Solvers for Wilson fermions in Grid

Felix Ziegler

The University of Edinburgh

Software Development & Machines, Mo August 8, 2022

Lattice2022 - August 8-13, 2022, Bonn

Motivation: $D\psi = \eta$

- generate physical point gauge ensembles with (stabilized) Wilson clover-improved fermions using **HMC**

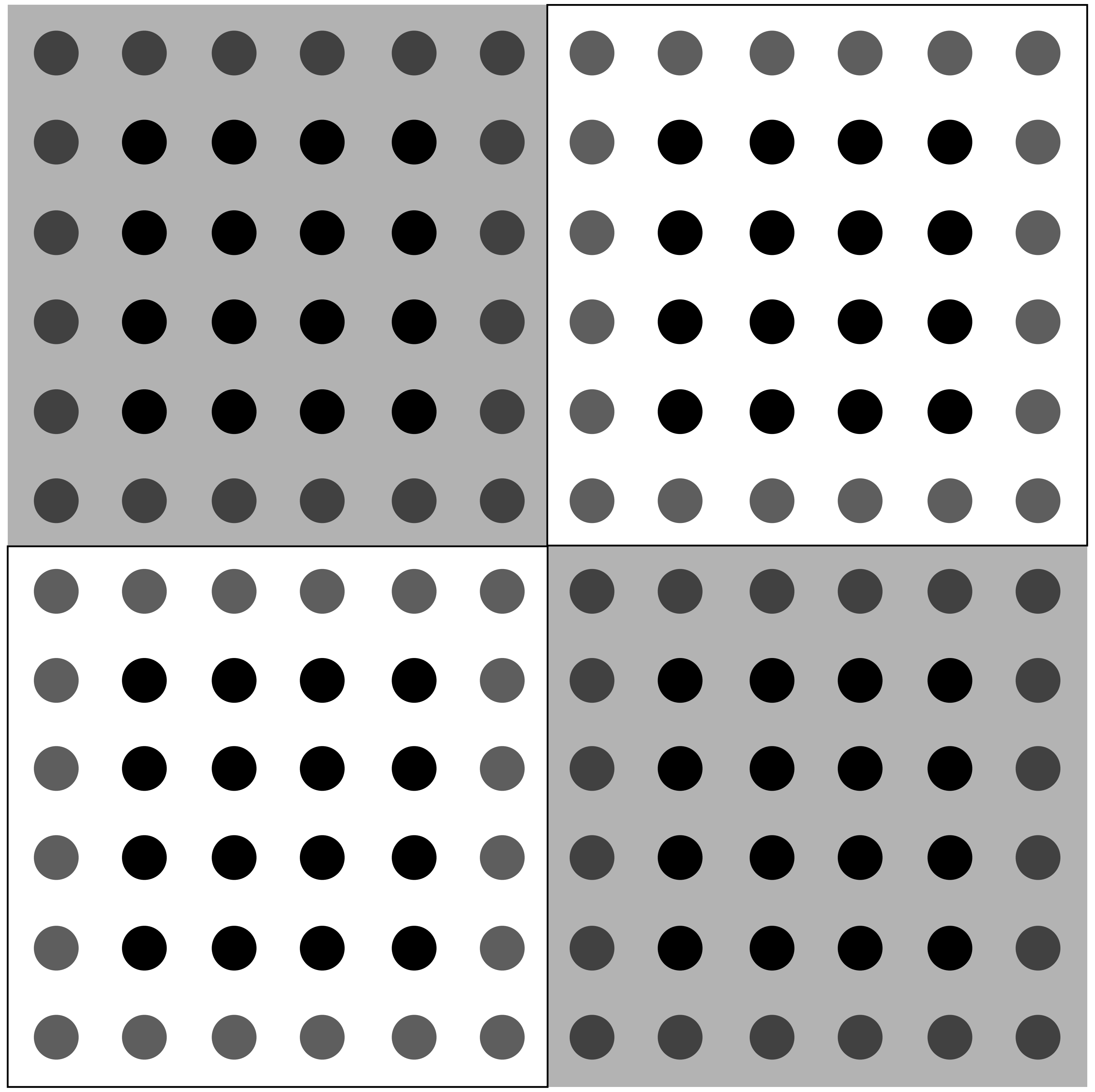
- fermion matrix inversion based on Krylov subspace solvers requires good **preconditioners**

- Ideas by Lüscher to use domain decomposition methods, **Schwarz alternating procedure (SAP) +**
  (inexact) Deflation (DFL), see Comput.Phys.Commun. 156 (2004) 209-220 and *JHEP* 07 (2007) 081

- **Advantage:** DFL can be cheaply combined with HMC, update deflation subspaces

