The Parallel Nonsymmetric QR Algorithm with Aggressive Early Deflation

Robert Granat[1], Bo Kågström[1], Daniel Kressner[2], and Meiyue Shao[1,2]

[1]Department of Computing Science and HPC2N, Umeå University
[2]MATHICSE, École Polytechnique Fédérale de Lausanne

Boston, February 2013

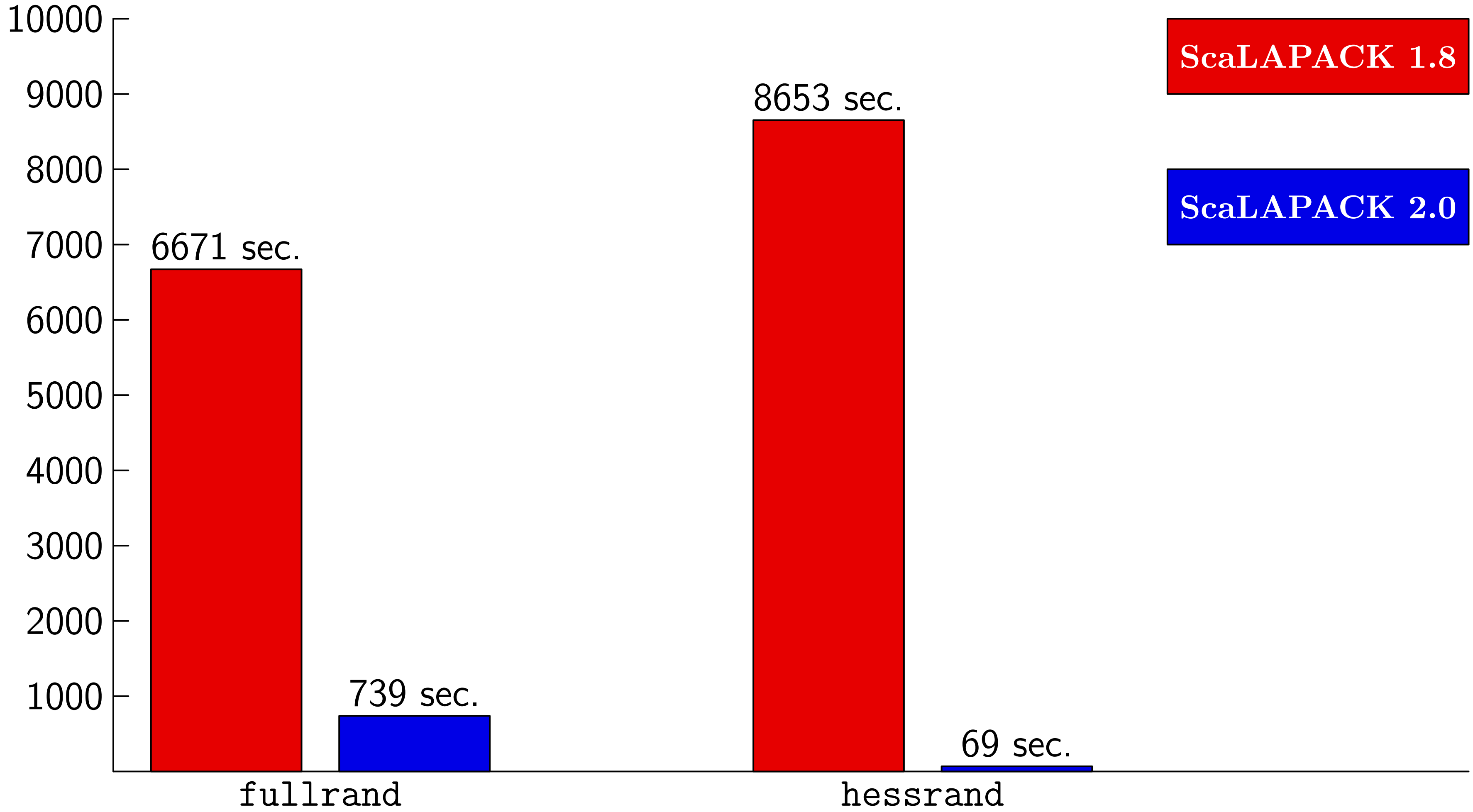
- Standard eigenvalue problem (SEP)

$$Ax = \lambda x, \qquad A \in \mathbb{C}^{N\times N}, \quad x \in \mathbb{C}^{N}, \quad x \neq 0.$$

- Schur form
  A can be factorized as
  $$A = QTQ^*,$$
  where Q is unitary ($QQ^* = Q^*Q = I$) and T is upper triangular.
  (If A is real, then Q is orthogonal and T is quasi-upper triangular.)

- Sometimes all eigenvalues of A are indeed required.
  For example, the Schur-Parlett algorithm for computing matrix functions:
  $$A = QTQ^* \quad \Rightarrow \quad f(A) = Qf(T)Q^*.$$

- How to compute all eigenvalues of A?
  Use the QR algorithm.



Overall execution time of the QR algorithm for two classes of $16,000 \times 16,000$ upper Hessenberg matrices on 4×4 processors (*akka*@HPC2N): ScaLAPACK 1.8 vs. ScaLAPACK 2.0.

- A high level abstraction of the QR algorithm:

  1. (optional) Balancing (isolating and scaling)
  2. Hessenberg reduction
  3. Repeat

