Can Block Iterative Eigensolvers Fulfill their Performance Promise?

Jonas Thies Melven Röhrig-Zöllner Achim Basermann

DLR Simulation and Software Technology
Distributed Systems and Component Software
High Performance Computing Group

project ESSEX

DLR

Motivation

Mathematical problem

- Find $\mathcal{O}(10)$ eigenpairs

$$Ax_i = \lambda_i x_i$$

of a large, sparse matrix A.

- Interior or extreme λ_i,
- symmetric or general A.

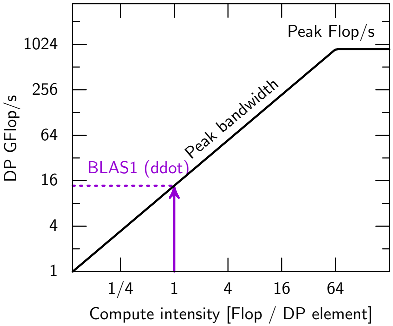
Memory gap

- Small memory bandwidth vs. high Peak Flop rate

$\rightarrow$ Increase the compute intensity

Roofline performance model

(2x 12 core Haswell EP)



Regular vs. block JDQR

1: start from initial spaces
$V^H V = I, V^A := AV, H = V^H AV$
2: **while** ... **do**
3: (sorted) Schur decomp.,
$H = STS^H, S^H S = I$
4: select shift $\theta = T_{1,1}$
5: $u = Vs_1, u^A = V^A s_1$
6: $r = u^A - \theta u$
7: *(lock converged eigenpairs in* Q*)*
8: *(shrink subspaces if necessary)*
9: $\tilde{Q} = [Q \;\; u]$
solve approximately for $t \perp \tilde{Q}$:
10: $(I - \tilde{Q}\tilde{Q}^H)(A - \theta I)(I - \tilde{Q}\tilde{Q}^H)t = -r$
11: orthogonalize t against V
12: extend $V = [V \tilde{t}]$, V^A, $H = V^H V^A$
13: **end while**

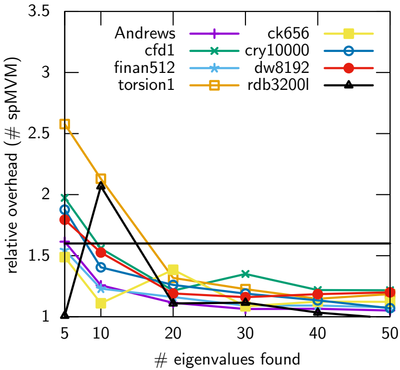
Regular vs. block JDQR

start from initial spaces
$V^H V = I, V^A := AV, H = V^H AV$
while ... **do**
    (sorted) Schur decomp.,
$H = STS^H, S^H S = I$
    select shift $\theta = T_{1,1}$
    $u = Vs_1, u^A = V^A s_1$
    $r = u^A - \theta u$
    (lock converged eigenpairs in Q)
    (shrink subspaces if necessary)
    $\tilde{Q} = [Q \;\; u]$
solve approximately for $t \perp \tilde{Q}$:
    $(I - \tilde{Q}\tilde{Q}^H)(A - \theta I)(I - \tilde{Q}\tilde{Q}^H)t = -r$
    orthogonalize t against V
    extend $V = [V \tilde{t}]$, V^A, $H = V^H V^A$
end while

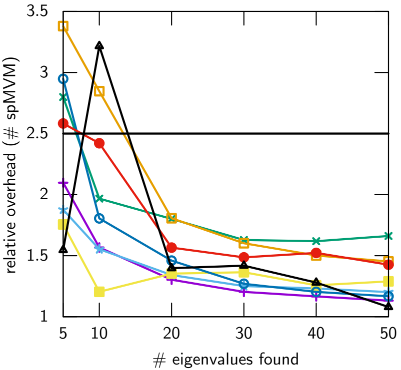
(...)
(...)
while ... **do**
    (...)
    (...)
    select shifts $\theta_i = T_{i,i}, i = 1 \dots n_b$
    $u = VS_{:,1:n_b}, u^A = V^A S_{:,1:n_b}$
    $r_{:,i} = u^A_{:,i} - \theta_i u_{:,i}$
    (...)
    (...)
    $\tilde{Q} = [Q \;\; u]$
    solve n_b independent systems
$(I - \tilde{Q}\tilde{Q}^H)(A - \theta_i I)(I - \tilde{Q}\tilde{Q}^H)t_{:,i} = -r_{:,i}$
    block-orthogonalize t against V,
    extend subspaces (by n_b vecs)
end while