- implemented in **openQCD**,
  see https://luscher.web.cern.ch/luscher/openQCD/

- **implement SAP + DFL in Grid (run on CPUs + GPUs)**



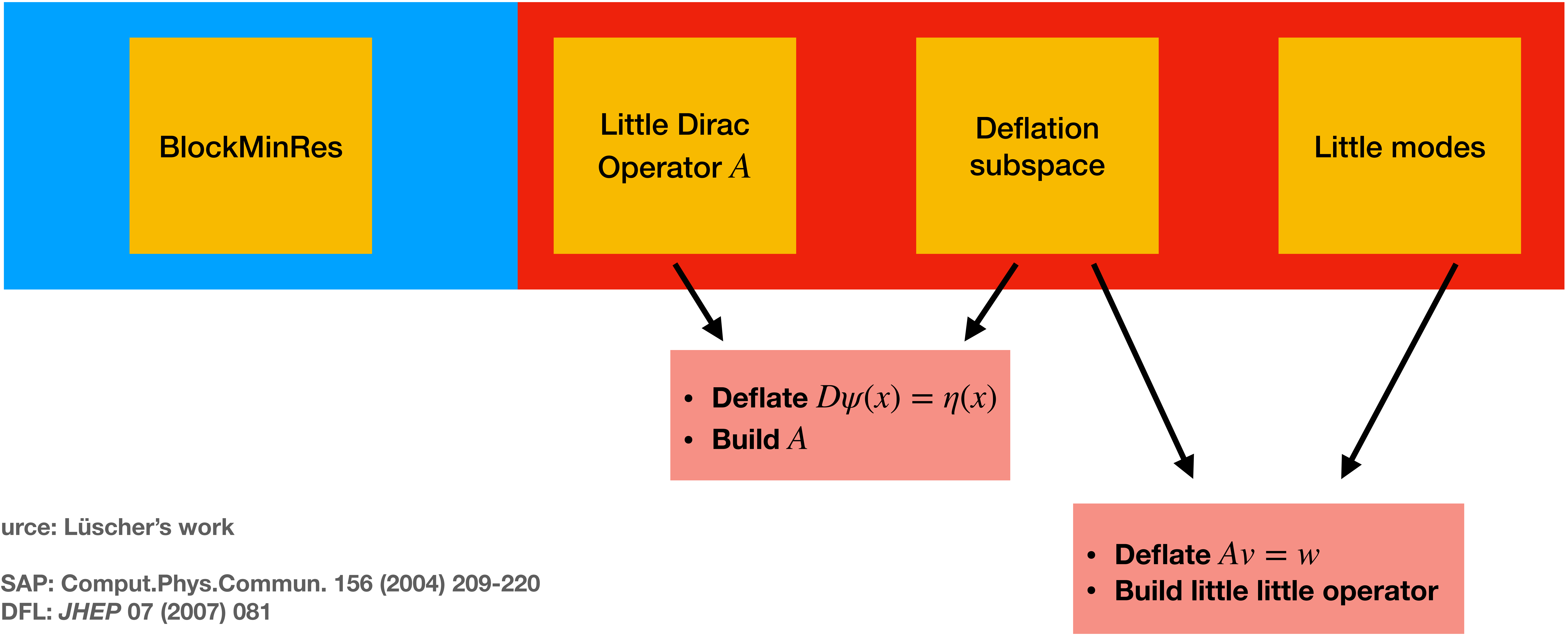
Grid

- General Lattice QCD library in C++
- Runs on many different architectures (CPU + GPU)
- **Boyle, Yamaguchi, Cossu, Portelli, 1512.03487 [hep-lat], „Grid: A next generation data parallel C++ QCD library“**
- https://github.com/paboyle/Grid
- many contributors across the lattice community
- see also https://github.com/lehner/gpt and
  C. Lehner’s talk (Mo, August 8 14:20, Software & Machines)

Implementation Plan

SAP

Inexact Deflation (DFL)



BlockMinRes

Little Dirac Operator A

Deflation subspace

Little modes

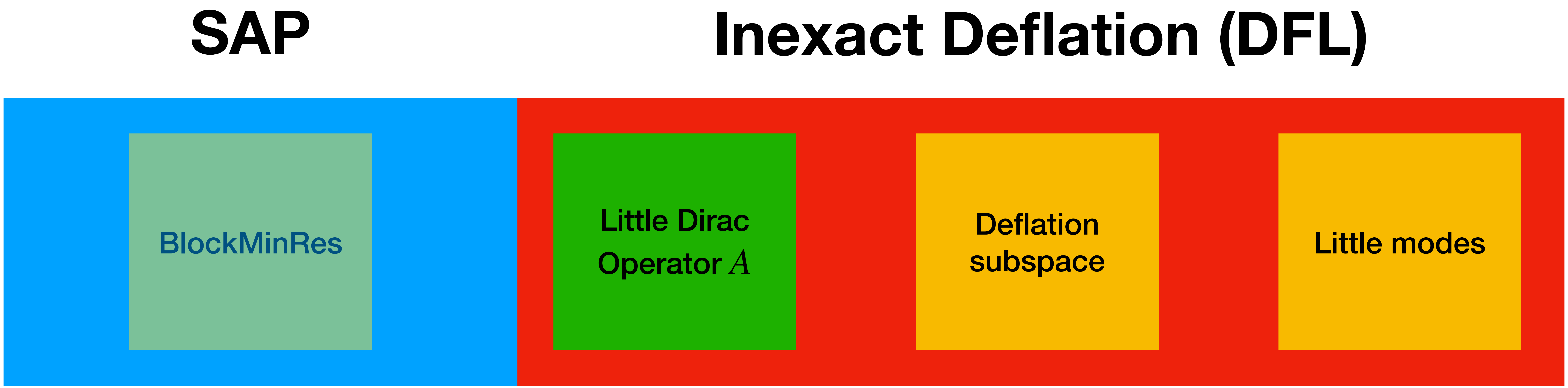
- Deflate $D\psi(x) = \eta(x)$
- Build A