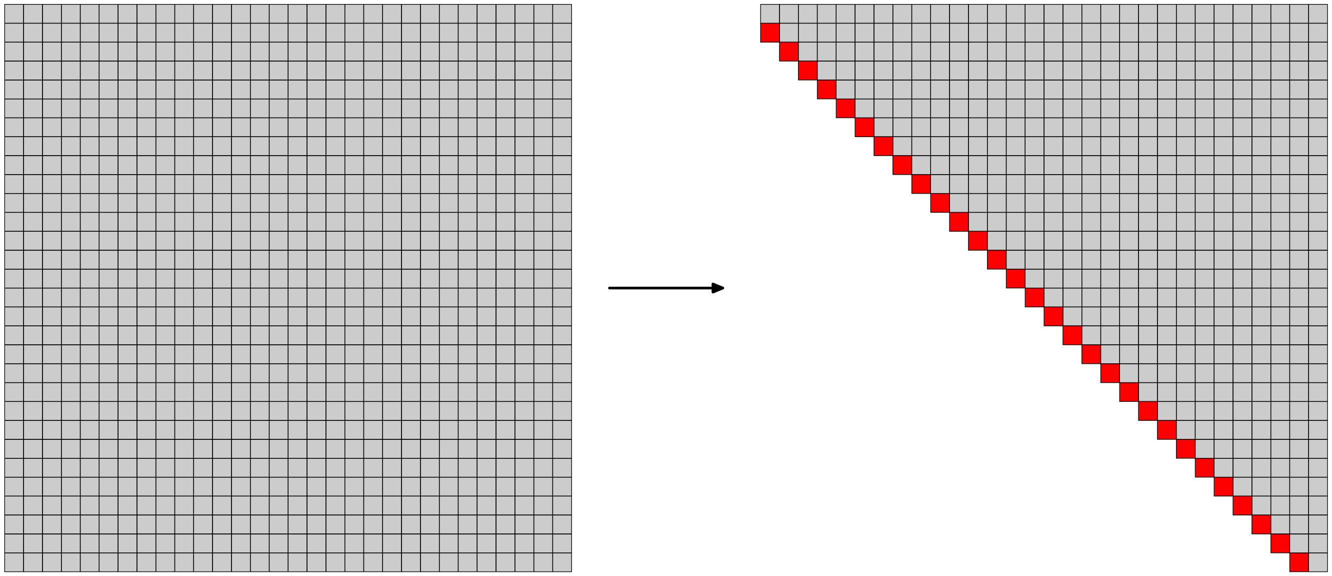
     Deflation

     QR sweep

     Until converge
  4. (optional) Eigenvalue reordering*
  5. (optional) Backward transformation

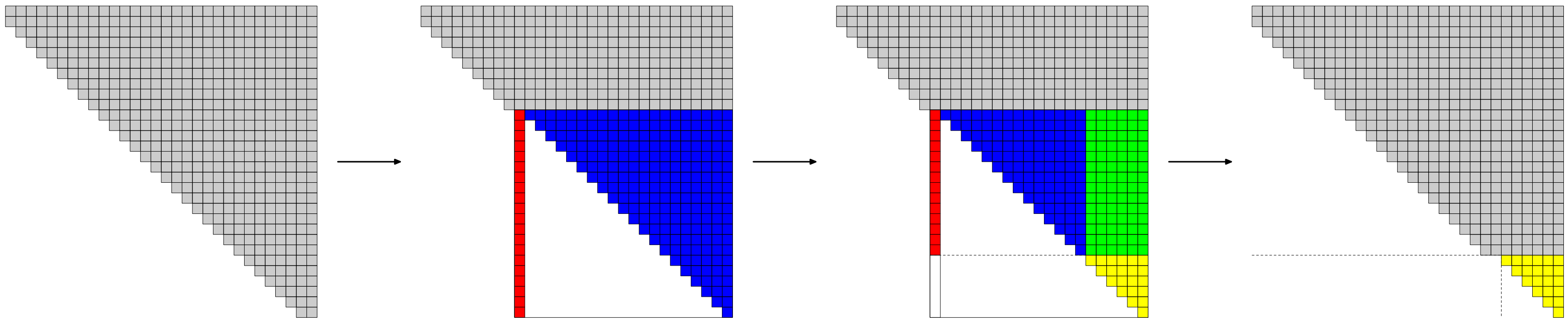
* Especially when a subspace associated with a specified set of eigenvalues is required.

- Stage 1 — Hessenberg reduction



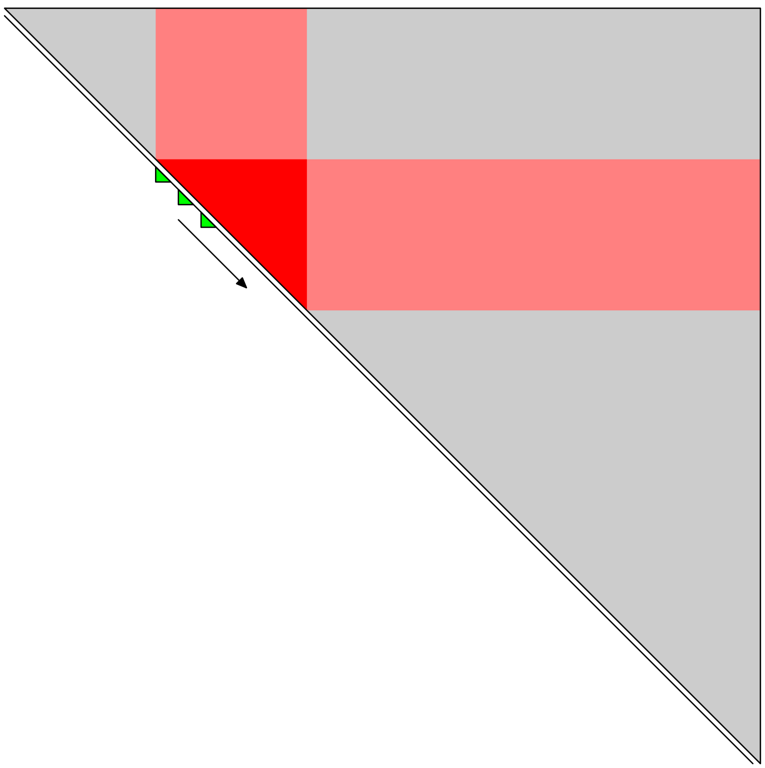
- Stage 2 — QR iteration
  - Aggressive early deflation (AED)
  - Small-bulge multishift QR sweep

- Stage 1 — Hessenberg reduction
- Stage 2 — QR iteration
  - Aggressive early deflation (AED)



  - Small-bulge multishift QR sweep

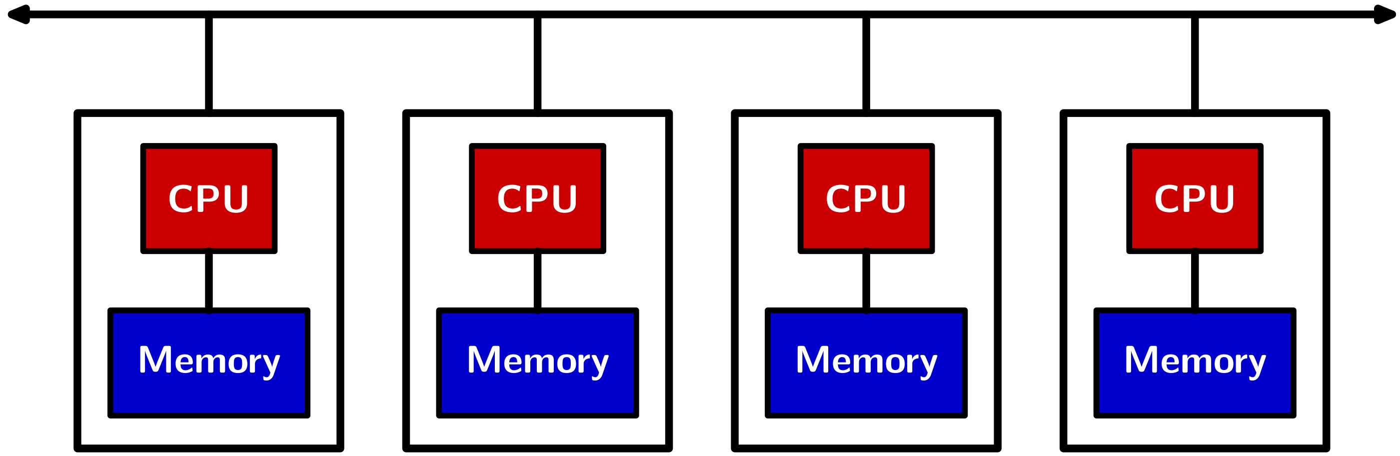
- Stage 1 — Hessenberg reduction
- Stage 2 — QR iteration
  - Aggressive early deflation (AED)
  - Small-bulge multishift QR sweep



