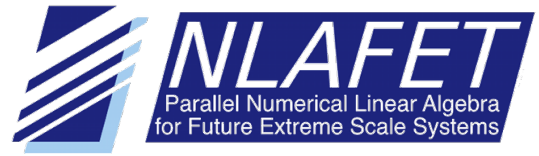


Optimized Polar Decomposition

for Modern Computer Architectures

Pierre.Blanchard[†]@manchester.ac.uk

NLA Group Meeting

Manchester, UK.

October 2, 2018

[†]School of Mathematics, The University of Manchester

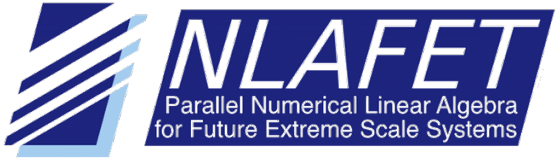
Joint work with

Introduction

Context

NLAFET (H2020 Project) - end April 2019

- Task-based implementation of algorithms in Plasma library
- Porting from **QUARK to OpenMP** Runtime System
- Novel SVD algorithms:
  - **2-stage SVD**: reduction to band format then eigensolver or SVD (D&C or QR).
  - **Polar Decomposition-based SVD**: QDWH iterations.
- Benefit of QDWH only at large scale on many-core architectures.



Polar Decomposition

The Polar Decomposition

For all *non-singular* $A \in \mathbb{C}^{m \times n}$, there exists a unique decomposition

$$A = UH$$

where

- $U \in \mathbb{C}^{m \times n}$ a set of n orthonormal columns
- $H \in \mathbb{C}^{n \times n}$ a Hermitian pos. semi-def. matrix

Reverse or Left PD

$$A = H_\ell U, \text{ with } H_\ell = UHU^* \in \mathbb{C}^{m \times m}$$

Relation with SVD

- Decompositions are equivalent
- Proofs of PD rely on SVD

Polar Decomposition

Applications

- Matrix nearness [Higham, 1986]
  - Nearest orthogonal matrix (*Procrustes* problem)
    - Factor Analysis, Multidimensional Scaling, ...
    - Optimization (Gradient descent)
  - Nearby (H) or nearest ($.5(H + A)$) pos. def. matrix

- Matrix functions: square root, p-th root, ...
  - ex: Square root of spd $A \in \mathbb{R}^{n \times n}$

$$\left.\begin{array}{ll} \text{If } A = LL^T & (\textbf{Chol}) \\ \text{and } L^T = UH & (\textbf{PD}) \end{array}\right\} \quad \Rightarrow \quad H = A^{1/2}$$

Application to Matrix Decompositions

- SVD of $A \in \mathbb{C}^{m \times n}$

$$\left.\begin{array}{ll} \text{If } A = UH & (\textbf{PD}) \\ \text{and } H = V\Sigma V^T & (\textbf{EVD}) \end{array}\right\} \Rightarrow \quad \begin{array}{ll} A = W\Sigma V^T & (\textbf{SVD}) \\ \text{with } W = UV & \end{array}$$

- QDWH-Eig not detailled here
  - Only a portion of the spectrum
  - Spectral D&C algorithm (QDWH + subspace iterations)

Polar Decomposition

Algorithms Root finding algorithms on singular values of U.

- Scaled **Newton's iterations**
  - 9 iterations (2nd order)
  - Backward stable
- **Q**R-based (**D**ynamically **W**eighted) **Halley's iterations**
  - 6 iterations (3rd order - Padé family)
  - Backward stable [Nakatsukasa et al., 2010]
  - Inverse-free + communications-friendly
  - Many cheap (Lvl3 Blas) flops

Polar Decomposition

Algorithms Root finding algorithms on singular values of U.

- Scaled **Newton's iterations**
  - 9 iterations (2nd order)
  - Backward stable
- **QR**-based (**D**ynamically **W**eighted) **Halley's iterations**
  - 6 iterations (3rd order - Padé family)
  - Backward stable [Nakatsukasa et al., 2010]
  - Inverse-free + communications-friendly
  - Many cheap (Lvl3 Blas) flops
- **Zolotarev functions**
  - 2 iterations (order 17!)
  - More flops but more paralellism
  - Well-suited for *high granularity computing* resources!
  - Best rational approximant of **matrix sign function**, see [Nakatsukasa and Freund, 2014] or [Higham, 2008, Ch.5].