- Deflate $Av = w$
- Build little little operator

Source: Lüscher's work

- SAP: Comput.Phys.Commun. 156 (2004) 209-220
- DFL: *JHEP* 07 (2007) 081

Implementation Plan

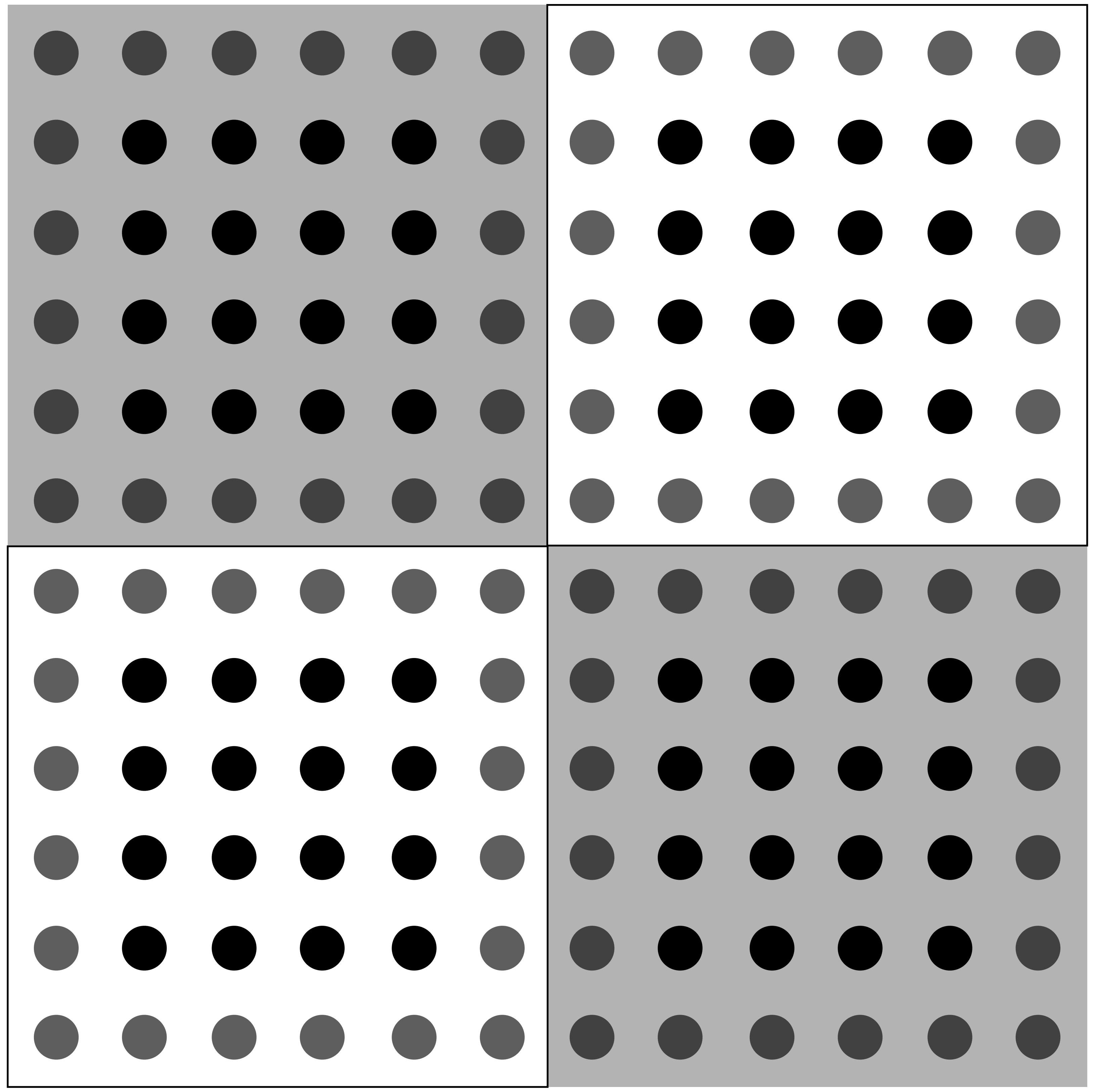


- Little Dirac operator already implemented (Grid/iterativeCoarsenedMatrix.h)
- also used in **Multi-Grid** code for Wilson Clover fermions, see Richtmann, Boyle, Wettig, *PoS* LATTICE2018 (2019) 032
- for news on Multi-Grid checkout **Nils Meyer’s poster (Tue, August 9 19:00, Poster Session A)**
- for SAP parts of the blockProject and blockPromote functions are being used

Status of SAP in Grid

Details on SAP

- SAP preconditioner in single precision
- Apply **SAP cycles** in alternating way to even (odd) domain
  - Step 1: **block minimal residue** algorithm
    see Y. Saad, Iterative methods for sparse linear systems,
    2nd ed. (SIAM, Philadel- phia, 2003),
    see also http://www-users.cs.umn.edu/~saad/



Details on SAP

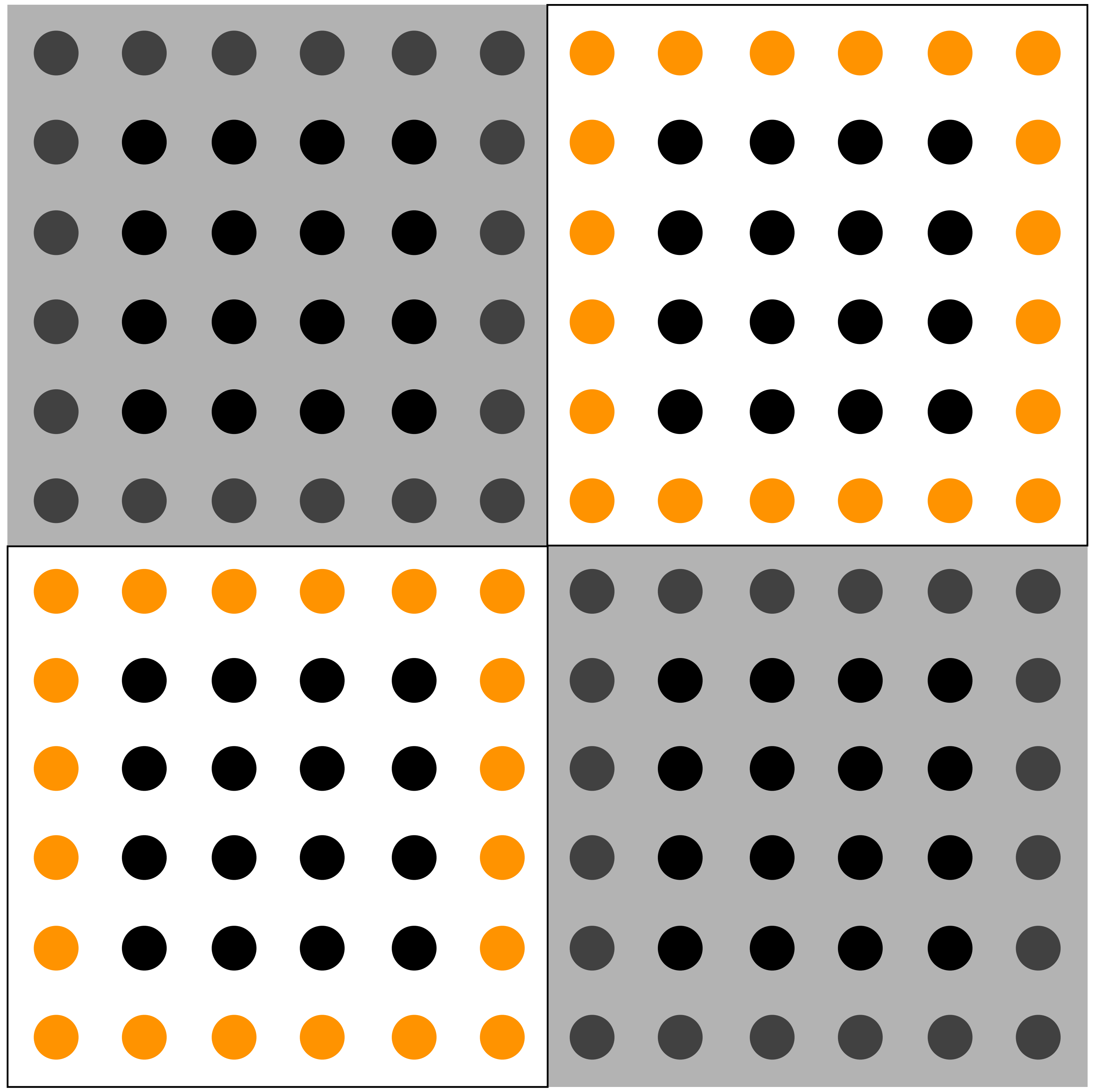
- SAP preconditioner in single precision

- Apply **SAP cycles** in alternating way to even (odd) domain

  - Step 1: **block minimal residue** algorithm

    see Y. Saad, Iterative methods for sparse linear systems, 2nd ed. (SIAM, Philadel- phia, 2003), see also http://www-users.cs.umn.edu/~saad/

  - Step 2: update domain boundary (comm.)