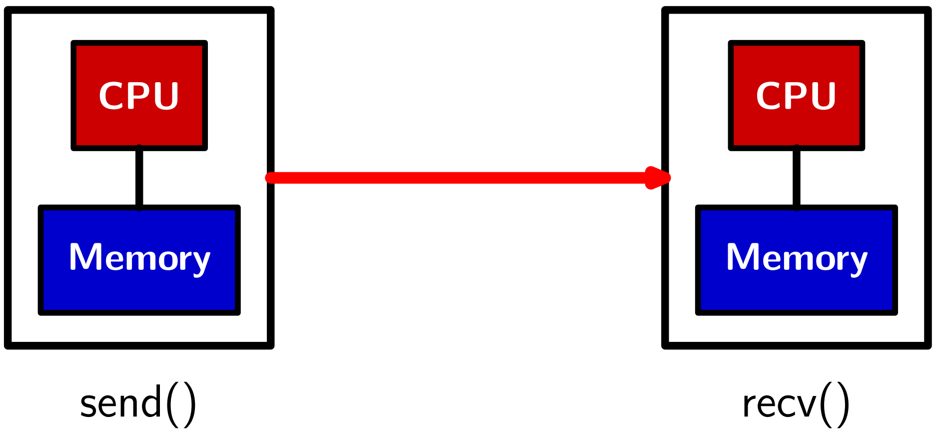
Stage	LAPACK	ScaLAPACK 2.0
0: Balancing	xGEBAL	PxGEBAL
1: Hessenberg reduction	xGEHRD	PxGEHRD
2: QR iteration	xLAHQR xHSEQR	PxLAHQR PxHSEQR
3: Eigenvalue reordering	xTRSEN	PxTRSEN PxTRORD