State of the art implementation

H. Ltaief, D. Sukkari & D. Keyes (KAUST)

- PD, PD-SVD and PD-eig (PD=QDWH or Zolo)
- Distributed memory: **Scalapack + Chameleon + StarPU**
  - *Massively Parallel PD* [Ltaief et al., 2018]
  - Zolo up to 2.3× faster than QDWH
  - Cray XC40 system on 3200 Intel 16-core Haswell nodes

Other implementations

- QDWH in **Elemental** library
- QDWH-(S,E)VD with **Scalapack + ELPA** [Li et al., 2018]

QDWH based matrix decomposition

Algorithm

Convergence

Flops count

Polar Factor: $U = \lim_{k\to\infty} X_k$

QDWH iterations

$[U] = \texttt{qdwh}(A, \alpha, \beta, \varepsilon)$

$X_0 = A/\alpha,\ \ell_0 = \beta/\alpha$
$k = 0$
while $|1 - \ell_k| < \varepsilon$
    $a_k = h(\ell_k),\ b_k = g(a_k),\ c_k = a_k + b_k - 1$

    $[Q_1; Q_2]\, R = \texttt{qr}(\left[\sqrt{c_k} X_k; I_n\right])$
    $X_{k+1} = (b_k/c_k) X_k + (1/c_k)(a_k - b_k/c_k) Q_1 Q_2^H$

    $\ell_{k+1} = \ell_k (a_k + b_k \ell_k^2)/(1 + c_k \ell_k^2)$
    $k = k + 1$
end
$U = X_{k+1}$

Convergence of QDWH iterations

Number of iterations depends on conditioning $\kappa_2(A) = 1/\ell_0$.

- **Goal:** Mapping all singular values of X_0 in $[\ell_0, 1]$ to 1
- **Criterion:** closeness of $\sigma(X_k)$ to 1, *i.e.* the distance $|1 - \ell_k|$
- Estimate # of iterations *a priori* using

$$\sigma_i(X_{k+1}) = \sigma_i(X_k)\frac{a_k + b_k\sigma_i^2(X_k)}{1 + c_k\sigma_i^2(X_k)}$$

- Parameters $(a_k, b_k, c_k) \to (3, 1, 3)$ and optimized to ensure **cubical convergence**
- In practice, **less than** 6 **iterations** in double precision

Convergence of QDWH iterations

Estimating parameters $\alpha = \|A\|_2$ and $\ell_0 = \beta/\alpha$ with $\beta = 1/\|A^{-1}\|_2$

- Upper bound for $\alpha = \sigma_{max}(A)$
  - Can use $\hat{\alpha} = \|A\|_F$ or
  - 2-norm estimate based on power iterations (`normest`)
- Lower bound for $\ell_0 = \sigma_{min}(X_0)$
  - $\hat{\ell_0} = \|X_0\|_1/(\sqrt{n}\kappa_1(X_0))$
  - Estimate $\kappa_1(X_0)$
    - using `condest` [Higham and Tisseur, 2000] $(\mathbf{8/3n^3})$
    - or simply $\|X_0^{-1}\|_1$ using QR + triangular solve $(\mathbf{5/3n^3})$
  - Poor (over)-estimation of ℓ can increase # of iterations

Optimized QR it. [Nakatsukasa and Higham, 2013]

- Re-use Q in first it_{QR}
- Use identity structure to decrease it_{QR} cost from $(6 + \frac{2}{3})n^3$ to $5n^3$

Fast Cholesky iterations

Optimized QDWH Iterations

$[U] = \texttt{qdwh}(A, \alpha, \beta, \varepsilon)$

```
1  [...]
2      if c_k < 100 // PO-based it.   - 3mn^2 + n^3/3
3          Z = I_n + c_k X_k^* X_k
4          W = chol(Z)
5          X_{k+1} = (a_k/b_k) X_k + (a_k - b_k/c_k) (U_k W^{-1}) W^{-*}
6      else // QR-based it.   - 5mn^2
7  [...]
```