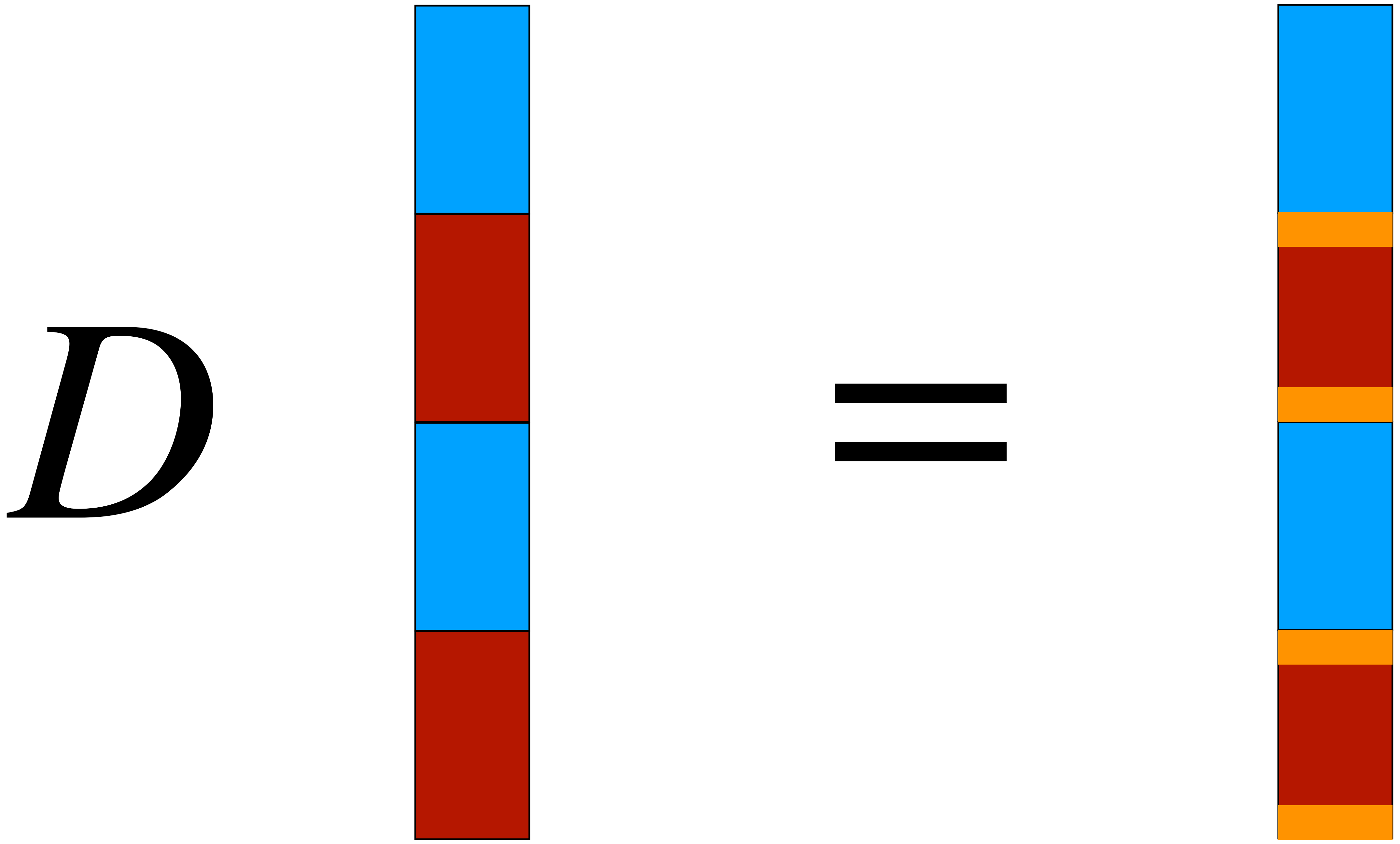


Strategy for SAP in Grid

- use standard **global spinor fields** (LatticeFermionF) as work space to simulate block operations

- for the **block minimal residue algorithm** apply the global Dirac operator

- e.g. when solving on blue blocks, set red ones to zero by applying masks after each application of D