Our contributions

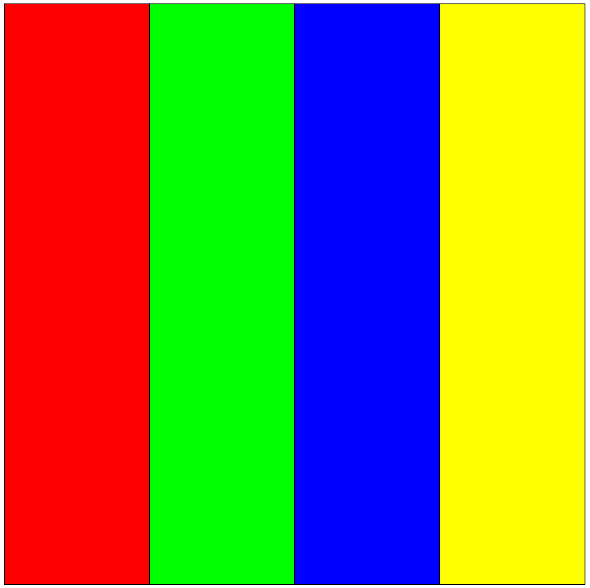
- Distributed memory systems



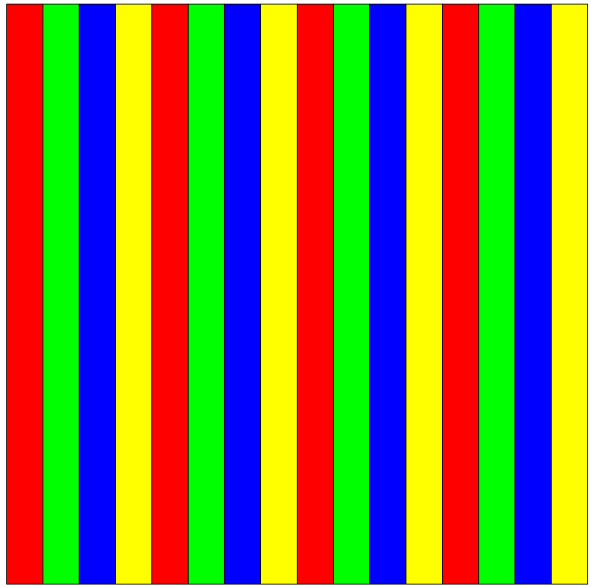
- Message passing



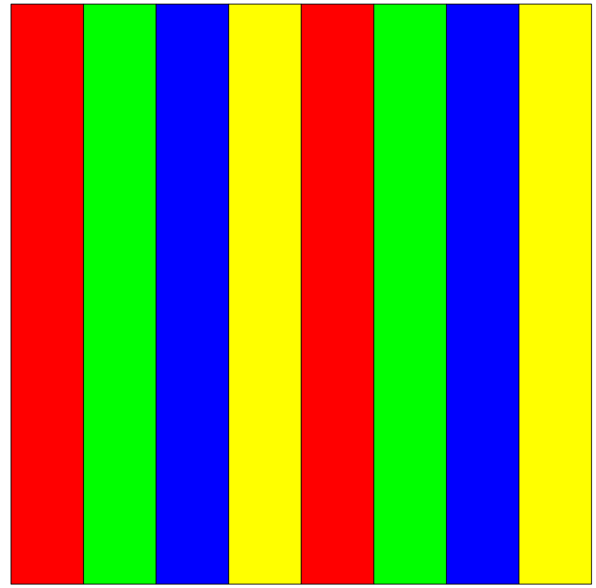
1D block



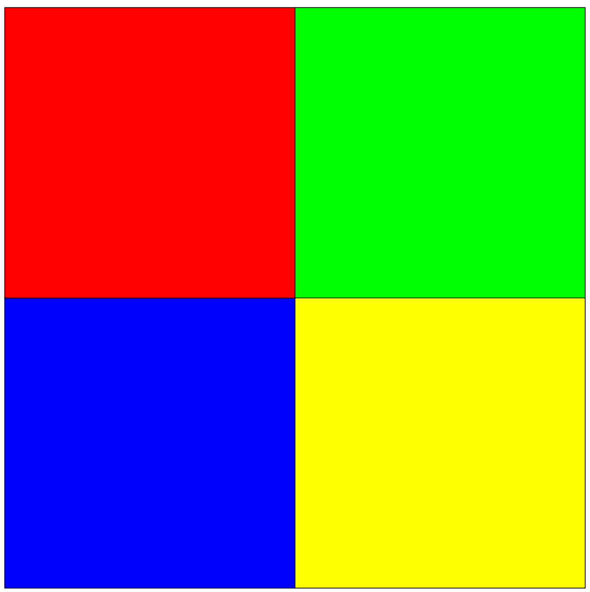
1D cyclic



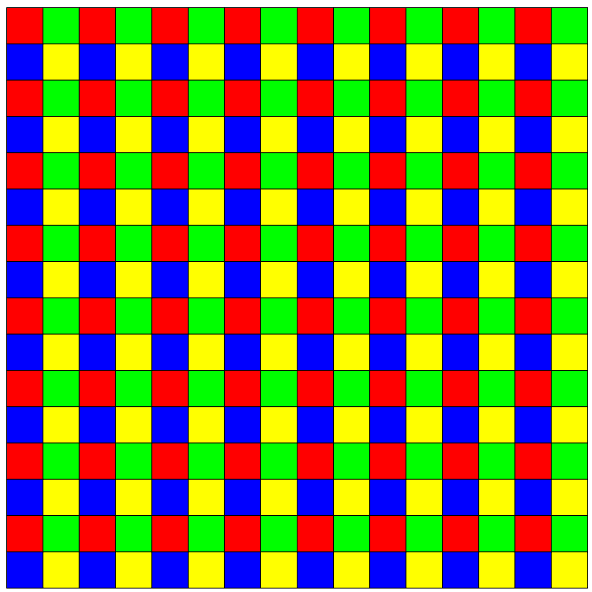
1D block cyclic



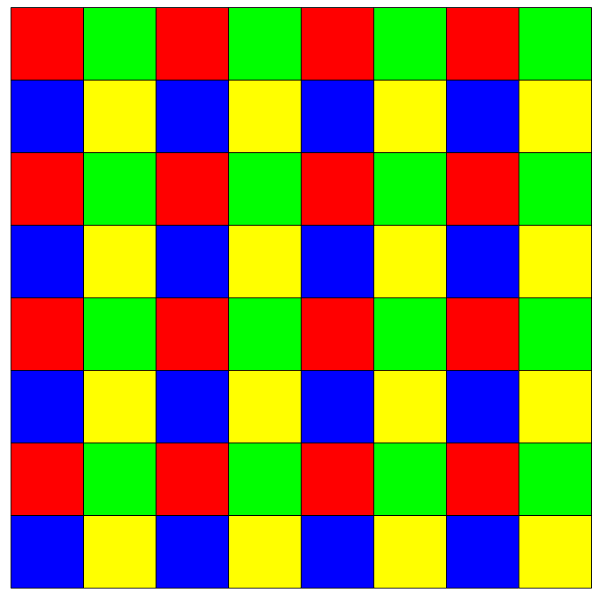
2D block



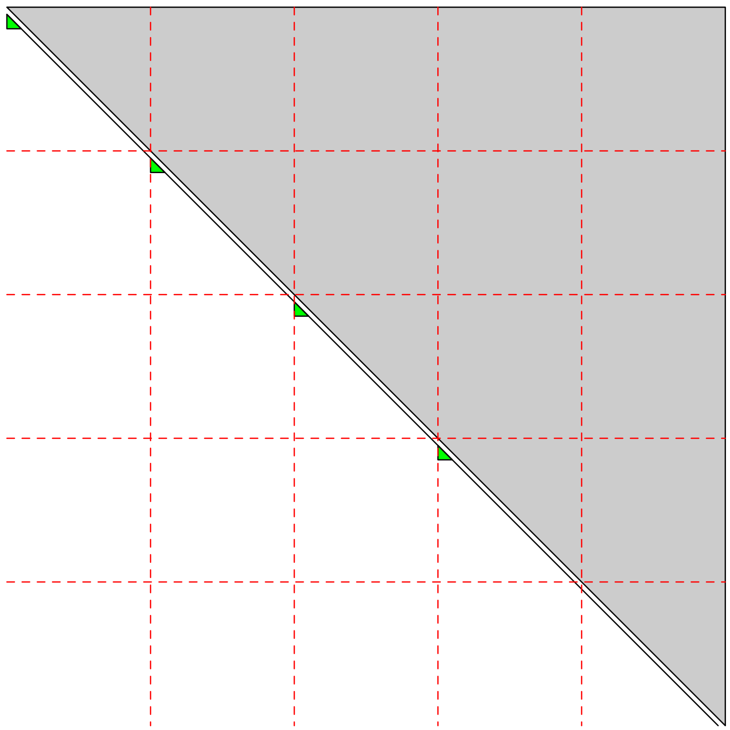
2D cyclic



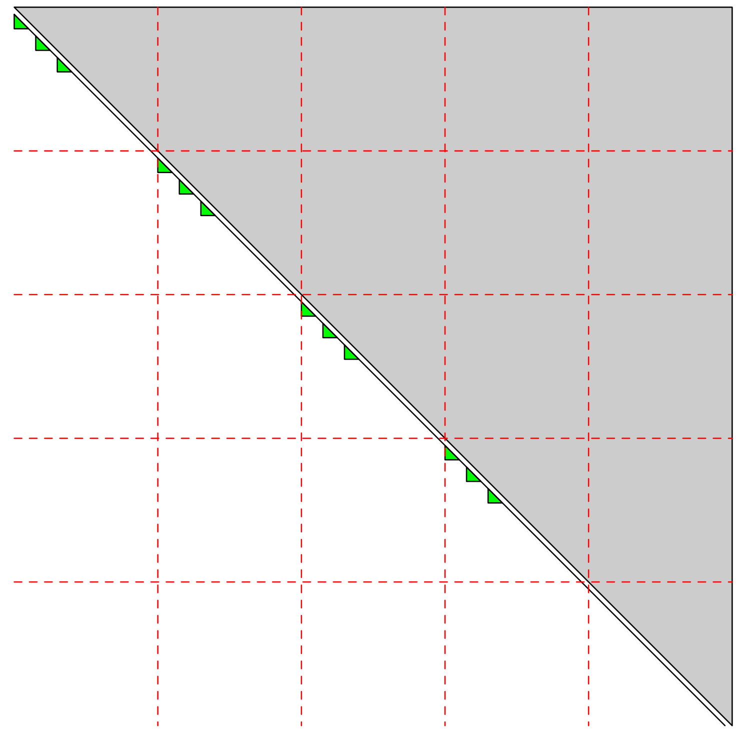
★ 2D block cyclic



