Eigenvalue Solvers – The ELPA Project and Beyond

Bruno Lang

Bergische Universität Wuppertal

August 29, 2012

Overview

The ELPA project

The project I

ELPA

Hoch-skalierbare Eigenwert-Löser für PetaFlop-Anwendungen

Highly Scalable **E**igenso**l**vers for **P**etaFlop **A**pplications

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

The project II

Situation:

- Large-scale eigenproblems are often a computational bottleneck
  (e.g., electronic structure calculations, network analysis)
- Limited scaling of ScaLAPACK routines

Goals:

- Develop a **direct** solver with
  - improved scaling and overall performance
  - ability to compute partial eigensystems
- Provide methods for large matrices

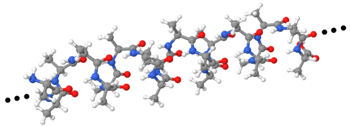
The project III

Fritz-Haber-Institut der Max-Planck-Gesellschaft	Electronic structure computations
Max-Planck-Institut für Mathematik in den Naturwissenschaften	Network analysis
BERGISCHE UNIVERSITÄT WUPPERTAL	Algorithmic development
TECHNISCHE UNIVERSITÄT MÜNCHEN	Parallelization
rzg RECHENZENTRUM GARCHING	Optimization, project coordination
IBM	State-of-the-art hardware and tools

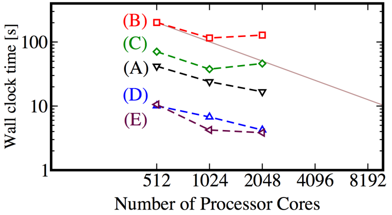
Algorithmic paths for eigenproblems I

Standard approach for solving generalized EPs $Hc = \epsilon Sc$:

(A) Reduce to standard EP (Cholesky decomp) $\rightsquigarrow Aq = \lambda q$
(B) Tridiagonalize A (Householder reflections) $\rightsquigarrow T$
(C) Solve tridiagonal EP (e.g., divide and conquer) $\rightsquigarrow \lambda, q_T$
(D) (Orthogonal) Back transformation of k eigenvectors $\rightsquigarrow q_A$
(E) (Non-orthogonal) Back transformation $\rightsquigarrow c$



Segment of α-helical Polyalanine molecule Ala$_{100}$
$n = 27069$, $k \approx 13\%$



ScaLAPACK performance on BG/P

Algorithmic paths for eigenproblems II

Standard approach for standard symmetric EPs:

$$A \xrightarrow[\text{tridiagonal form}]{\text{reduction to}} T \xrightarrow[\text{–vectors of T}]{\text{compute eigenvalues and}} (\lambda, q_T) \xrightarrow[\text{eigenvectors}]{\text{transform}} q_A$$

QR

BisInvIt

D & C

MRRR

Algorithmic paths for eigenproblems III

Problems with this approach:

$$A \xrightarrow{\text{reduction to tridiagonal form}} T \xrightarrow{\text{compute eigenvalues and –vectors of } T} (\lambda, q_T) \xrightarrow{\text{transform eigenvectors}} q_A$$

reduction to tridiagonal form:

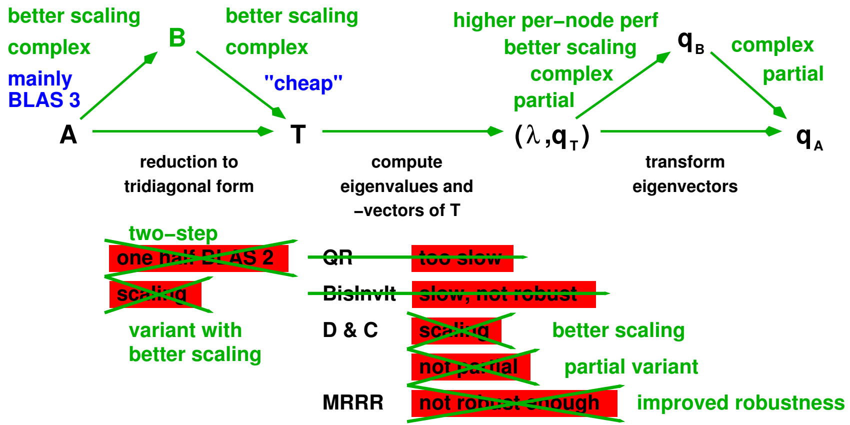
- one half BLAS 2
- scaling

compute eigenvalues and –vectors of T:

QR	too slow
BisInvIt	slow, not robust
D & C	scaling
	not partial
MRRR	not robust enough

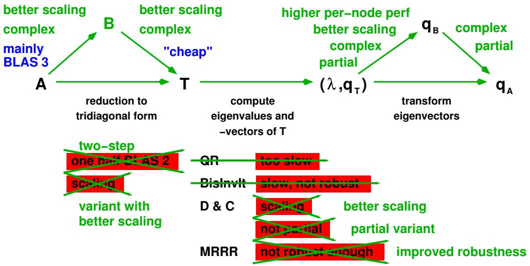
Algorithmic paths for eigenproblems IV

Improvements within ELPA:



Improvements with ELPA I

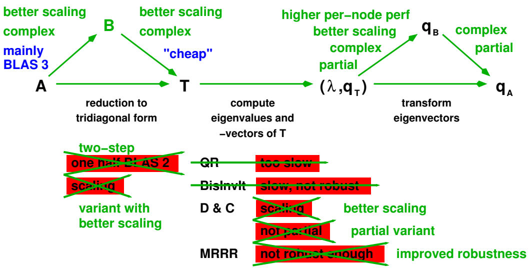
Optimized one-step tridiagonalization:



- \+ Substantially streamlined to reduce overhead
- \+ Improved memory accesses

Improvements with ELPA II

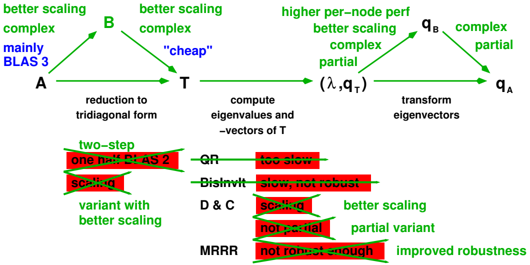
Optimized D & C:



- + Improved parallelization approach
- + Partial eigensystems at reduced cost
- + Streamlined

Improvements with ELPA III

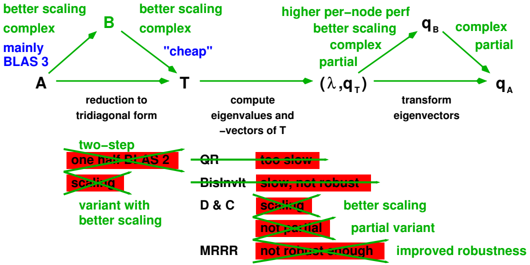
Optimized one-step back transformation:



+ Substantially streamlined

Improvements with ELPA IV

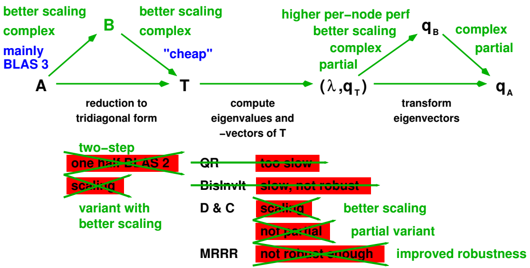
Two-step reduction I: full → banded:



+ Extended to complex
+ Optimized data distribution

Improvements with ELPA V

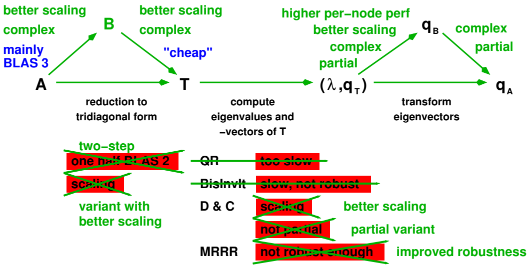
Two-step reduction II: banded → tridiagonal:



+ Extended to complex
+ Improved parallelization

Improvements with ELPA VI

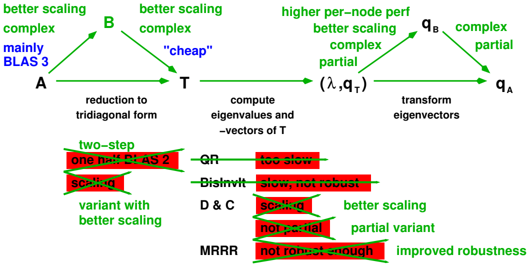
Two-step back transformation I: tridiagonal → banded:



- \+ Extended to complex and to partial eigensystems
- \+ Variants with 1D and 2D data distribution
- \+ Optimized kernels instead of WY for higher performance

