Non-ML preconditioners and ILU(0)

Eloy Romero[*], Heather Switzer[+], Andreas Stathopoulos[+] Kostas Orginos[*o]

SciDAC-5 collaboration: Sherry Xiaoye Li (LBNL)

[+]Computer Science Department, William & Mary
[o] Physics Department, William & Mary
[*]Jefferson Lab

December 2, 2022

Motivation

LQCD linear solver challenge:
"Critical slowing down", i.e., condition number increases with lattice volume

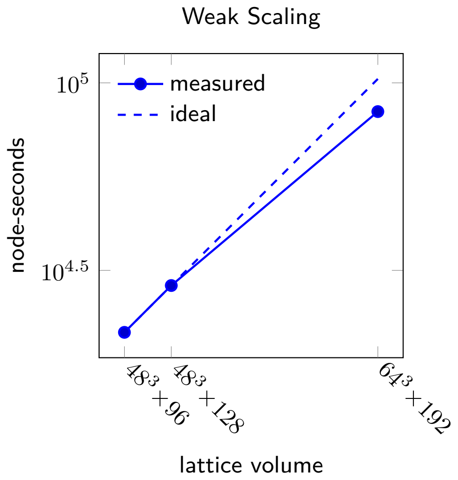
Denser near null spectrum creates problems for HMC, propagators, higher variance in trace computations, etc

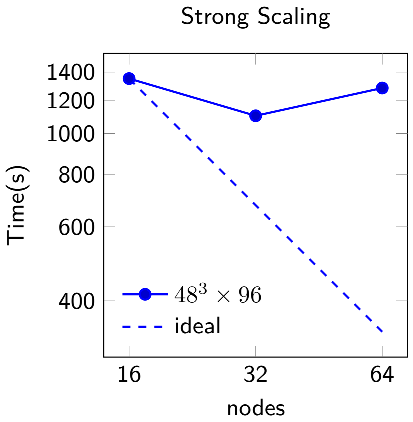
Multigrid has bought us time, but

- Volumes are further increasing
- Coarsest level matrix is much denser
- Coarsest level matrix is very small so parallelism is communication bound
- There is no efficient preconditioner to help at the coarsest level
- Solutions at the coarsest level takes most of the time

Motivation: Multigrid displays low parallel scalability

Performance example on Juwels Booster (4 GPUs A100 per node)
Our application needs to run on at least 32 nodes





Approaches to improve the performance

- Make the coarsest operator small enough to be solved with direct methods in a few nodes/devices
  - Large coarsening may deteriorate the capacity to reduce iterations
  - Direct methods for sparse matrices have poor parallel performance, eg on GPUs
  - Densify the matrix may be too expensive also
- Deflating with the smallest singular vectors of the course operator
  - Packed spectrum around the origin, slow decay of the singular vectors of $(V^T AV)^{-1}$
- Incomplete LU factorization
  - Lots of communication overhead for general nonzero patterns
  - What is the ordering that reduces communications overheads? How effective is it?

Parallelizing $L^{-1}\mathbf{x}$ and $U^{-1}\mathbf{x}$

Forward solution for block sparse $\mathbf{y} = L^{-1}\mathbf{x}$, where $L_{i,j}$ is a block, inherently sequential:

```
for $i = 1 : n$,
    $\mathbf{y}_i = L_{i,i}^{-1}(\mathbf{x}_i - L_{i,1:i-1}\mathbf{y}_{1:i-1})$
```

The i row depends on the solution of the nonzero elements of $L_{i,1:i-1}$

Parallelizations proposed:

- levels (by Anderson, Saad), update all rows at the same distance from the first row
- synchronization-free (by W. Liu, J. Hogg): working by columns and asynchronous communications of $\mathbf{y}_i$ while updating the subsequent rows $L_{i:n,i}\mathbf{y}_i$

But

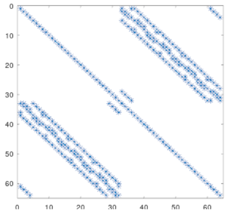
- We can only afford a few halo exchanges and with a limited volume of communications for the backward and forward solution
- We propose to work with matrix orderings based on coloring to minimize the communication exchanges

Parallelizing $L^{-1}\mathbf{x}$ and $U^{-1}\mathbf{x}$: turn them into sparse matvecs

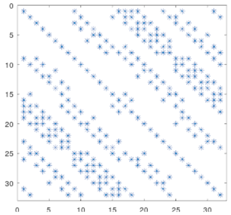
- Start with the recursive formula of the inverse for a 2×2-block matrix:

$$\begin{bmatrix} L_{0,0} & \\ L_{1,0} & L_{1,1} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \end{bmatrix} = \begin{cases} \mathbf{y}_0 = L_{0,0}^{-1}\mathbf{x}_0 \\ \mathbf{y}_1 = L_{1,1}^{-1}(\mathbf{x}_1 - L_{1,0}\mathbf{y}_0) \end{cases}$$

Example:



8 by 8 lattice



Schur complement on the even sides ($A_{\text{eo}}A_{\text{oe}}$)

- 2 colors (even-odd or red-black ordering)
- $L_{0,0}$ and $L_{1,1}$ are block diagonal, easy to invert
- The cost of applying implicitly L^{-1} is $L_{1,0}\mathbf{y}_0$

- Four colors, what to do now?