- Chase multiple chains of tightly coupled bulges



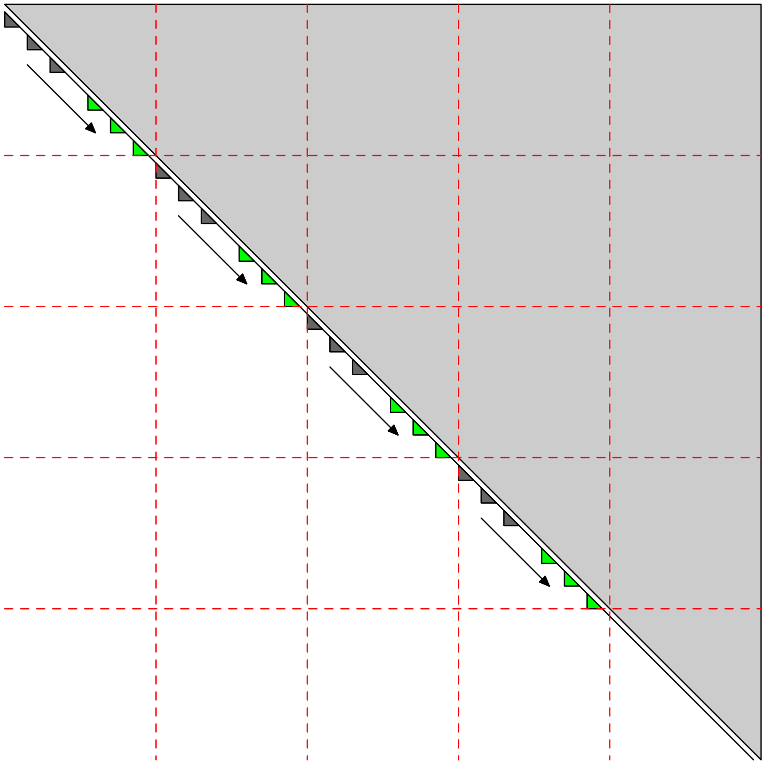
ScaLAPACK 1.8
loosely coupled bulges
for small matrices



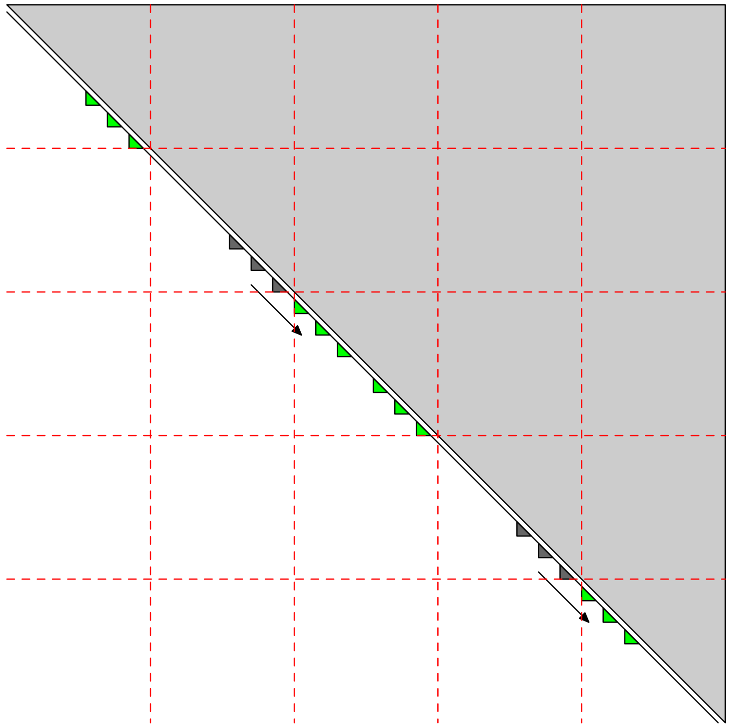
ScaLAPACK 2.0
tightly coupled bulges
for large matrices

Level 1 BLAS ☹ ⟶ Level 3 BLAS ☺

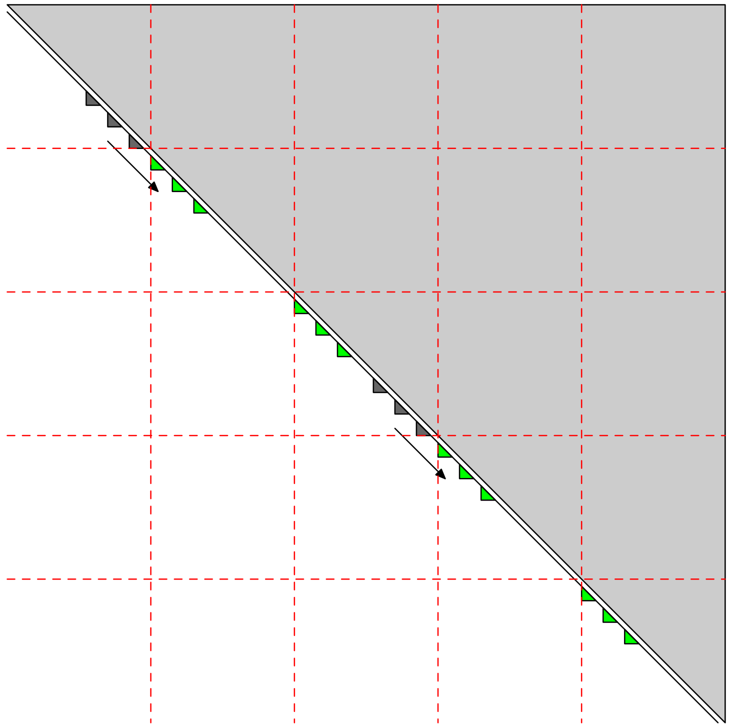
- Intrablock chase can be performed simultaneously



- Interblock chase are performed in an odd-even manner
  to avoid conflicts between different tightly coupled chains

