Direct coarsest level solves in DDalphaAMG

Henning Leemhuis

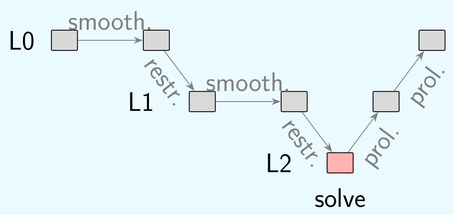
BERGISCHE UNIVERSITÄT WUPPERTAL

DDαAMG and Emerging Challenges

Modern lattice QCD simulations produce linear systems that are increasingly hard to solve, with a rapidly growing **condition number**. Two effects drive this:

- **Twisted mass**: shifts the spectral properties of the Dirac operator and introduces problematic small eigenvalues (cf. [1])
- **Finer lattices and larger volumes**: more degrees of freedom and a wider eigenvalue spectrum.

DDαAMG [4] is a domain-decomposed adaptive algebraic multigrid solver that separates the error by frequency: a **smoother** damps high-frequency components, while low-frequency components are passed to a **coarser level** for correction (see figure).

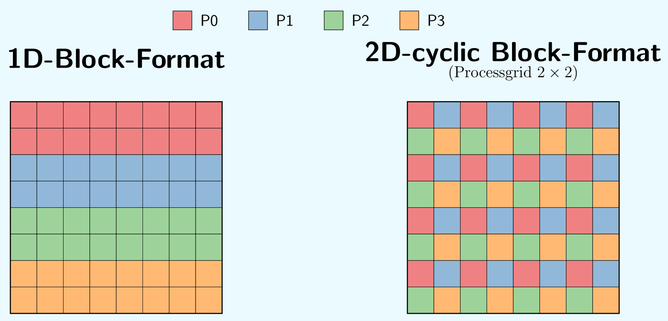


These low-frequency modes coincide with the small eigenvalues that stall conventional Krylov solvers – making the coarse-grid solve the dominant cost.

Approach: Direct Solves and Implementations

Direct solvers provide a robust alternative to iterative coarse-grid solvers, but their cost grows rapidly with system size. Making a direct solve feasible therefore requires that we **reduce the coarse-grid problem sufficiently** – through multiple multigrid levels and/or more aggressive coarsening.

We solve the coarsest-level system directly with **ScaLAPACK** [2], exploiting its distributed-memory parallelization. To enable efficient parallel linear algebra, the coarse-grid matrix is redistributed from a **1D block** to a **2D block-cyclic** layout across the participating processes.



The factorization consists of an **LU decomposition** followed by the **explicit construction of the inverse**,

$$A_c = LU, \qquad A_c^{-1} = U^{-1}L^{-1}.$$

This adds to the factorization cost, but reduces every subsequent coarse-grid solve to a single **matrix-vector multiplication**, executed efficiently via optimized BLAS routines:

$$x_c = A_c^{-1} b_c.$$

DDαAMG normally uses **agglomeration** – fewer processes on coarser levels – to limit communication overhead and avoid a communication-dominated coarse-grid step. For the direct solve, however, we want **all available processes** accessible to ScaLAPACK: the 1D-to-2D redistribution is therefore folded directly into the existing transfer routine, at no extra cost.

Stationary Method and Multiple RHS

Stationary preconditioner. With an iterative coarse solve, the number of coarse-grid iterations can vary between applications of the preconditioner, making it effectively non-stationary. A direct solve with a fixed factorization removes this variability: the same coarse-grid operation is applied every time, so the preconditioner becomes **stationary** – simplifying convergence behavior for Krylov-subspace methods.

Multiple right-hand sides. The direct approach is particularly attractive for **multiple right-hand sides (MRHS)**,