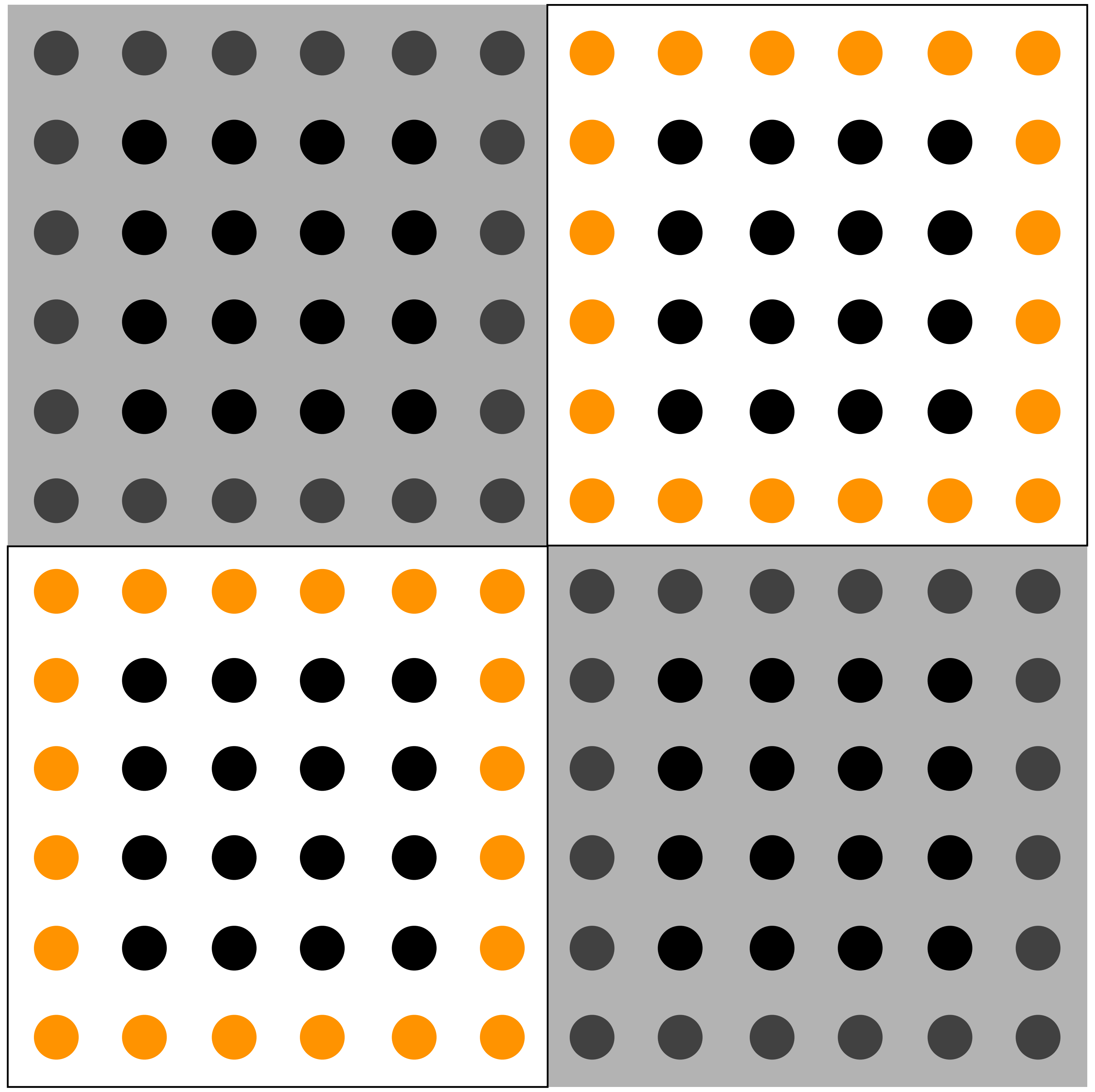
Strategy for SAP in Grid

- take care of exterior boundaries by applying masks

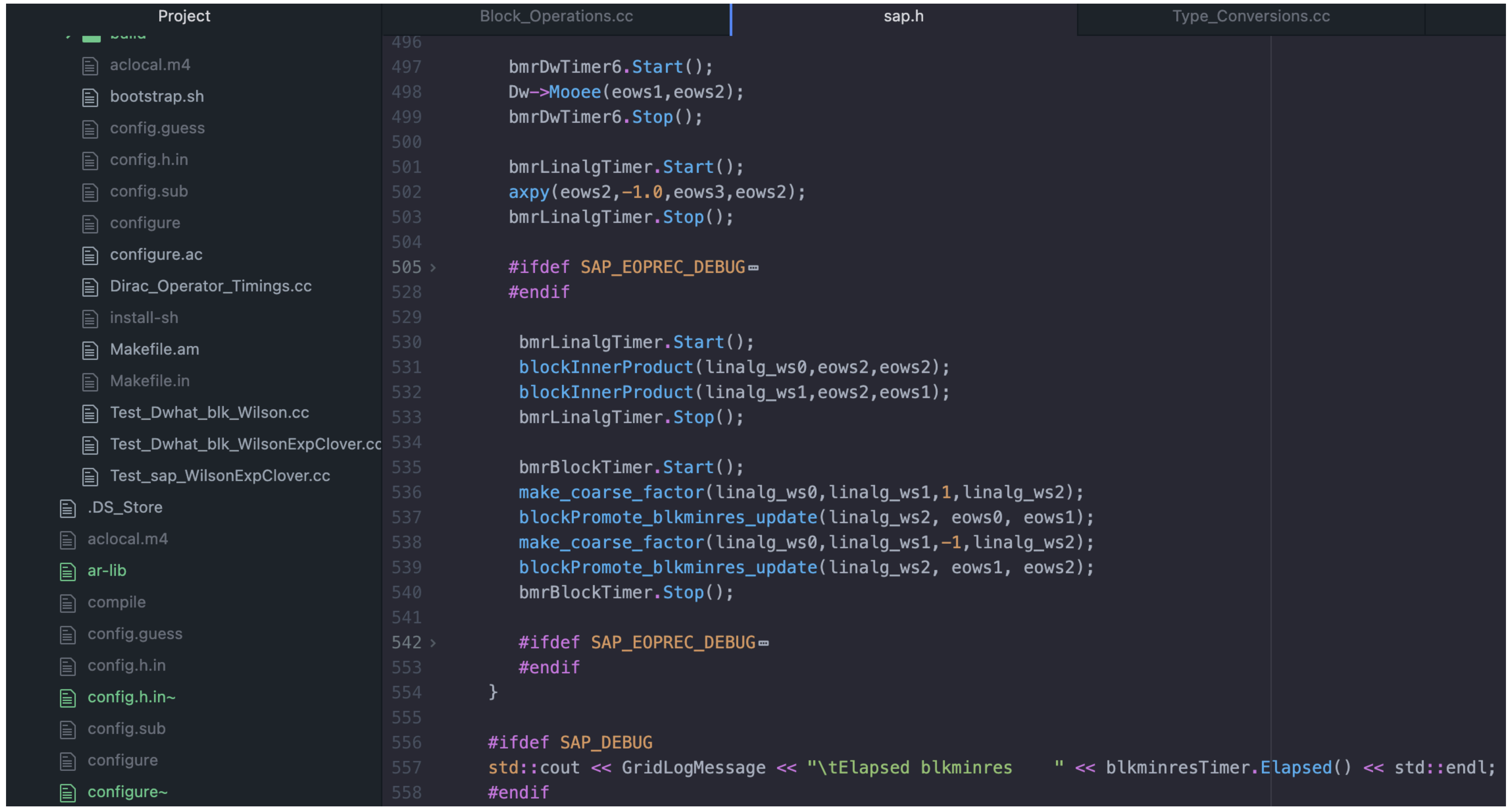
- **Performance issue:** when running BlockMinRes algorithm on even (odd) blocks the odd (even) blocks are also acted on, twice as many applications of Dirac operator per lattice point as in openQCD

→ FACTOR 2 SLOWER

Way out: block fields à la openQCD?



Strategy for SAP in Grid



```
      bmrDwTimer6.Start();
      Dw->Mooee(eows1,eows2);
      bmrDwTimer6.Stop();

      bmrLinalgTimer.Start();
      axpy(eows2,-1.0,eows3,eows2);
      bmrLinalgTimer.Stop();

      #ifdef SAP_EOPREC_DEBUG
      #endif

       bmrLinalgTimer.Start();
       blockInnerProduct(linalg_ws0,eows2,eows2);
       blockInnerProduct(linalg_ws1,eows2,eows1);
       bmrLinalgTimer.Stop();

       bmrBlockTimer.Start();
       make_coarse_factor(linalg_ws0,linalg_ws1,1,linalg_ws2);
       blockPromote_blkminres_update(linalg_ws2, eows0, eows1);
       make_coarse_factor(linalg_ws0,linalg_ws1,-1,linalg_ws2);
       blockPromote_blkminres_update(linalg_ws2, eows1, eows2);
       bmrBlockTimer.Stop();

       #ifdef SAP_EOPREC_DEBUG
       #endif
    }

    #ifdef SAP_DEBUG
    std::cout << GridLogMessage << "\tElapsed blkminres    " << blkminresTimer.Elapsed() << std::endl;
    #endif
```