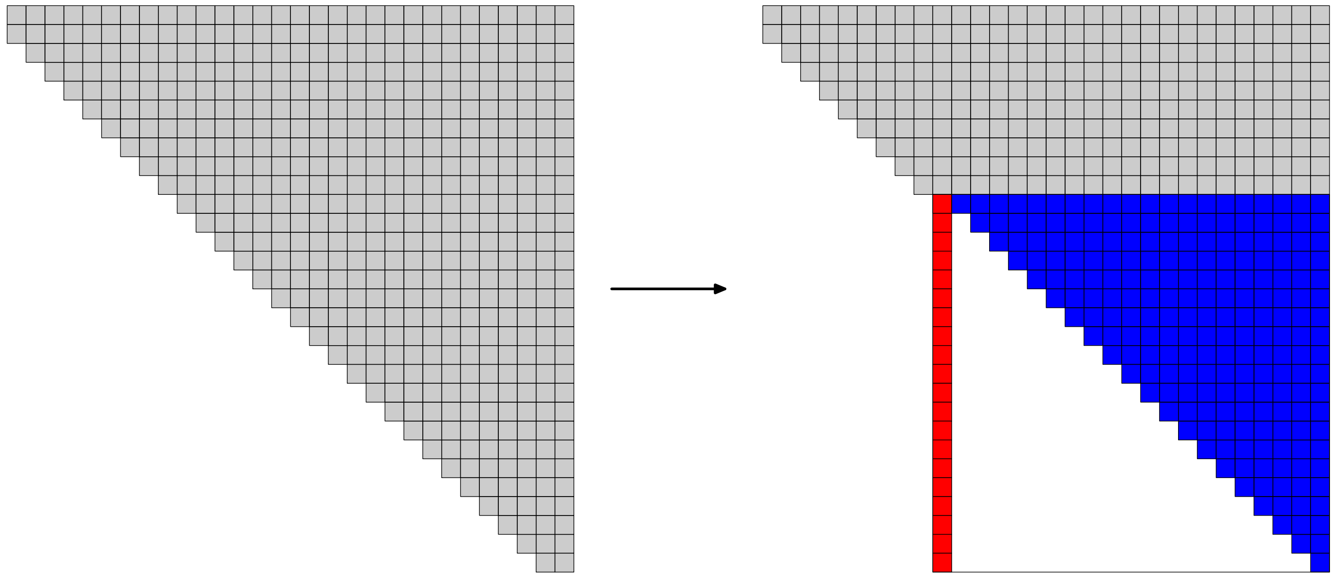


first round



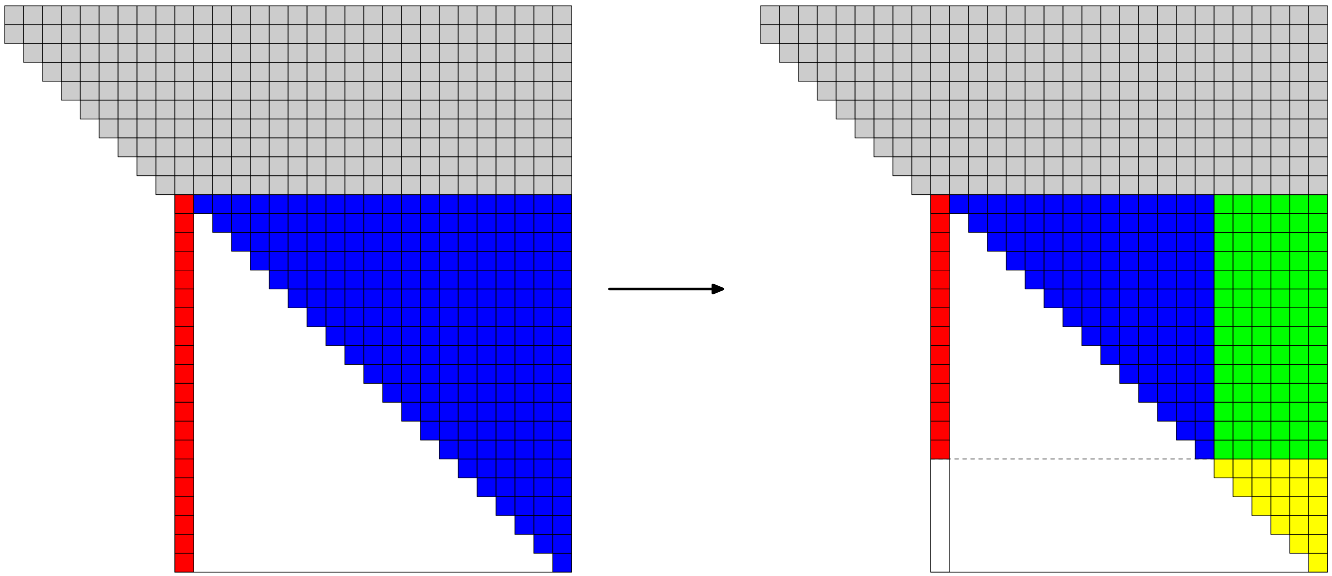
second round

- Stage 1 — Schur decomposition



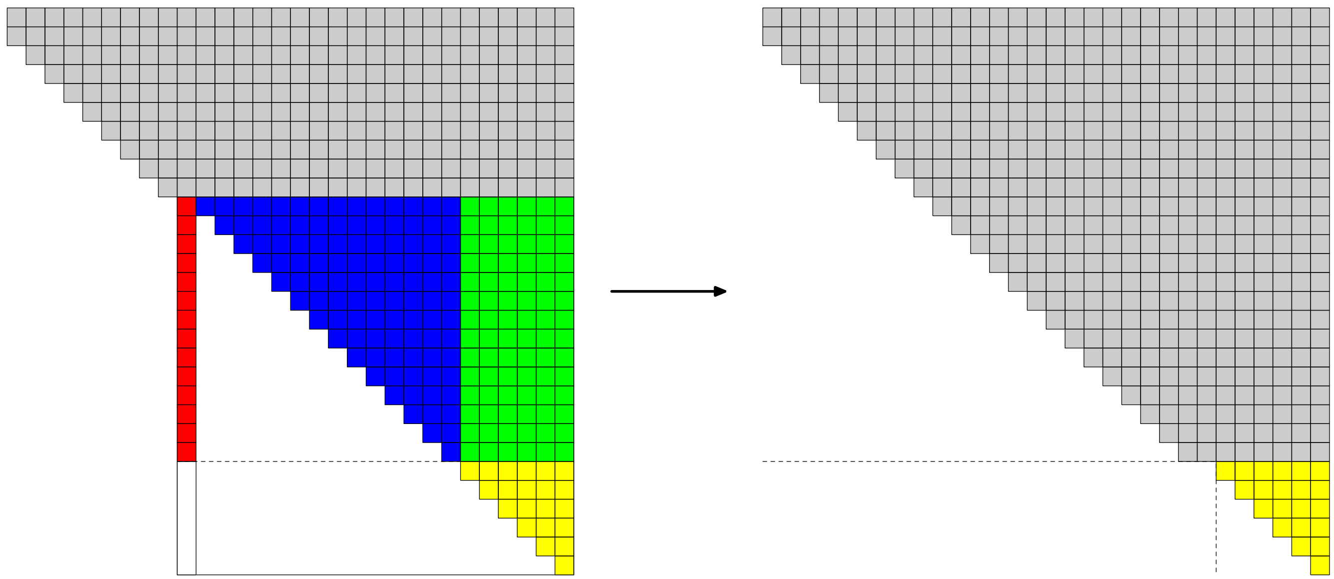
  - The Schur decomposition is computed by either the new parallel QR algorithm (recursively), or the pipelined QR algorithm + another level of AED, depends on n_{AED} and $P_r \times P_c$.
  - Reduce parallel overhead via data redistribution to a subgrid.
- Stage 2 — Eigenvalue reordering
- Stage 3 — Hessenberg reduction

- Stage 1 — Schur decomposition
- Stage 2 — Eigenvalue reordering



  - Check possible deflation at the bottom of the spike.
  - Undeflatable eigenvalues are moved to the top-left corner.
  - Reorder eigenvalues in groups to avoid frequent communication.
- Stage 3 — Hessenberg reduction

- Stage 1 — Schur decomposition
- Stage 2 — Eigenvalue reordering
- Stage 3 — Hessenberg reduction



Simply call the ScaLAPACK routine `PxGEHRD`.

