$ and $Cg(i) = 1$
        $S(i) = 0$ /* Release variable i */
    end if
end for

Load Balancing

- A^{k+1} becomes smaller and smaller for each coarsening level k
- Move operator towards a single core (easier coarse level solve)
- Use a load balanced linear distribution $N = PL + R$,

$$L = \left\lfloor \frac{N}{P} \right\rfloor,$$
$$R = N \bmod P,$$
$$n = \left\lfloor \frac{N + P - p - 1}{P} \right\rfloor$$



AMG Solver

- Multigrid Cycling
  - Matrix vector products
  - Redistribution routines (load balancing)
- Coarse Level Solver
  - Direct solver (single PE)
  - Krylov solver (multiple PEs)
- Smoother
  - Hybrid *CF* Gauss Seidel

$$x_i^k = \left(b_i - \sum_{j<i} a_{ij} x_j^k - \sum_{j>i} a_{ij} x_j^{k-1} \right) \Big/ a_{ii}$$

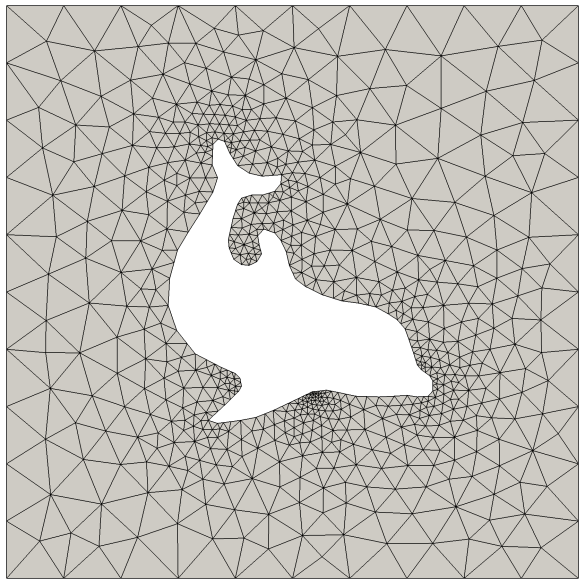
  - Straightforward implementation (PGAS)
  - Work across PE boundaries

Benchmark Problem

Poisson's equation on the unit square

$$
\begin{aligned}
-\Delta u(x,y) &= f(x,y), && (x,y) \in \Omega, \\
u(x,y) &= 0, && (x,y) \in \Gamma_0, \\
\partial_n u(x,y) &= g(x,y), && (x,y) \in \Gamma_1, \\
\partial_n u(x,y) &= 0, && (x,y) \in \partial\Omega \setminus (\Gamma_0 \cup \Gamma_1),
\end{aligned}
$$

$$
\begin{aligned}
f(x,y) &= 500 \exp(-((x-0.5)^2 + (y-0.5)^2)/0.02) \\
g(x,y) &= 25 \sin(5\pi y).
\end{aligned}
$$



Evaluation

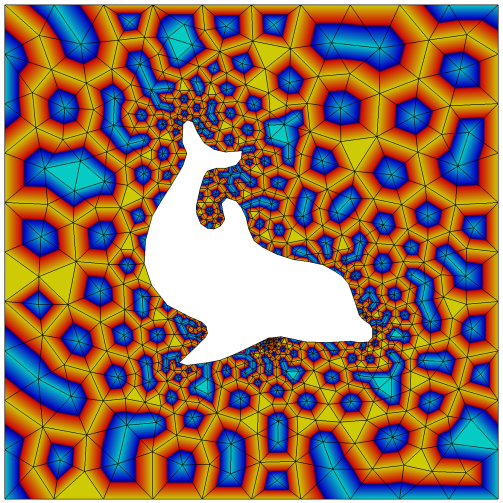
Benchmark Problem

- Discretize PDE by FEM
- Use the FEM framework FEniCS
  - FEniCS assembles the stiffness matrix
  - Linear system solved by external libraries
- Hybrid MPI + PGAS
- Use PETSc as a reference krylov solver
- All experiments performed on the Cray XC40 Beskow at PDC/KTH