- take care of exterior boundaries by applying masks

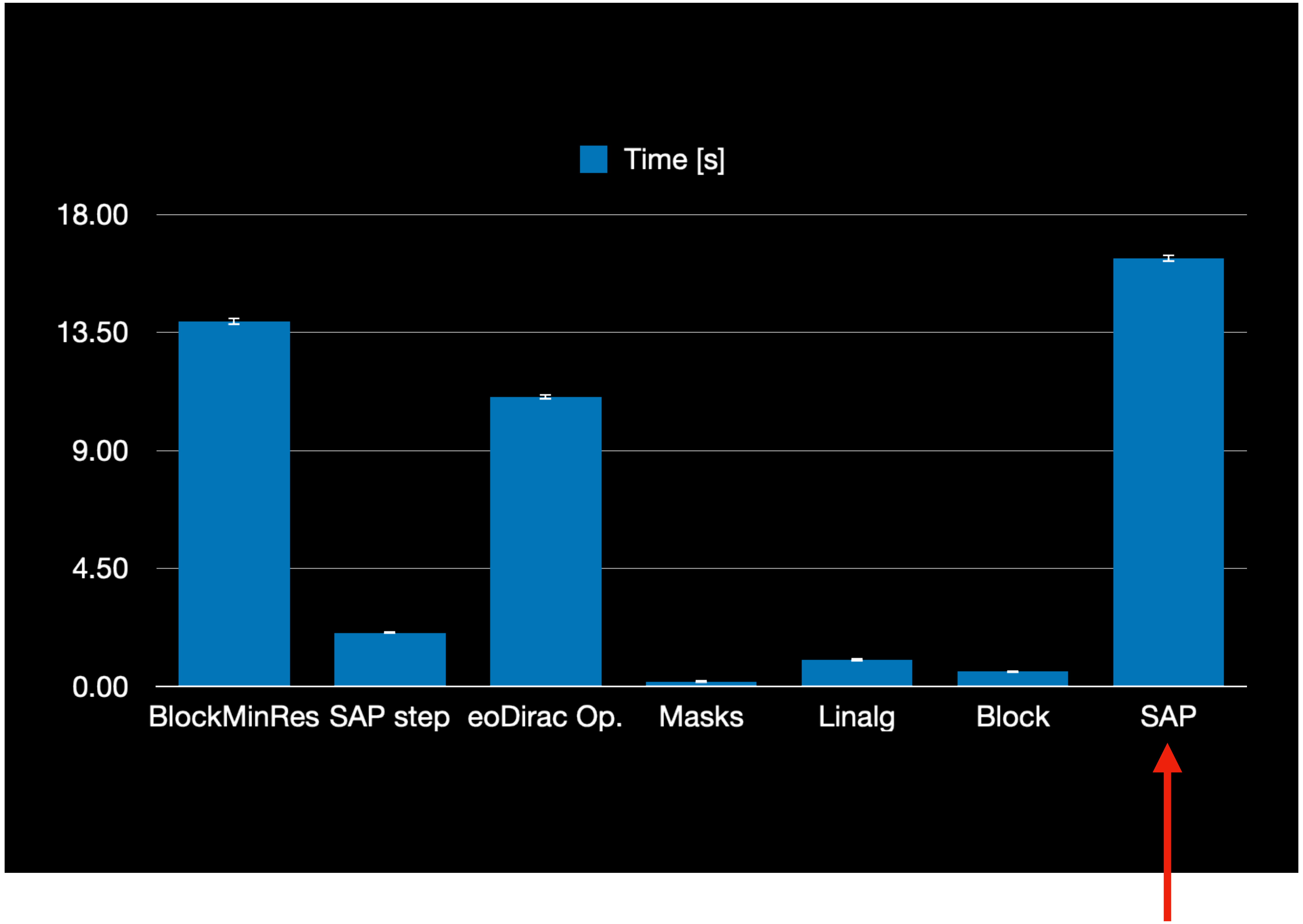
- **Performance issue:** when running BlockMinRes algorithm on even (odd) blocks the odd (even) blocks are also acted on, twice as many applications of Dirac operator per lattice point as in openQCD

→ FACTOR 2 SLOWER

Way out: block fields à la openQCD?

Results SAP on CPU

- 24x24x24x96, stabilised Wilson Clover (provided by OPEN LATtice initiative)
- 16 nodes à 24 CPUs (Tesseract)
- solve Dirac equation using SAP preconditioned FGMRES solver
- **openQCD: 2 sec**
- **Grid: 16 sec**