- AED is mathematically efficient, but becomes a BOTTLENECK in practice

  The Schur decomposition is too expensive to calculate because of

  - frequent communication
  - heavy task dependence
  - significant overhead in the start-up and ending stages

- AED is mathematically efficient, but becomes a BOTTLENECK in practice

  The Schur decomposition is too expensive to calculate because of

  - frequent communication
  - heavy task dependence
  - significant overhead in the start-up and ending stages

- Remedy

  - Small problems — use only one processor

    Copy the AED window to one processor and call LAPACK's `xLAQR3`.
    Implemented in the modified version of ScaLAPACK's pipelined QR algorithm.

- AED is mathematically efficient, but becomes a BOTTLENECK in practice
  The Schur decomposition is too expensive to calculate because of
  – frequent communication
  – heavy task dependence
  – significant overhead in the start-up and ending stages

- Remedy
  – Small problems — use only one processor
    Copy the AED window to one processor and call LAPACK's `xLAQR3`.
    Implemented in the modified version of ScaLAPACK's pipelined QR algorithm.
  – Larger problems — use a subset of the processor grid
    Redistribute the AED window to a subset of processors and solve it in parallel.
    Implemented in the new parallel QR algorithm.

- Repeated runs with different parameters

- Taking into account both N and P
  Some crossover points are determined based on N^2/P (i.e. average memory load).

- The former computational bottleneck in AED is removed by
  - Multi-level AED
  - Data redistribution technique
  - Well tuned parameters

- Total execution time model

$$T = \#(\text{messages}) \cdot \alpha + \#(\text{data}) \cdot \beta + \#(\text{flops}) \cdot \gamma,$$

  where

  - α: communication latency
  - β: reciprocal of bandwidth
  - γ: time for one floating point operation

- Processor grid is square: $P_r = P_c = \sqrt{P}$

- Balanced load: block cyclic data distribution
  N/N_b, # block rows and columns, $\gg \sqrt{P}$

- Execution time of our parallel Hessenberg QR algorithm

$$T(N,P) = k_{\text{AED}} T_{\text{AED}} + k_{\text{QRSW}} T_{\text{QRSW}} + k_{\text{shift}} T_{\text{shift}},$$

where

  - k_{AED}: # super-iterations (AED+QRSW)
  - k_{QRSW}: # multishift QR sweeps
  - k_{shift}: # times when new shifts are computed (AED does not provide sufficiently many)

Therefore we have $k_{\text{AED}} \geq k_{\text{QRSW}} \geq k_{\text{shift}} \geq 0$.

(These numbers usually depend on the property of the matrix and the algorithmic parameter settings.)

- Under certain assumptions of the convergence rate, the execution time of our parallel Hessenberg QR algorithm is

$$T(N,P) = \Theta\Big(\frac{N^2 \log P}{\sqrt{P}\, N_b^2}\Big)\alpha + \Theta\Big(\frac{N^3}{\sqrt{P}\, N_b}\Big)\beta + \Theta\Big(\frac{N^3}{P}\Big)\gamma.$$

- The pipelined QR algorithm (in ScaLAPACK 1.8) requires

$$T(N,P) = \Theta\Big(\frac{N^2 \log P}{\sqrt{P}\, N_b}\Big)\alpha + \Theta\Big(\frac{N^2 \log P}{\sqrt{P}} + \frac{N^3}{P\, N_b}\Big)\beta + \Theta\Big(\frac{N^3}{P}\Big)\gamma.$$

- The new algorithm reduces #(messages) by a factor of $\Theta(N_b)$.

  The serial term $\Theta(N^3/P)\,\gamma$ is also improved because most operations in the new algorithm are of Level 3 computational intensity.

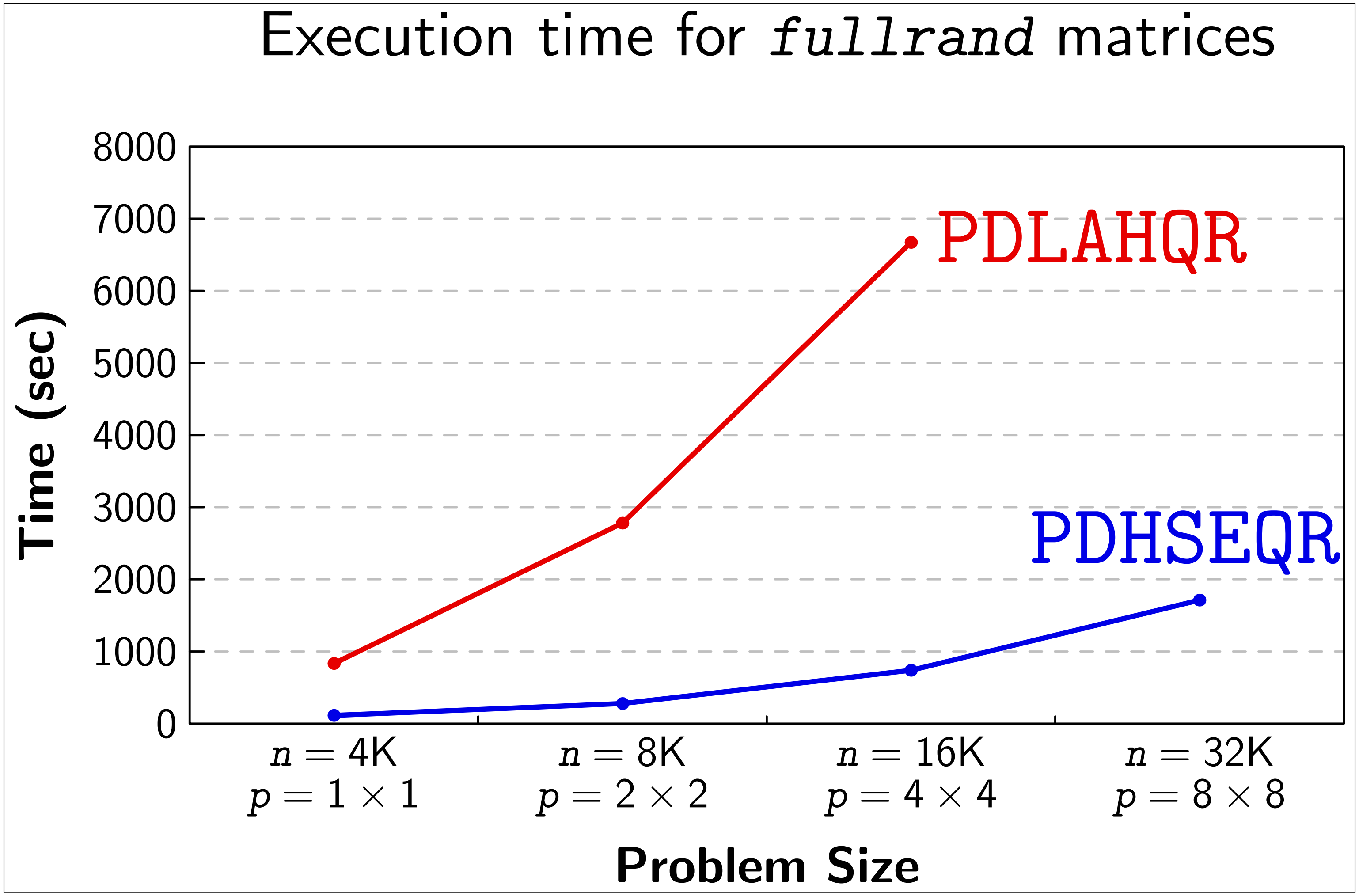
- In practice, $T(N,P) \sim N^{1.3}$ is observed when N^2/P is a constant.
  This is consistent with the theoretical model ($\Theta(N) < T(N,P) < \Theta(N^2)$).

