Distributed NVAMG

Design and Implementation of a Scalable Algebraic Multigrid Framework for a Cluster of GPUs

Istvan Reguly (istvan.reguly at oerc.ox.ac.uk)
Oxford e-Research Centre

NVIDIA – Summer Internship Project

Jonathan Cohen, Robert Strzodka and lots of others.
Algebraic Multigrid

- Solving sparse linear systems:
  \[ Ax = b \]

- 100K – 400M unknowns (CFD, ResSim)

- Iterative solver, trying to minimise
  \[ \| A\tilde{x} - b \|_k \]

- Usually part of a non-linear outer solver, tolerance may be low or high
Why AMG? - Smoothers

Error of initial guess

Error after 5 steps

Error after 10 steps

Error after 15 steps
Algebraic Multigrid

Multi-level sparse iterative solver
Algebraic Multigrid

- Setup: Fine matrix $A$
  - Compute Interpolator and Restrictor $P, R (= P^T)$
  - Compute coarse matrix $A_c = RAP$
  - Repeat
  - Set up smoothers for every level

- Solve: input: $x, b$
  - Smooth $x$
  - Restrict residual $b_c = R(Ax - b)$
  - Repeat
  - Prolongate & apply correction $x_+ = Px_c$

- Convergence independent of problem size $O(N)$
NVAMG

- Main focus on AMG as a preconditioner
- Classical and aggregation AMG, implements Krylov solvers, various smoothers
  - Jacobi, Gauss-Seidel, Diagonal ILU0 etc.
- Dynamic solver hierarchy, user specifies preconditioners, smoothers, coarse solvers etc.
  - FGMRES preconditioned by AMG with a DILU smoother and a PBiCGStab coarse solver, preconditioned by...
- Supports arbitrary block sizes (e.g. 4*4 u,v,w,p)
- ~200k lines, makes extensive use of libraries
  - Thrust, cuSPARSE, CUSP
AMG on the GPU

- Lots of sequential algorithms in the literature
  - SpMM, aggregation, Gauss-Seidel, ILU, coloring
  - Integer/logical algorithms
  - Difficult to parallelise and even then bandwidth limited

- Less and less work on coarser levels
  - Utilisation problems

- Expensive setup
  - Depending on tolerance, may be up to 60% of runtime
  - No way around it, convergence is not just a matter of time...

- Memory-limited (~5 GB/GPU)
Colored smoothers

Algorithm on vertices+edges updating vertices (in $x$)
Data races during updates (of $x$)

For $k = 1$ until convergence
For each row $i$

$$x_i^{k+1} = \frac{1}{a_{ii}} \left( - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^{k} + b_i \right)$$

Update solution on node $i$ based on already updated information 1...$i$-1 and old information $i+1$...$n$
Colored smoothers

Execute sets of vertices color by color

For $k = 1$ until convergence
For each row with color $i$

$$\bar{x}_{i}^{k+1} = \frac{1}{\bar{a}_{ii}} \left( - \sum_{j=1, j \neq n}^{n} \bar{a}_{ij} \bar{x}_{j}^{k+1} + \bar{b}_{i} \right)$$

No write/read conflicts!
Still, not quite the same…
Colored smoothers

- Colors 1-6
- Update result (spMV) for each color
- Forward sweep: Only use nonzeros: color < current color (DILU)
- Backward sweep: 6->1 colors > current color (DILU)

CUSPARSE bsrxmv

How do we get the coloring?

\[
\bar{x}_{i}^{k+1} = \frac{1}{\bar{a}_{ii}} \left( - \sum_{j=1, j \neq n}^{n} \bar{a}_{ij} \bar{x}_{j}^{k+1} + \bar{b}_{i} \right)
\]
Parallel coloring

- Very sequential problem (propagate information)
  - Parallel version will use more colors (local decision)

- Iterative algorithm: assign random number to uncolored nodes, if local maximum, assign current color
  - Jones-Plassman
Min-Max coloring

1. Assign random number to vertices
2. For i=1...
   1. Assign color i to local maxima
   2. Marked colored vertices as ‘done’
3. Stop at x% uncolored

6 colors – 6 iterations
Min-max:
6 colors – 3 iterations
Parallel coloring

- Very sequential problem (propagate information)
  - Parallel version will use more colors (local decision)

- Iterative algorithm: assign random number to uncolored nodes, if local maximum, assign current color
  - Jones-Plassman
  - Next, for local minima

- All nodes need the same information
  - Generate vector of random numbers
  - Use a hash (on node IDs)
  - Multi-hash: several random values per node – several possible colors
Graph matching - Aggregation