$$Ax_i = b_i, \qquad i = 1, \ldots, N_{\mathrm{RHS}},$$

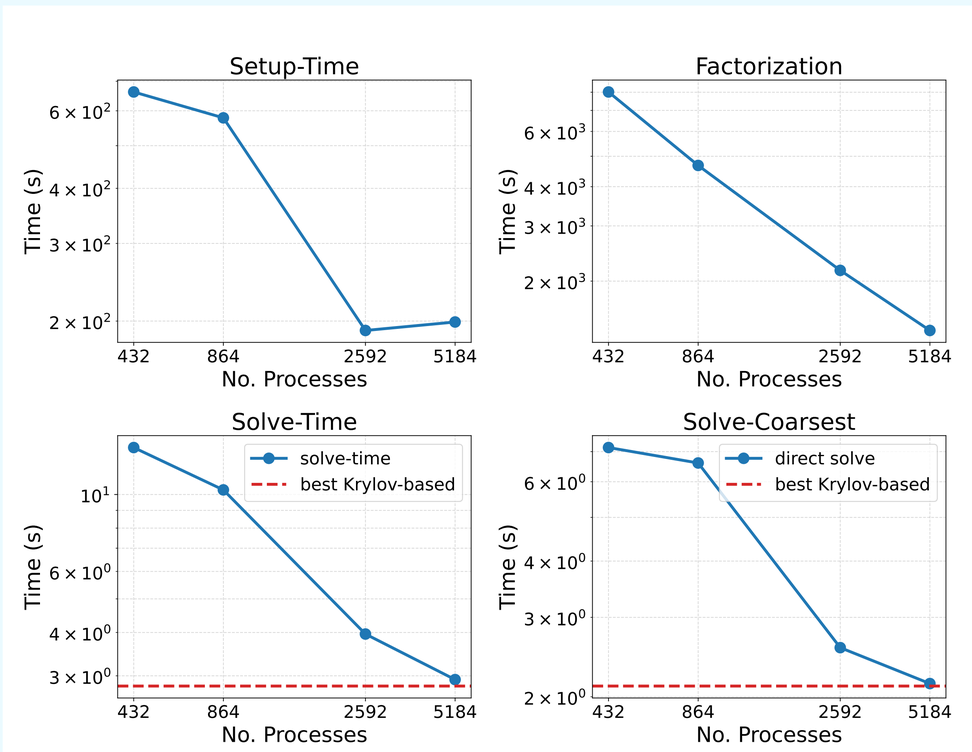
since the coarse solve then naturally generalizes from a matrix-vector to a matrix-matrix multiplication:

$$x_c = A_c^{-1} b_c \quad \longrightarrow \quad X_c = A_c^{-1} B_c.$$

This lets all right-hand sides be processed simultaneously with optimized BLAS/ScaLAPACK routines, amortizing the higher one-time factorization cost over many solves.

Performance

We study the **strong scaling** of the direct coarse-grid solver from 432 to 5184 processes. For comparison we show the best timing of a Krylov-based method (cf. [3]):



- **Setup-Time** (no direct solve yet): scales well for the first three data points. Beyond 2592 processes, **agglomeration** caps the number of processes active on the coarsest level – at 5184 total processes, only 2592 of them participate there. The coarsest-level cost then dominates the setup, so the last data point brings essentially no further speedup over the third.
- **Factorization**: the one-time ScaLAPACK **LU factorization** of the coarse-grid matrix scales well across the entire range, since it runs on however many processes are active on the coarsest level.
- **Solve-Time**: with the direct coarse solve in place, the overall solver again scales well throughout. Although agglomeration still limits the coarsest level itself to 2592 processes at the largest process count, the coarse-grid solve $x_c = A_c^{-1} b_c$ is a **matrix-vector multiplication that can use all 5184 processes** – decoupling the solve's parallel efficiency from the agglomeration limit that constrains setup and factorization.
- **Solve-Coarsest**: the runtime of the coarsest level in isolation shows moderate-to-good scaling. This is where the cost of **communication** – in particular the redistribution between the 1D and 2D block-cyclic formats – becomes increasingly visible as process counts grow.

Outlook

Red-black Schur complement. The lattice's nearest-neighbor coupling allows the coarse-grid system to be reordered **red-black**, splitting it into two subsets:

$$A_c = \begin{pmatrix} A_{RR} & A_{RB} \\ A_{BR} & A_{BB} \end{pmatrix}.$$

Since same-color sites are not directly coupled, A_{RR} and A_{BB} are block-diagonal. Eliminating one color (e.g. black) yields the **Schur complement**

$$S = A_{RR} - A_{RB} A_{BB}^{-1} A_{BR},$$

which halves the effective problem size. The full inverse of A_c then follows from standard block-inversion identities. The remaining operations reduce mostly to matrix-matrix multiplications, while inverting the block-diagonal submatrices is **cheap and independent** – each block is small and can be inverted on its own, in parallel across blocks.

GPU acceleration. These matrix-matrix multiplications are well suited to **GPU** architectures. Porting the direct coarse-grid operations to GPUs, using highly optimized GPU linear-algebra libraries, could substantially accelerate this step further.

Reducing communication overhead. At the largest process counts, the **redistribution** between the 1D block and 2D block-cyclic formats accounts for a growing share of the coarsest-level time (cf. Performance). Overlapping this communication with computation, or reducing how often the redistribution is required, is a promising direction for improving scalability further.

[1] C. Alexandroua, C. Bacchio, J. Finkenrath, A. Frommer, K. Kahl, and M. Rottmann. "Adaptive aggregation-based domain decomposition multigrid for twisted mass fermions". (2016). arXiv: 1610.02370v2 [hep-lat].

[2] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. "ScaLAPACK: a linear algebra library for message-passing computers". *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing (Minneapolis, MN, 1997)*. SIAM, Philadelphia, PA, 1997, 15 pp.

[3] J. Espinoza-Valverde, A. Frommer, G. Ramirez-Hidalgo, and M. Rottmann. "Coarsest-level improvements of multigrid for lattice QCD on large-scale computers". *Comput. Phys. Commun.* (2023), p. 108869.

[4] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann. "Adaptive aggregation based domain decomposition multigrid for the lattice wilson dirac operator". (2014). arXiv: 1303.1377v3 [hep-lat].

BERGISCHE UNIVERSITÄT WUPPERTAL