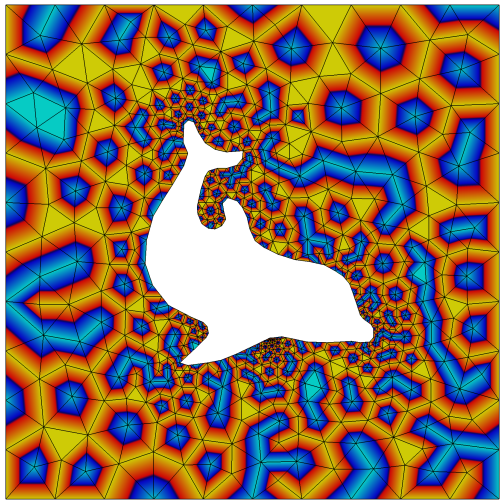
Coarsening Quality

Parallel Coarsening

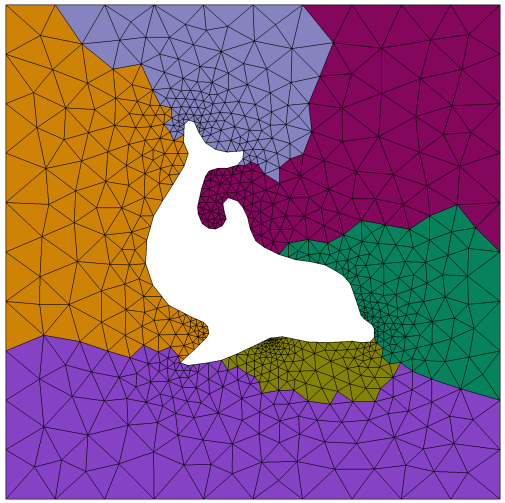
- No artifacts from boundary between partitions