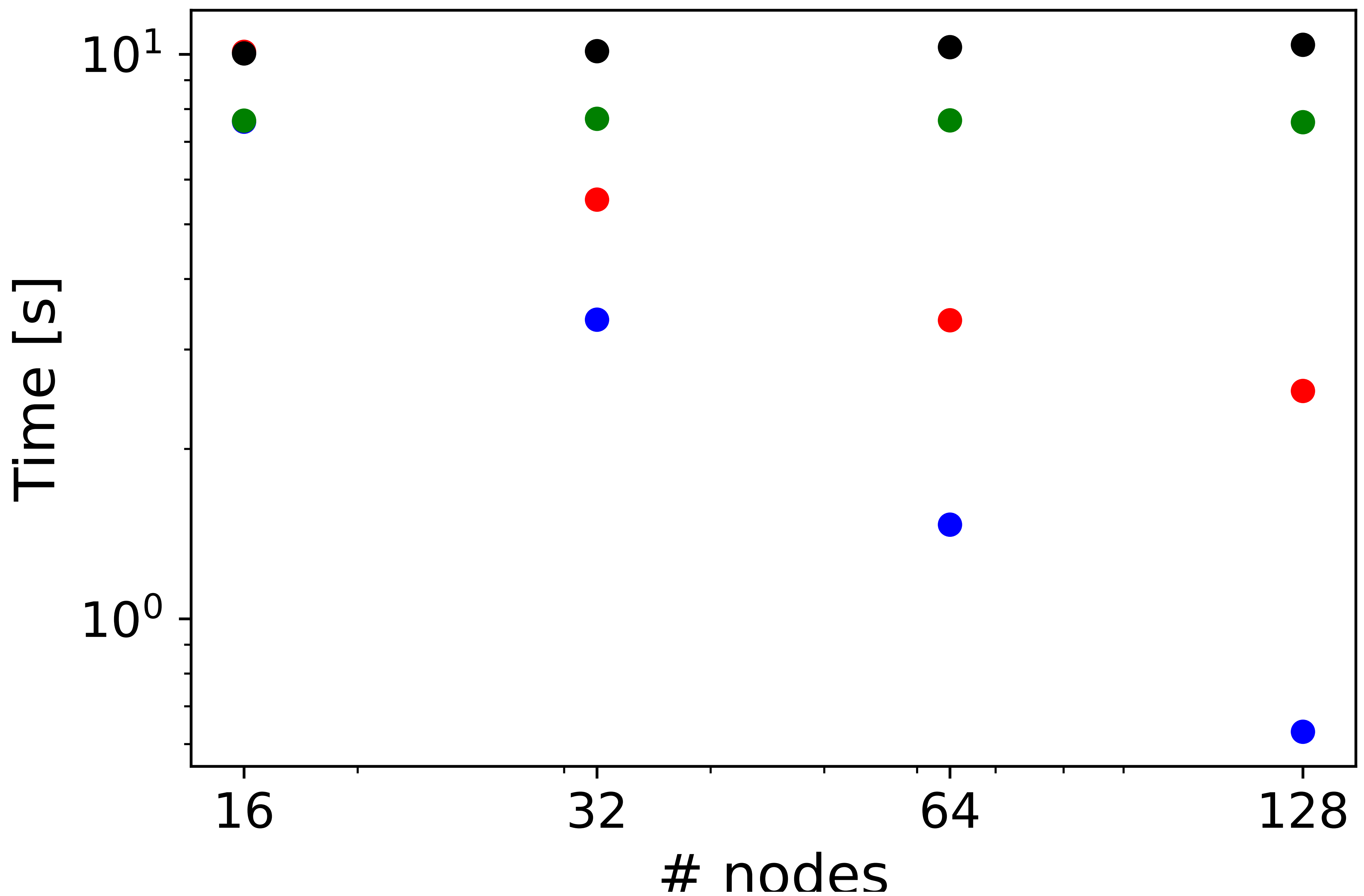
total time for preconditioner

Block Dirac Operator Improvements

comms / scaling	strong	weak
on	🔴	⚫
off	🔵	🟢

- unfair comparison as openQCD’s Dw_blk has no comms

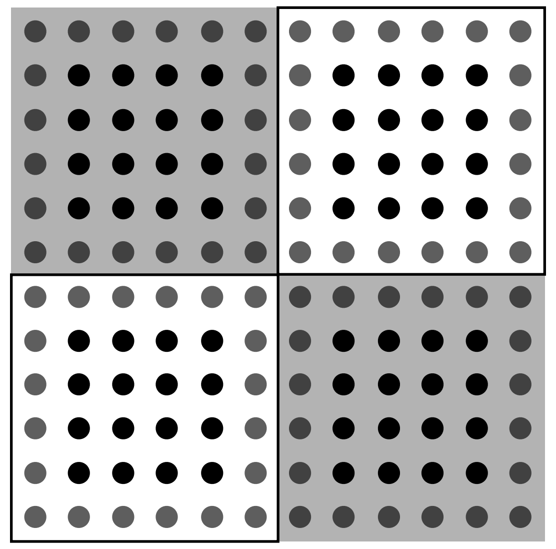
- **switch off comms** between MPI processes, here: unimproved Wilson



Timings of the Block Dirac Operator implementation in Grid

Status of DFL in Grid

Details on DFL



- Apply smoother (SAP, GMRES,…) to set of fields $\{\psi_l(x), l = 1,...,N_s\}$ to make **deflation subspace** (block projection) -> **approximate low modes** of the Dirac operator up to small deficits

$$\psi(x) = \chi(x) + \sum_{k=1}^{N} \phi_k(x)\ (A^{-1})_{kl}\ (\phi_l, \eta)$$

  - separation of modes:
    - (a) solve Dirac equation in orthogonal complement of deflation subspace (better conditioned)

$$P_L D\chi(x) = P_L \eta(x)$$

    - (b) solve little Dirac equation (using deflation again)
  - recombine full solution (a) + (b)

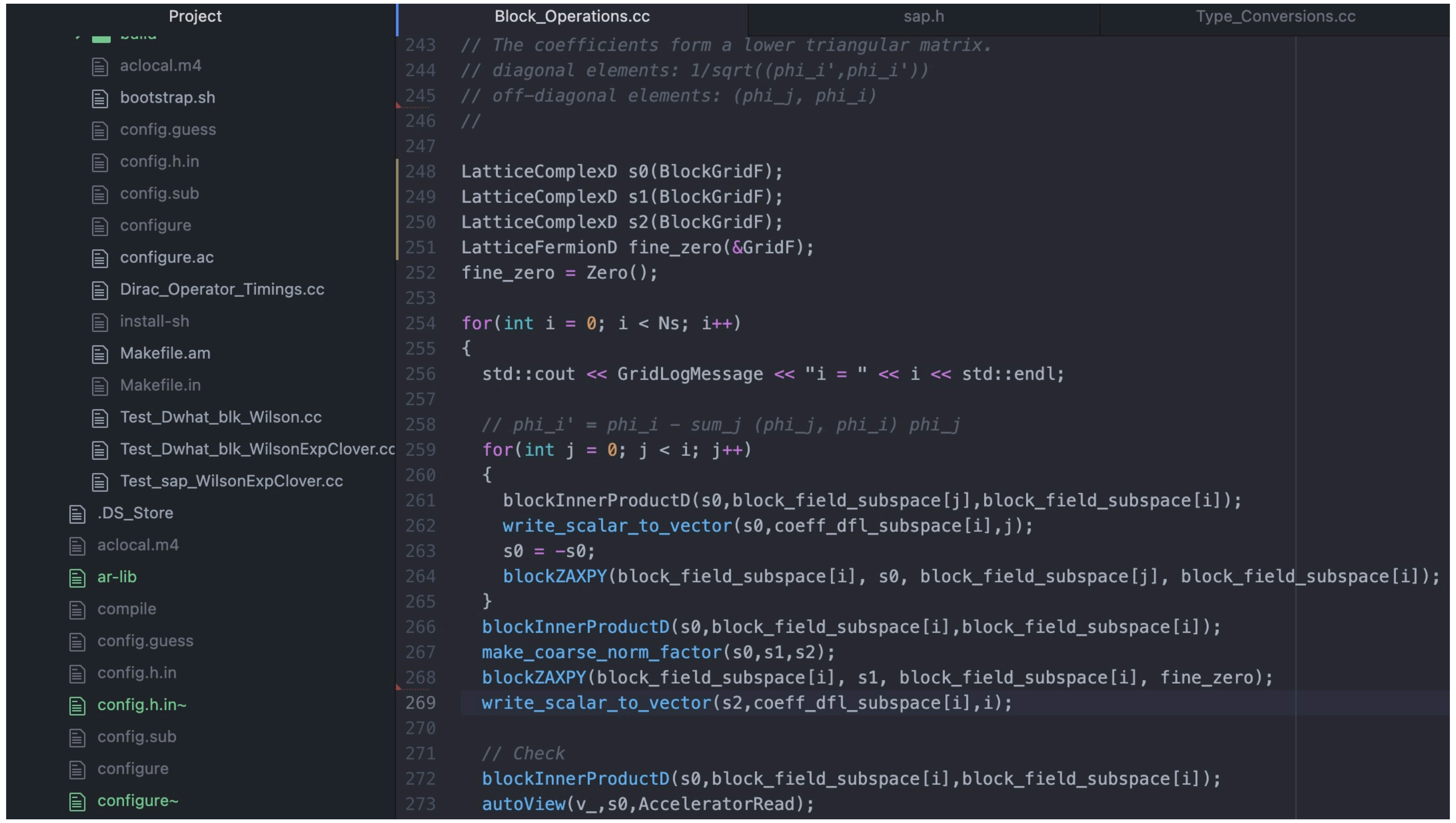
DFL in Grid

- use **CoarsendMatrix** module (P. Boyle, D. Richtmann) also used in the multigrid module to make **little Dirac matrix** $A_{ij} = (\phi_i, D\phi_j)$