Parallelizing $L^{-1}\mathbf{x}$ and $U^{-1}\mathbf{x}$: turn them into sparse matvecs

$$\begin{bmatrix} L_{0,0} & \\ L_{1,0} & L_{1,1} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \end{bmatrix} = \begin{cases} \mathbf{y}_0 = L_{0,0}^{-1}\mathbf{x}_0 \\ \mathbf{y}_1 = L_{1,1}^{-1}(\mathbf{x}_1 - L_{1,0}\mathbf{y}_0) \end{cases}$$

1. The leaves of the recursion (at level 0) are the submatrices of the diagonal color blocks
2. But recursion does not have to reach the leaves at level 0.
   Stop at level $\ell > 0$, and compute explicitly $L_{(\ell)}^{-1}$.
   Then $L_{(\ell)}^{-1}x$ requires only 1 communication point, but with a denser matrix
3. We want to find an ordering of the colors to minimize the fill-in for $L_{(\ell)}^{-1}$

For the Schur complement on a 4D lattice, the number of edges between a node of color i and nodes of color j:

Col	0	1	2	3	4	5	6	7
0	1							
1	4	1						
2	4	4	1					
3	4	4	4	1				
4	8	4	4	4	1			
5	4	8	4	4	4	1		
6	4	4	8	4	4	4	1	
7	4	4	4	8	4	4	4	1

- Note that color pairs (0,4), (1,5), (2,6), (3,7) have a heavier connection, and they should not be in the same partition for levels less than ℓ
- The colors inside the groups (0,1,2,3) and (4,5,6,7) can be in any order

Parallelizing $L^{-1}\mathbf{x}$ and $U^{-1}\mathbf{x}$: turn them into sparse matvecs

Cost of stopping the recursion at different levels for a Schur complement on a 4D lattice:

Explicit level	# Halo exchanges	Nonzeros per row (FLOPS)	
$\ell = 0$	7	136	(the $L^{(i,i)}$ are explicitly inverted)
$\ell = 1$	7	136	
$\ell = 2$	3	232	(good trade-off)
$\ell = 3$	1	1368	(the whole L is explicitly inverted)

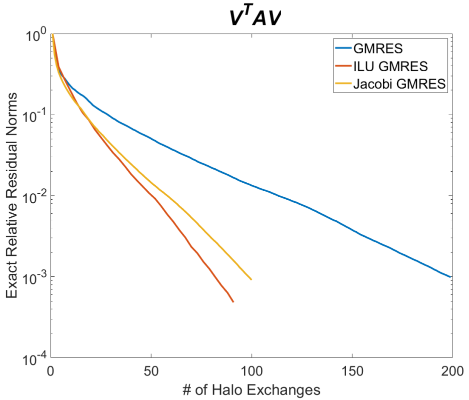
The number of edges between a node of color i and nodes of color j

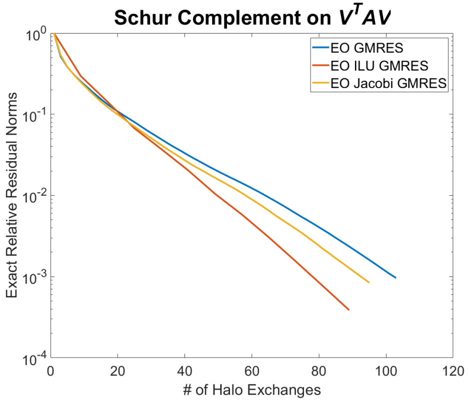
$L^{(i,j)}$	0	1	2	3	4	5	6	7
0	1							
1	4	1						
2	4	4	1					
3	4	4	4	1				
4	8	4	4	4	1			
5	4	8	4	4	4	1		
6	4	4	8	4	4	4	1	
7	4	4	4	8	4	4	4	1

${L^{-1}}^{(i,j)}$	0	1	2	3	4	5	6	7
0	1							
1	4	1						
2	12	4	1					
3	36	12	4	1				
4	64	24	12	4	1			
5	120	64	24	12	4	1		
6	156	92	64	24	12	4	1	
7	204	156	120	64	36	12	4	1

Preconditioning comparisons on coarse operators

Example of convergence history with the coarsest operator:





However, smoothing at coarsest level requires acurracy O(0.3) currently!

To explore

- Better quality prolongators so that better accuracy of coarsest grid is useful?
- altenative colorings?
- Engineer an effective implementation of recursive triangular solves
- Use ILU at higher levels, as a smoother or preconditioner?
- Better parallelizable variants for the triangular solves
- Can the ILU(0) variants be used as variance reduction for trace computations?
- Applicability to HMC with or w/o domain decomposition?
- Explore efficient ways to construct the ILU(0)

Example: Computing the smallest SV with inexact eigensolver

- Using GD+k in PRIMME with GMRES as the operator to compute the 10 largest magnitude singular values of the coarse operator $A_c^{-1} = (V^T A_c V)^{-1}$
- GMRES unpreconditioned on A_c, preconditioned with ILU (L, U) on A_c, on the Schur complement (S), and ILU on the Schur complement (L_S, U_S)

	GMRES	Halo exchanges		
Operator	inner its.	per GMRES it.	Total	Speedup
gmres(A_c)	13457	1	13457	1
gmres(A_c,L,U) [A_c nat]	1467	–	–	–
gmres(A_c,L,U) [A_c EO]	2761	3	8283	1.6
gmres(S)	3018	2	6036	2.2
gmres(S,L_S,U_S) [S nat]	551	–	–	–
gmres(S,L_S,U_S) [S col. ord.]	550	8	4400	3.1

Halo exchanges for the ILU factors:

- A_c, L^{-1}, and U^{-1} take 1
- for the Schur complement on the coloring ordering: S takes 2, and L_S^{-1} and U_S^{-1} take 3