Serial coarsening



Parallel coarsening



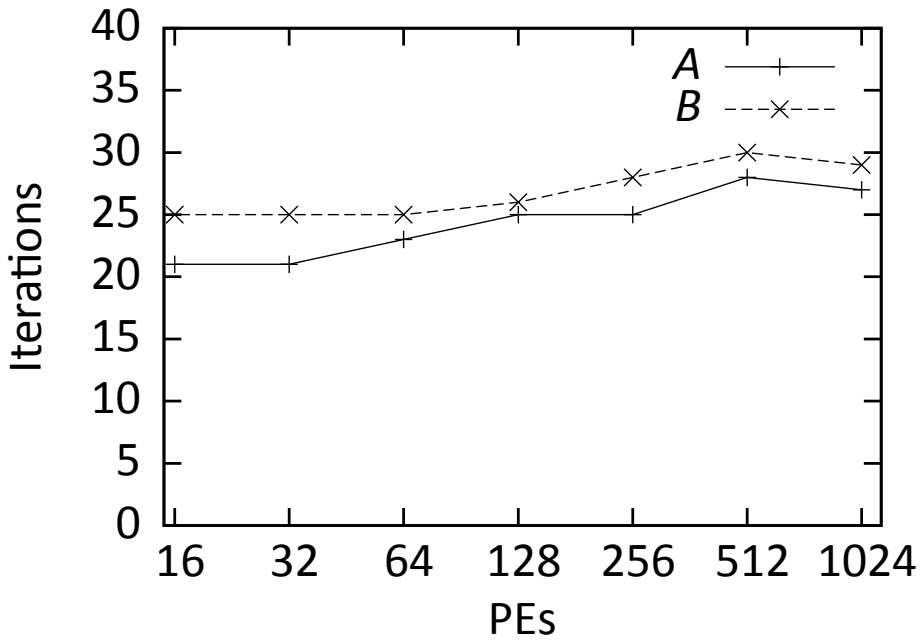
Mesh partitions

Coarsening Quality

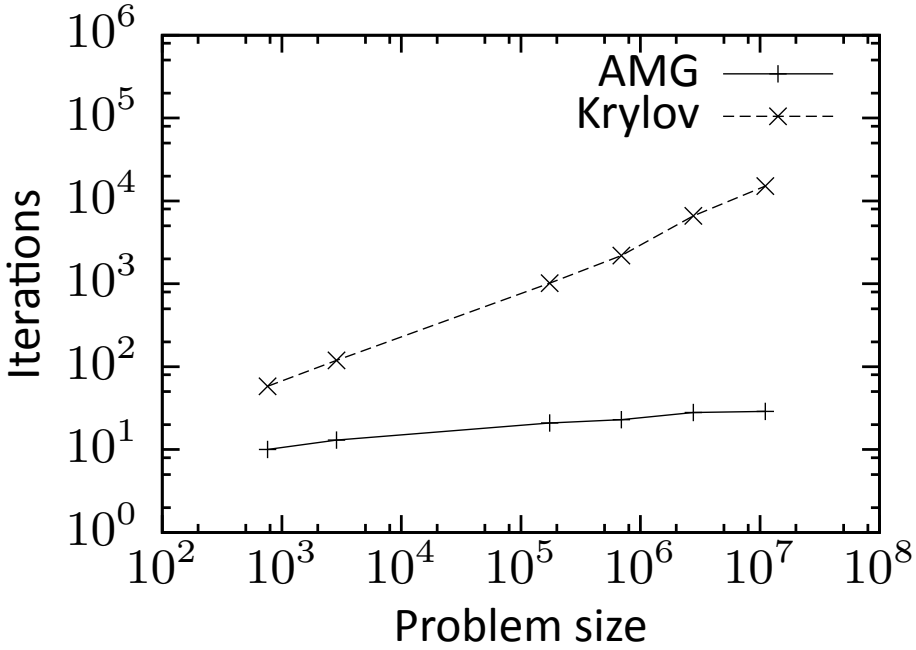
Asymptotically optimal complexity

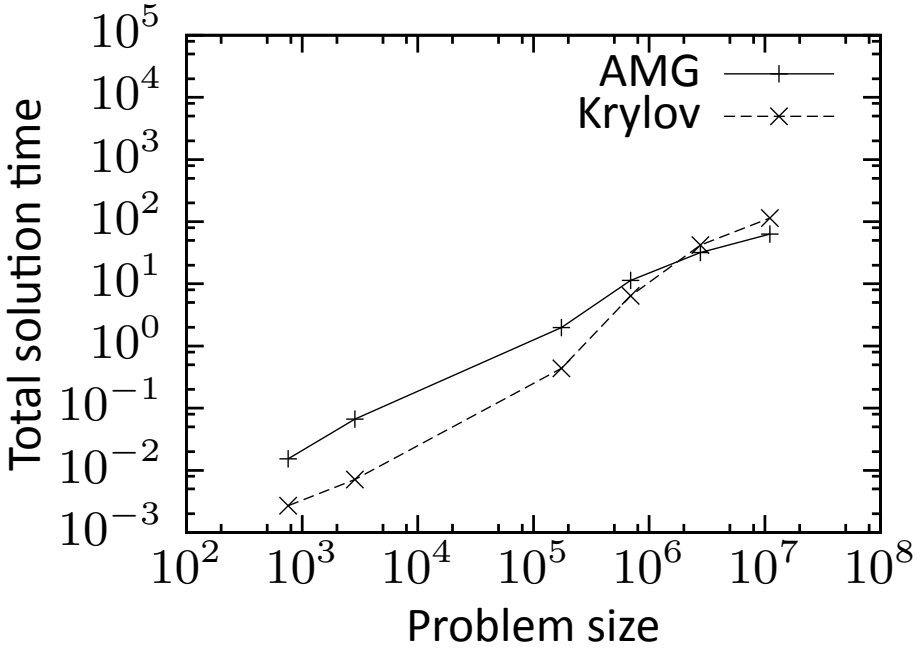
- Number of V-cycles independent of problem size
- Number of V-cycles independent of PEs

Benchmark solved for two matrix sizes and diff. numbers of PEs



Performance





Performance

					AMG				Krylov	
PEs	n	C_{op}	C_{g}	l	Iters	t_{setup}	t_{solve}	t_{tot}	Iters	t_{tot}
1	759	1.821	1.480	2	10	0.008	0.008	0.015	58	0.003
4	2868	2.137	1.552	4	13	0.030	0.037	0.067	119	0.007
32	174144	2.265	1.529	7	21	0.712	1.265	1.977	1023	0.436
128	693888	2.098	1.495	8	23	3.779	7.635	11.414	2196	6.432
512	2770176	2.065	1.486	9	28	8.190	23.601	31.791	6624	42.117
1024	11069952	2.044	1.480	10	29	11.476	51.787	63.263	15139	114.841

- AMG overhead costs too high for small matrices
- AMG setup costs (t_{setup}) less than solve (t_{solve})
- Operator (C_{op}) and grid complexity (C_{g}) doesn't grow
- AMG iterations doesn't grow too much with large numbers of PEs

Summary

- New parallel formulation of Ruge-Stüben
  - Not possible to formulate using MPI
- Retains similar properties as the serial algorithm
- Easier implementation due to the PGAS abstraction

Future work

- Reduce AMG overhead
- Optimized collective operations
  - No subset collectives in std. UPC
  - Handwritten versions not optimized
- Investigate UPC atomics
  - Our Cray compiler didn't support it
  - Alternative instead of strict variables