- Graph matching: set of edges so that no two share a vertex
- Select neighbors that are “strongly” connected, merge them together on the next coarser level
- Trivial serial implementation, similar to coloring
- One-phase handshaking: extend a hand to your strongest neighbor, if a handshake happens, we have an aggregate
  - Repeat with unmatched
- Two-phase handshaking
One-phase handshaking
Graph matching performance

Figure from Cohen, Castonguay @ GTC 2012
NVAMG vs. HYPRE (12 core Xeon)

- Reservoir case 1 (191k, 1.3m)
- Reservoir case 2 (350k, 2.2m)
- Reservoir case 3 (1.4m, 10m)
- Reservoir case 4 (289k, 1.66m)
- 9-pt Poisson (1m, 9m)
Other GPU algorithms

- SpMM with dynamic size hash-maps
  - See Julien Demouth @ GTC 2012 ID S0285

- For more on aggregation & coloring
  - Jonathan Cohen @ GTC 2012 ID S0332

- Classical AMG – PMIS, D1, D2 selectors
  - Simon Layton @ GTC 2012 ID S0305
Distributed NVAMG

\[ \text{B2L}(\alpha, \beta) \rightarrow \text{L2H}(\alpha, \beta) \]

\[ \text{L2H}(\beta, \alpha) \rightarrow \text{B2L}(\beta, \alpha) \]
- Node id
- Neighbors
- Index ranges
- B2L
- Halo rows
- Comms module
Internal representation

- Numbering from 0…N-1
- Renumbering & reordering in a way that interior nodes are first, then the boundary ones
  - Support latency hiding
- Mapping from original numbering to new one and from new one back to global indices
- Halo rows appended to the matrix, neighbor-by-neighbor, renumbered but not reordered, throw away unknown edges
Communications

- It’s not as difficult as it sounds
  - Some bookkeeping
  - In ‘user code’ insert ‘I need up-to-date data’
  - Everything else happens behind the scenes

- Isolated communications module
  - Nothing else knows about the distributed environment
  - Plug-in style modules for OpenMP, MPI
  - Can do all sorts of tricks: P2P, GPU Direct, blocking communications etc.
GPU Direct vs. Host buffers - Solve

Matrix1 ~80M
Matrix2 ~80M
Matrix3 ~50M
Matrix4 ~20M
Matrix5 ~90M
Matrix6 ~50M
Matrix7 ~200M
Matrix8 ~310M

2 GPUs
Represented by a vector: for every fine node, which coarse node it is aggregated into.
Galerkin product \( (A^c = R \ A^f \ P) \)

- The Galerkin product merges edges between fine nodes that went into the same aggregate
- Perform spMM on locally consistent matrices
- Some values in the coarse halo may be different
  - Update them
Algorithmic differences

- No aggregation through boundaries
- No globally consistent coloring
  - Doing halo exchange between colors would be expensive
- Latency hiding
  - Separation of interior-boundary vertices
  - Init comms & execute interior -> wait -> execute boundary
  - Differences in both colored & non-colored smoothers
  - Occupancy & latency issues
  - Numerical stability issues
DILU sensitivity to latency hiding

Row threshold for stopping latency hiding (2 nodes)

- Matrix1 ~80M: 3.2%
- Matrix2 ~80M: 3.2%
- Matrix3 ~50M: 59.9%
- Matrix4 ~20M: 52.9%
- Matrix5 ~90M: 13.0%
- Matrix6 ~50M: 3.0%
- Matrix7 ~200M: 27.5%
- Matrix8 ~310M: 42.9%
Scaling 1 - Setup

- Scaling with FGMRES preconditioned by Jacobi/GS/DILU, latency hiding turned on/off, no AMG
- 3D Poisson problem

![Graph showing speedup against number of GPUs for different preconditioners (Jacobi, GS, DILU).]
Scaling 2 – Incompressible N.S.

- Set of incompressible Navier-Stokes matrices (4*4 blocks for u,v,w,pressure)
- Numerically difficult
- Sensitive to DILU/GS decoupling due to latency hiding
- Level cap at something “sensible”
- Inside non-linear solver, so solved to high tolerance (0.1)
  - Setup time is important
- By default deterministic (strict coloring)
  - Often it can be turned off for some performance increase
Scaling 2 – Total speedup

- Matrix1 ~80M
- Matrix2 ~80M
- Matrix3 ~50M
- Matrix4 ~20M
- Matrix5 ~90M
- Matrix6 ~50M
- Matrix7 ~200M
- Matrix8 ~310M
Optimisations

- Partitioning
  - Currently we use naïve partitioning – bad edge cuts

- Support for already partitioned problems
  - Since the underlying solver may already do partitioning

- Partition migration
  - Less communication, more work on coarser levels
  - Better numerical behavior

- Tuning, tuning, tuning...

- Public release TBA