- **new code developed for:**

  - vector fields for the little modes $v^{(k)}_{\Lambda,l} \quad k, l = 1,\ldots,N_s \,, \; \Lambda = 1,\ldots,N_b$ serve as **second (inner) deflation subspace** to solve $Av = w$

  - little little operator $B_{k,l} := (v^{(i)}_{\Lambda',k}, A_{(\Lambda',i),(\Lambda,j)} v^{(j)}_{\Lambda,l})$ matrix of size $N_s \times N_s$

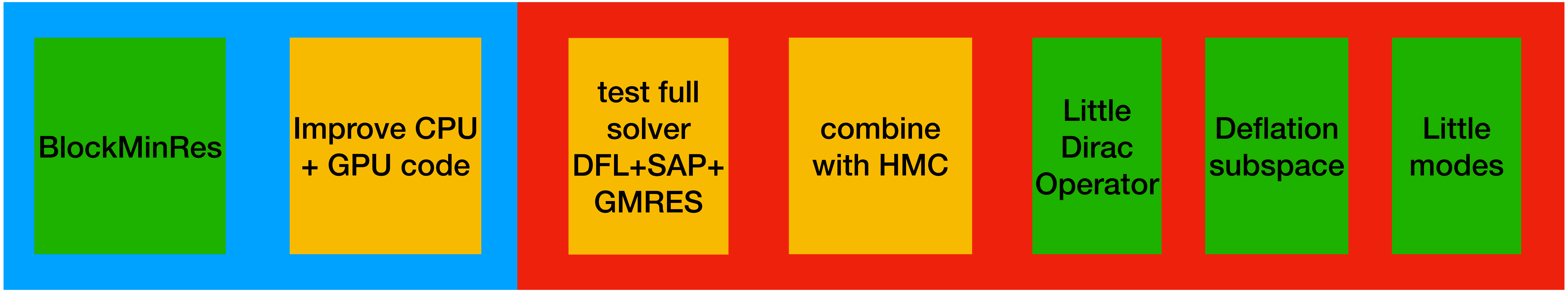


```
// The coefficients form a lower triangular matrix.
// diagonal elements: 1/sqrt((phi_i',phi_i'))
// off-diagonal elements: (phi_j, phi_i)
//

LatticeComplexD s0(BlockGridF);
LatticeComplexD s1(BlockGridF);
LatticeComplexD s2(BlockGridF);
LatticeFermionD fine_zero(&GridF);
fine_zero = Zero();

for(int i = 0; i < Ns; i++)
{
  std::cout << GridLogMessage << "i = " << i << std::endl;

  // phi_i' = phi_i - sum_j (phi_j, phi_i) phi_j
  for(int j = 0; j < i; j++)
  {
    blockInnerProductD(s0,block_field_subspace[j],block_field_subspace[i]);
    write_scalar_to_vector(s0,coeff_dfl_subspace[i],j);
    s0 = -s0;
    blockZAXPY(block_field_subspace[i], s0, block_field_subspace[j], block_field_subspace[i]);
  }
  blockInnerProductD(s0,block_field_subspace[i],block_field_subspace[i]);
  make_coarse_norm_factor(s0,s1,s2);
  blockZAXPY(block_field_subspace[i], s1, block_field_subspace[i], fine_zero);
  write_scalar_to_vector(s2,coeff_dfl_subspace[i],i);

  // Check
  blockInnerProductD(s0,block_field_subspace[i],block_field_subspace[i]);
  autoView(v_,s0,AcceleratorRead);
```

Concluding remarks

SAP

Inexact Deflation (DFL)



BlockMinRes

Improve CPU + GPU code

test full solver DFL+SAP+GMRES

combine with HMC

Little Dirac Operator

Deflation subspace

Little modes

SAP in Grid

- runs on CPU and GPU machines
- reduce comm. overhead to make SAP faster

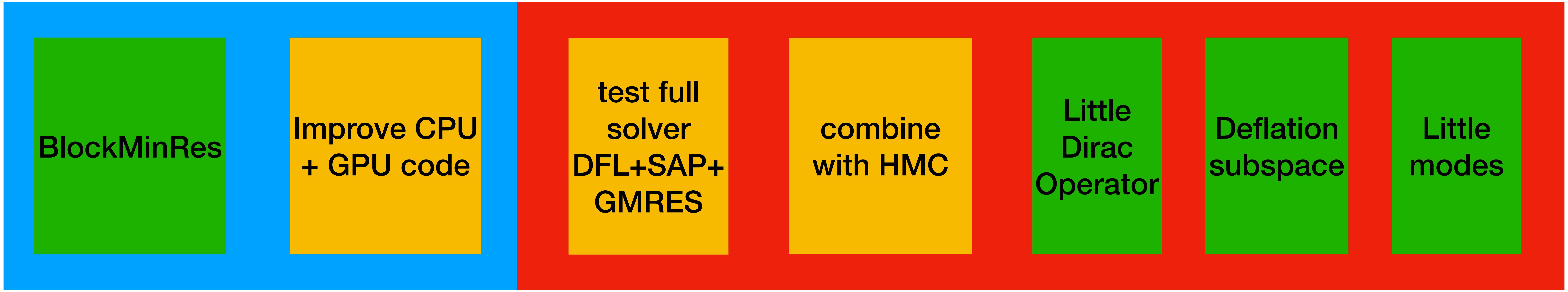
Inexact deflation in Grid

- Functions for DFL subspace projections ready
- **Next steps:** run solves (full and little system) and compare with other algorithms + study use of different smoothers on CPUs & GPUs

Concluding remarks

SAP

Inexact Deflation (DFL)



BlockMinRes

Improve CPU + GPU code

test full solver DFL+SAP+GMRES

combine with HMC

Little Dirac Operator

Deflation subspace

Little modes

SAP in Grid

- runs or
- reduce SAP fa

e projections ready

and little system)

orithms + study

on CPUs & GPUs

This is work in progress! I am happy to receive feedback and feel free to contribute! Public Grid branch will be released soon on github.

Thank you very much!