Cholesky-based iterations are not stable [Nakatsukasa and Higham, 2012]

- Forward error in X_{k+1} is bounded by $c_k \varepsilon$
- $c_k \to 3$ and $c_k \leq 100$ for $k \geq 2$ for all practical ℓ_0
- Hence, switch to PO iterations when $c_k < 100$

Matrix Decomposition

QDWH-PD

$[U, H] = \texttt{qdwh} - \texttt{pd}(A)$

1. $U = \texttt{qdwh}(A)$
2. $H = U^H A$ $\quad +2m^2 n$
3. $H = (H + H^H)/2$

QDWH-SVD

$[W, \Sigma, V^H] = \texttt{qdwh} - \texttt{svd}(A)$

1. $[U, H] = \texttt{qdwh} - \texttt{pd}(A)$
2. $[V, \Sigma] = \texttt{syev}(H)$ $\quad +4n^3$
3. $W = UV$ $\quad +2mn^2$

Additional cost

- PD needs 1 extra matrix multiplication (gemm)
- SVD needs 2 extra gemm + 1 syev
- Can both be implemented with similar memory footprint

Flops count: QDWH-PD

Overall count[1] of QDWH-PD with $m = n$

$$\left(8 + \frac{2}{3}\right) n^3 \#it_{QR} + \left(4 + \frac{1}{3}\right) n^3 \#it_{PO} + 2n^3$$

Nature of iterations with respect to condition number

κ_2	1	10^1-10^2	10^3-10^5	10^6-10^{13}	10^{14}-10^{16}
QR	1	1	2	2	3
PO	3	4	3	4	3
flops	$23 + \frac{2}{3}$	28	$32 + \frac{1}{3}$	$36 + \frac{2}{3}$	41
$+it_{QR}^{opt}$	$20 + \frac{1}{3}$	$24 + \frac{2}{3}$	29	$33 + \frac{1}{3}$	$37 + \frac{2}{3}$

Table 1: # of QR and PO iterations and flops count ($/n^3$) for QDWH-PD.

[1] without $\frac{5}{3}n^3$ for estimating ℓ_0 or exploiting trailing identity matrix structure.

Flops count: QDWH-PD

Overall count[1] of QDWH-PD with $m = n$

$$\left(8 + \frac{2}{3}\right) n^3 \#it_{QR} + \left(4 + \frac{1}{3}\right) n^3 \#it_{PO} + 2n^3$$

Nature of iterations with respect to condition number

κ_2	1	10^1-10^2	10^3-10^5	10^6-10^{13}	10^{14}-10^{16}
QR	0		...		2
PO	2		...		4
flops	$10 + \frac{2}{3}$		...		$36 + \frac{2}{3}$
$+it_{QR}^{opt}$	$12 + \frac{1}{3}$		...		$33 + \frac{1}{3}$

Table 1: # of QR and PO iterations and flops count ($/n^3$) for QDWH-PD.

[1] without $\frac{5}{3}n^3$ for estimating ℓ_0 or exploiting trailing identity matrix structure.