Improvements with ELPA VII

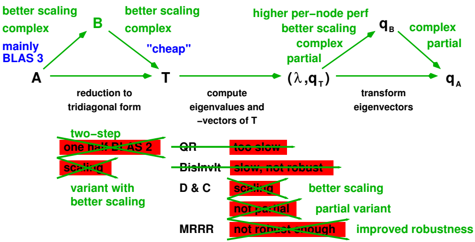
Two-step back transformation II: banded → full:



+ Extended to complex and to partial eigensystems
+ Substantially streamlined

Improvements with ELPA VIII

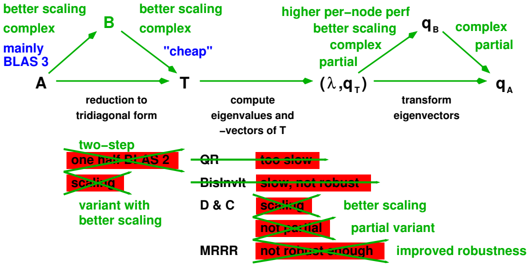
MRRR:



- \+ Much better understanding of the algorithm
- \+ Improved robustness with new representations and block decompositions
- \+ Often improved performance with optimized bisection strategy, etc.

Improvements with ELPA IX

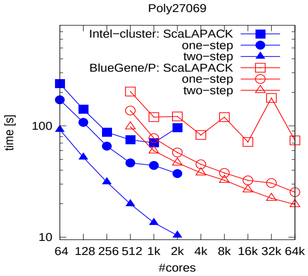
Hybrid D & C / MRRR:



+ Replace the lowest D & C recursion levels

Improvements with ELPA X

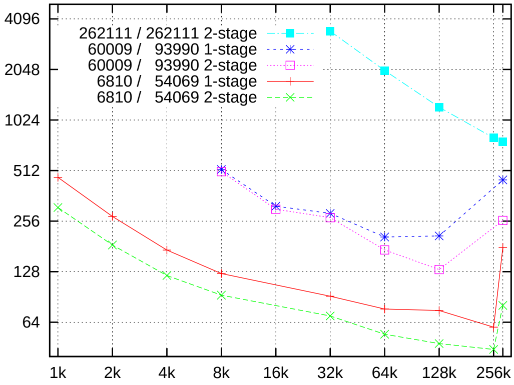
Overall performance improvement:



ScaLAPACK	`pdsyevd`: One-step reduction/back transform + (slightly) improved D & C
One-step	ELPA one-step red/back transform + ELPA D & C
Two-step	ELPA two-step red/back transform + ELPA D & C

Improvements with ELPA XI

Scaling to very large numbers of cores:



Improvements with ELPA XII

Status of the methods:

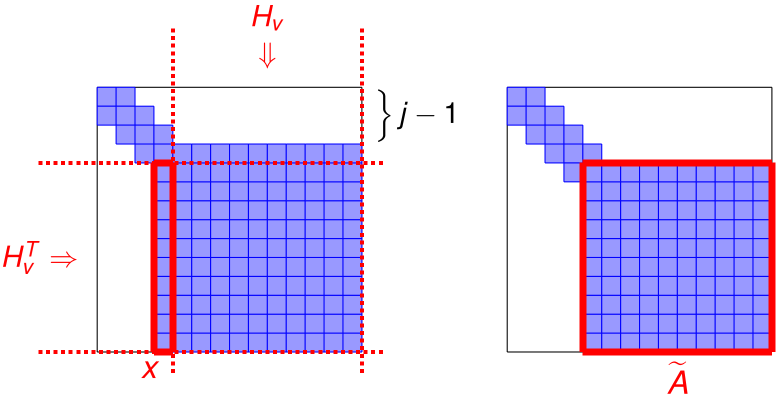
	parallel	ELPA-Lib	FHI-aims
ELPA one-step red + back transform		×	auto
reduction full–banded + back		×	auto
reduction banded–tridiag + back		×	auto
ELPA D&C partial		×	×
MRRR	×		

More information and software:

http://elpa.rzg.mpg.de/

Efficient tridiagonalization I

One-step reduction, step j (i)



A_{j-1}

$A_j = H_j^T A_{j-1} H_j$

Efficient tridiagonalization II

One-step reduction, step j (ii)

$$\widetilde{A} := H_v^T \cdot \widetilde{A} \cdot H_v = (I - v\delta v^T)^T \cdot \widetilde{A} \cdot (I - v\delta v^T) = \widetilde{A} - vw^T - wv^T$$

with

$$w = z - \frac{1}{2} v \underbrace{\delta v^T z}_{\in \mathbb{R}}, \quad z = \widetilde{A} v\delta \,.$$