Numerical behavior

Block size 2



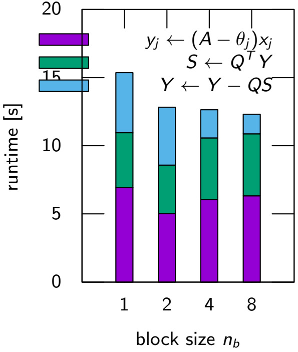
Block size 4



Kernels needed for BJDQR

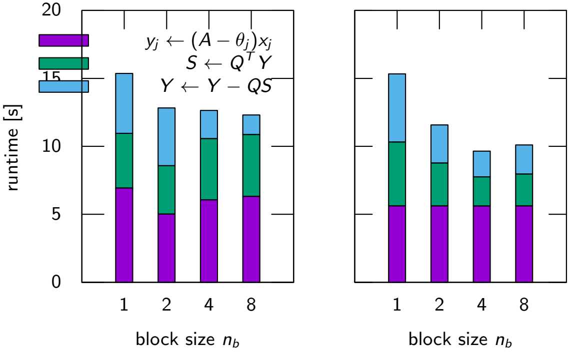
- spMMVM: sparse Matrix times Multiple Vector Multiplication
- block projection, $y \leftarrow (I - VV^H)x$
- block vector orthogonalization
  given $V, W : V^H V = I$, find $Q, R : W = QR, Q^H Q = I, V^H Q = 0.$
- subspace transformation for 'thick restart'
  $V_{:,1:k} \leftarrow V_{:,1:m} \cdot S, S \in \mathbb{C}^{m \times k}, k < m$
- preconditioning operation approximating $(A - \tau I)^{-1}x$
  (so far unpreconditioned blocked Krylov $\rightarrow$ ESSEX-II)

Row major vs. column major storage



Tpetra/TBB, col major

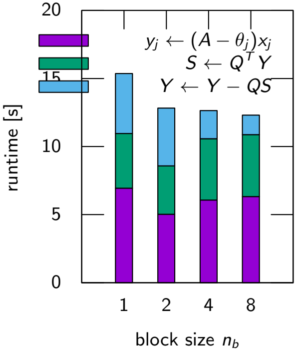
Row major vs. column major storage



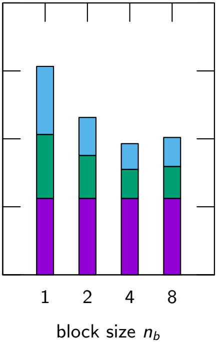
Tpetra/TBB, col major

GHOST, col major

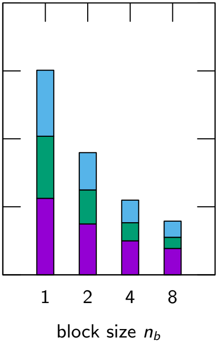
Row major vs. column major storage



Tpetra/TBB, col major

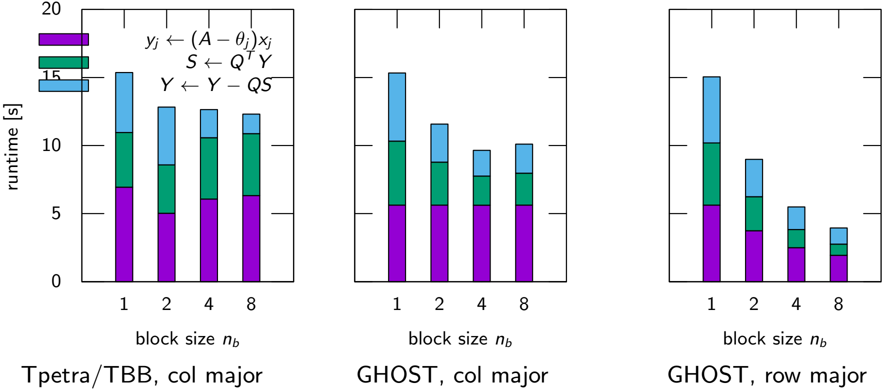


GHOST, col major



GHOST, row major

Row major vs. column major storage



- consequences for overall implementation (avoid strides!)
- SELL-C-σ most helpful if SIMD/SIMT width $> n_b$

Overall block speedup (strong scaling)

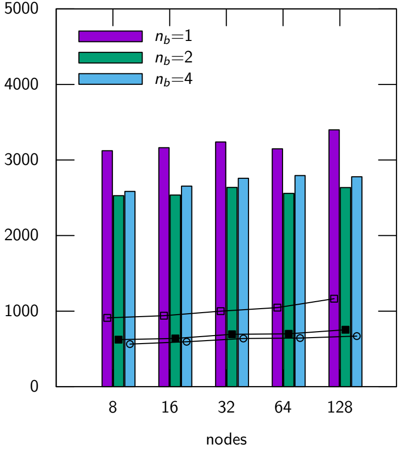
Setup

- Non-symmetric matrix from 7-point 3D PDE discretization ($n \approx 1.3 \cdot 10^8,\ n_{nz} \approx 9.4 \cdot 10^8$)
- Seeking 20 eigenvalues
- Ivy Bridge Cluster

Results

- $n_b = 2$: significantly faster
- $n_b = 4$: no further improvement

See Röhrig-Zöllner *et al* SISC 37(6), 2015



(const. bar heigth ≡ linear strong scaling)