Flops count: QDWH-SVD

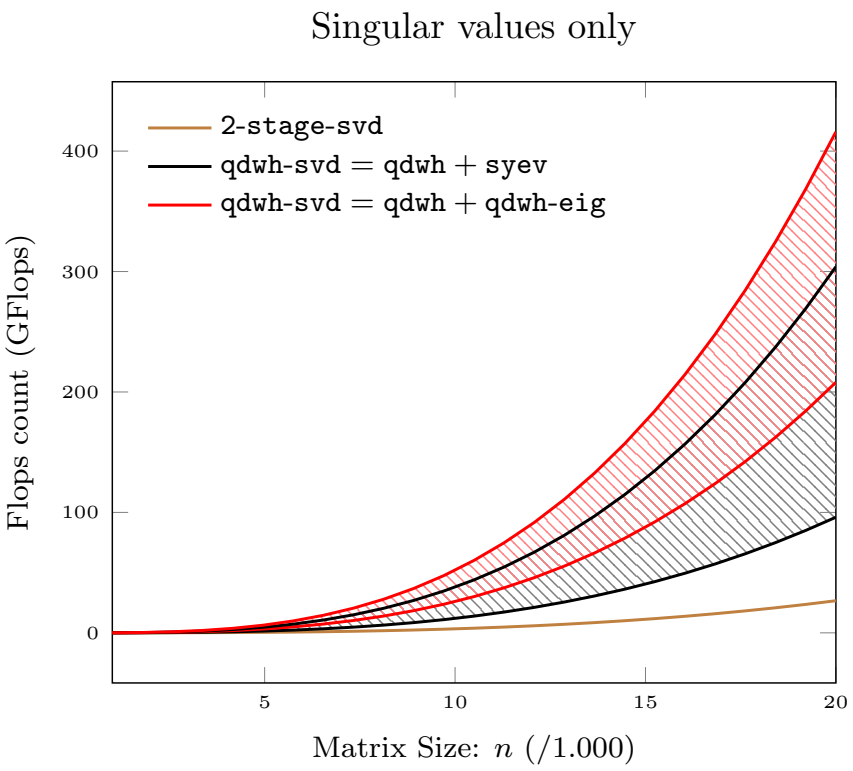
- QDWH-PD: $(10 + \frac{2}{3}) \leq \frac{\#flops}{n^3} \leq (36 + \frac{2}{3})$
- Symmetric Eigensolver + Multiplication

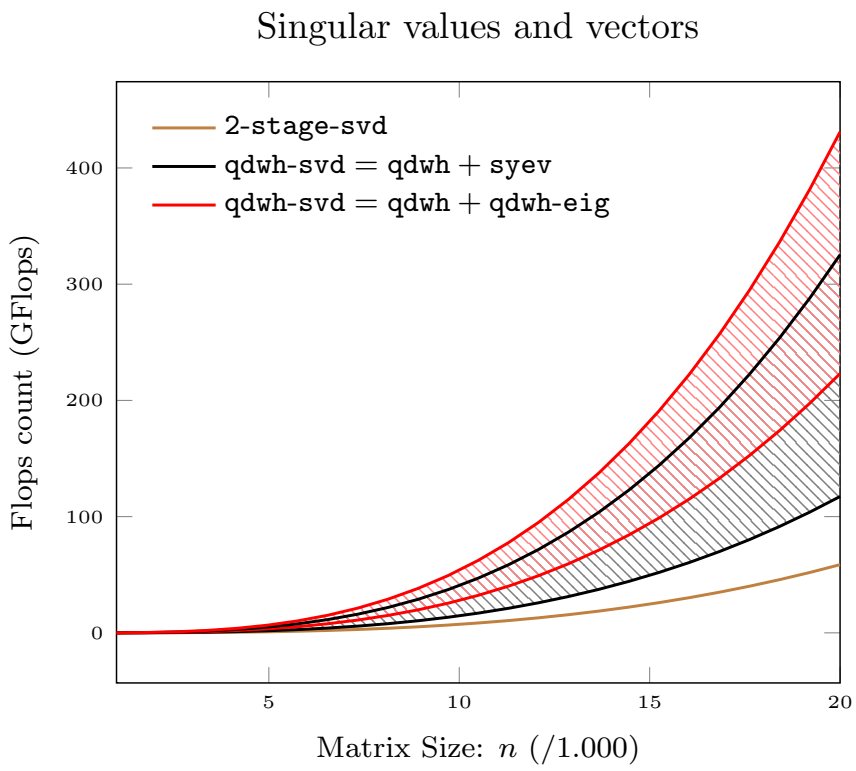
- 2-stage-eig:
  - $\frac{4}{3}$
  - 4 with vecs
- QDWH-eig:
  - $(16 + \frac{1}{9}) \leq \ldots \leq (50 + \frac{7}{9})$
  - $(17 + \frac{4}{9}) \leq \ldots \leq (52 + \frac{4}{9})$ with vecs