- This research was conducted using the resources of the High Performance Computing Center North (HPC2N).

- Platform — *akka*@HPC2N

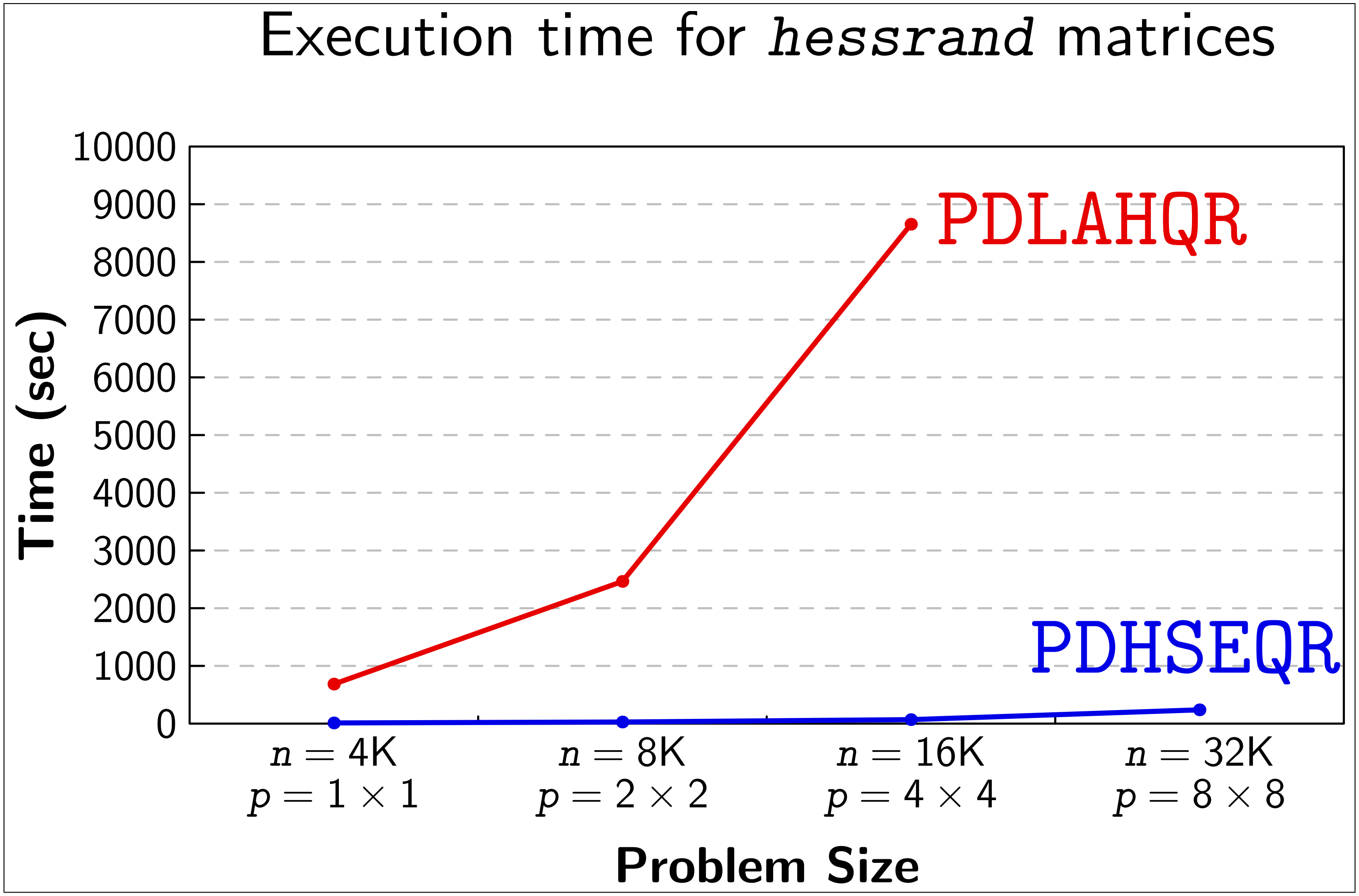
  64-bit low power Intel Xeon Linux cluster
  672 dual socket quadcore L5420 2.5GHz nodes
  256KB dedicated L1 cache, 12MB shared L2 cache
  16GB RAM per node
  Cisco Infiniband and Gigabit Ethernet, 10 GB/sec bandwidth

- Test matrices — `fullrand` (well-conditioned)



Our new routine PDHSEQR is up to $10\times$ faster than PDLAHQR.

- Test matrices — `hessrand` (ill-conditioned)



Our new routine PDHSEQR is up to 125× faster than PDLAHQR.

- A $100,000 \times 100,000$ `fullrand` matrix

# Procs	16×16	24×24	32×32
Total time	5.87 hrs	3.97 hrs	3.07 hrs
Balancing	0.24 hrs	0.24 hrs	0.24 hrs
Hess. red.	2.92 hrs	1.78 hrs	1.08 hrs
QR+AED	2.72 hrs	1.95 hrs	1.75 hrs
AED/(QR+AED)	44%	44%	42%
Shifts per eig	0.30	0.22	0.16

The preliminary version of `PDHSEQR` (Granat et al., SISC 2010) requires 7 hours for the QR iteration (using 32×32 processors).
Now the execution time is close to that for Hessenberg reduction.

- Summary
  - Chasing multiple chains of tightly coupled bulges.
  - Multiple levels AED via data redistribution.
  - A performance model is established.
  - Software published in ScaLAPACK 2.0.
  - Numerical experiments confirm the high performance.

- Summary
  - Chasing multiple chains of tightly coupled bulges.
  - Multiple levels AED via data redistribution.
  - A performance model is established.
  - Software published in ScaLAPACK 2.0.
  - Numerical experiments confirm the high performance.

Thank you for your attention!

Contact: Meiyue Shao, meiyue.shao@epfl.ch