Block orthogonalization schemes

- Given orthogonal vectors $(v_1, \ldots, v_k) = V$
- For $X \in \mathbb{R}^{n \times n_b}$ find orthogonal $Y \in \mathbb{R}^{n \times \tilde{n}_b}$ with

$$YR_1 = X - VR_2, \qquad \text{and} \qquad V^T Y = 0$$

Two phase algorithms

Phase 1 Project: $\tilde{X} \leftarrow (I - VV^T)X$

Phase 2 Orthogonalize: $Y \leftarrow f(\tilde{X})$

- communication optimal f:
  - CholQR or SVQB (Stathopoulos and Wu, SISC 2002)
  - TSQR (Demmel et al., SISC 2012)
- Each phase messes with the accuracy of the other. $\rightarrow$ iterate

Raising the computational intensity

Kernel fusion

Phase 2	$\tilde{X} \leftarrow XB,$	$C \leftarrow V^T \tilde{X}$
Phase 1	$\tilde{X} \leftarrow X - VC,$	$\tilde{B} \leftarrow \tilde{X}^T \tilde{X}$
Phase 3	$\tilde{X} \leftarrow XB,$	$\tilde{B} \leftarrow \tilde{X}^T \tilde{X}$

$\Rightarrow$ use SVQB

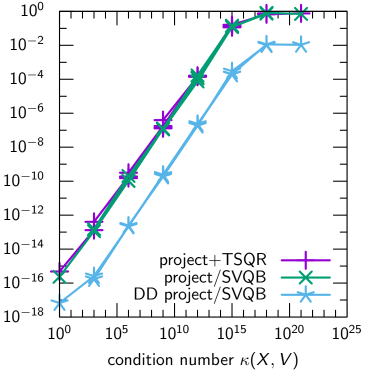
Increased precision

Idea Calculate value and error of each arithmetic operation

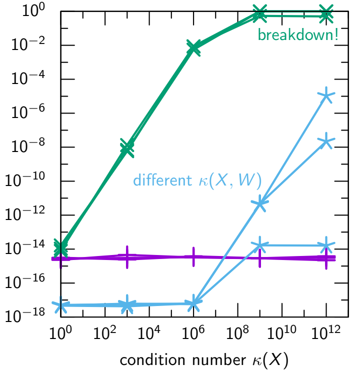
- Store intermediate results as double-double (DD) numbers
- Based on arithmetic building blocks (2Sum, 2Mult)
  Muller et al.: Handbook of Floating-Point Arithmetic, Springer 2010
- Exploit FMA operations (AVX2)

Results: accuracy after one iteration

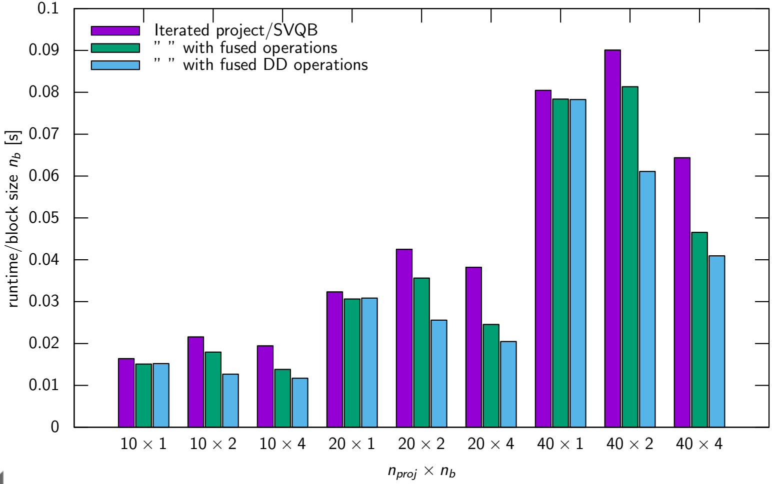
Error in $V^T Y$



Error in $Y^T Y$



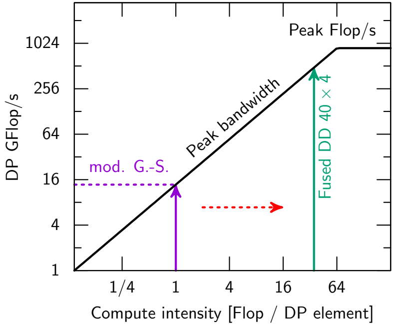
Results: runtime to orthogonality



Block orthogonalization aleviates memory gap

Faster through

- block algorithms
- kernel fusion
- increased precision *arithmetic* (not data)



Summary

Lessons learned

- don't use BLAS-3 for 'tall, skinny matrices'
- row-major storage "
- implement algorithms accordingly (avoid strides)
- use 'free' operations to increase accuracy/reduce iterations

Outlook: Peta and beyond

- reduction of synchronization kicks in
- increasing memory gap favors block methods
- crucial for ESSEX-II: matrix reordering and preconditioning