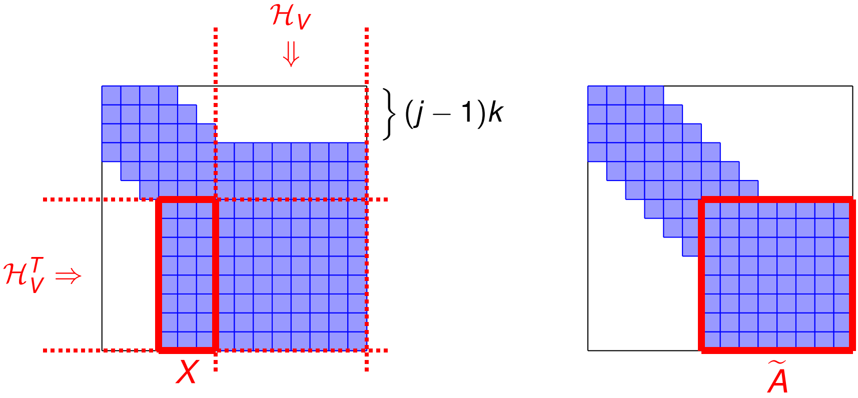
Therefore

1. Determine v and δ
2. Compute $z := \widetilde{A}v\delta$
3. Compute $w := z - \frac{1}{2}v\delta v^T z$
4. Replace $\widetilde{A}$ with $\widetilde{A} - vw^T - wv^T$ (can be blocked)

Reduces 1 column/row at a time, 50% BLAS 3, 50% **BLAS 2**.

Efficient tridiagonalization III

Reduction to banded form, step j (i)



A_{j-1}

$A_j = \mathcal{H}_V^T A_{j-1} \mathcal{H}_V$

Efficient tridiagonalization IV

Reduction to banded form, step j (ii)

$$\widetilde{A} := \mathcal{H}_V^T \cdot \widetilde{A} \cdot \mathcal{H}_V = (I - V\Delta V^T)^T \cdot \widetilde{A} \cdot (I - V\Delta V^T) = \widetilde{A} - VW^T - WV^T$$

with

$$W = Z - \frac{1}{2} V \underbrace{\Delta^T V^T Z}_{\in \mathbb{R}^{k \times k}}, \quad Z = \widetilde{A} V \Delta .$$

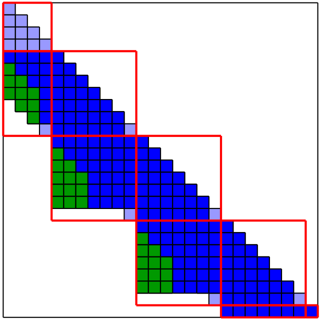
Therefore

1. Determine V and Δ (QR decomp of j-th block column)
2. Compute $Z := \widetilde{A} V \Delta$
3. Compute $W := Z - \frac{1}{2} V \Delta^T V^T Z$
4. Replace $\widetilde{A}$ with $\widetilde{A} - VW^T - WV^T$

Reduces k columns/row at a time, almost completely BLAS 3.

Beyond the basic ELPA-Lib

Blocked reduction of banded matrices



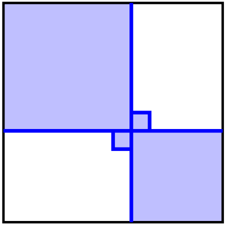
- Reduction $b_1 \to b_2$ eliminates $n_b \leq b_2$ columns per sweep
- Allows using BLAS 3 $\Rightarrow$ much faster than direct tridiagonalization

Out-of-core reduction

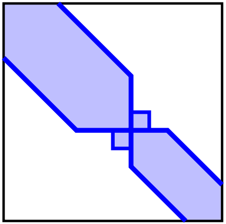
- Allows reducing of matrices of size $n \sim 100$ k on standard workstations (4 GB main memory),
- $n \sim 500$ k on workgroup shared-memory server.
- Performance at least competitive with in-core LAPACK, but complexity remains $\mathcal{O}(n^3)$.
- Similar technique also available for banded matrices ($n \sim 2$ M on workgroup server), complexity $\mathcal{O}(n^2 b)$.

Split reduction

Substantial savings for **sparse** matrices if they can be reordered as



or



Iterative solvers

- Under investigation
- Not yet [?] competitive in the situations considered