	Σ	U, Σ, V
`dges(vd/dd)`	$\mathbf{2} + \frac{2}{3}$	**22**
`2-stage-svd`	$\frac{10}{3} = \mathbf{3} + \frac{1}{3}$	$\frac{10}{3} + 4 = \mathbf{7} + \frac{1}{3}$
`QDWH-svd`	(+`2-stage-eig`) $\mathbf{12} \leq \ldots \leq \mathbf{38}$ (+`QDWH-eig`) $\mathbf{26} + \frac{5}{9} \leq \ldots \leq \mathbf{52} + \frac{5}{9}$	$\mathbf{14} + \frac{2}{3} \leq \ldots \leq \mathbf{40} + \frac{2}{3}$ $\mathbf{27} + \frac{8}{9} \leq \ldots \leq \mathbf{53} + \frac{8}{9}$

Table 2: Floating point operations counts $(/n^3)$ for SVD.

Flops Count


Singular values only
2-stage-svd
qdwh-svd = qdwh + syev
qdwh-svd = qdwh + qdwh-eig
Flops count (GFlops)
400
300
200
100
0
5
10
15
20
Matrix Size: n (/1.000)

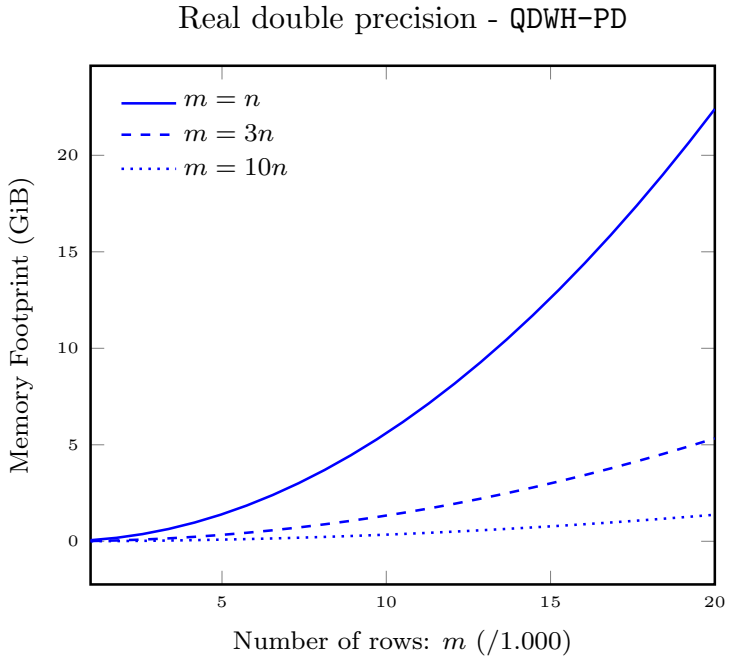

Singular values and vectors
2-stage-svd
qdwh-svd = qdwh + syev
qdwh-svd = qdwh + qdwh-eig
Flops count (GFlops)
400
300
200
100
0
5
10
15
20
Matrix Size: n (/1.000)

Memory footprint

Stored matrices

- $A \in \mathbb{R}^{m \times n}$
- $U \in \mathbb{R}^{m \times n}$ and $H \in \mathbb{R}^{n \times n}$
- $B = \left[\sqrt{c_k} X_k; I_n\right] \in \mathbb{R}^{(m+n) \times n}$
- $Q = [Q_1; Q_2] \in \mathbb{R}^{(m+n) \times n}$

$\Rightarrow 4mn + 3n^2$ entries



Memory available on Intel nodes or NVIDIA accelerators

- Intel KNL has up to **16GiB** of MCDRAM (depending on mode)
- Haswell/Sandy Bridge around 64GiB
- Skylake: 64/128GiB
- Tesla V100 GPU: 16/32GiB

Experiments

Runtime System

QR-Optimization

Architecture

Numerical Experiments

Real square matrices in double precision

- `dlatms`: prescribed condition number and spectrum
- `dlarnv`: entries sampled at random in $[0, 1]$
- $m = n = 2.000, \ldots, 16.000$

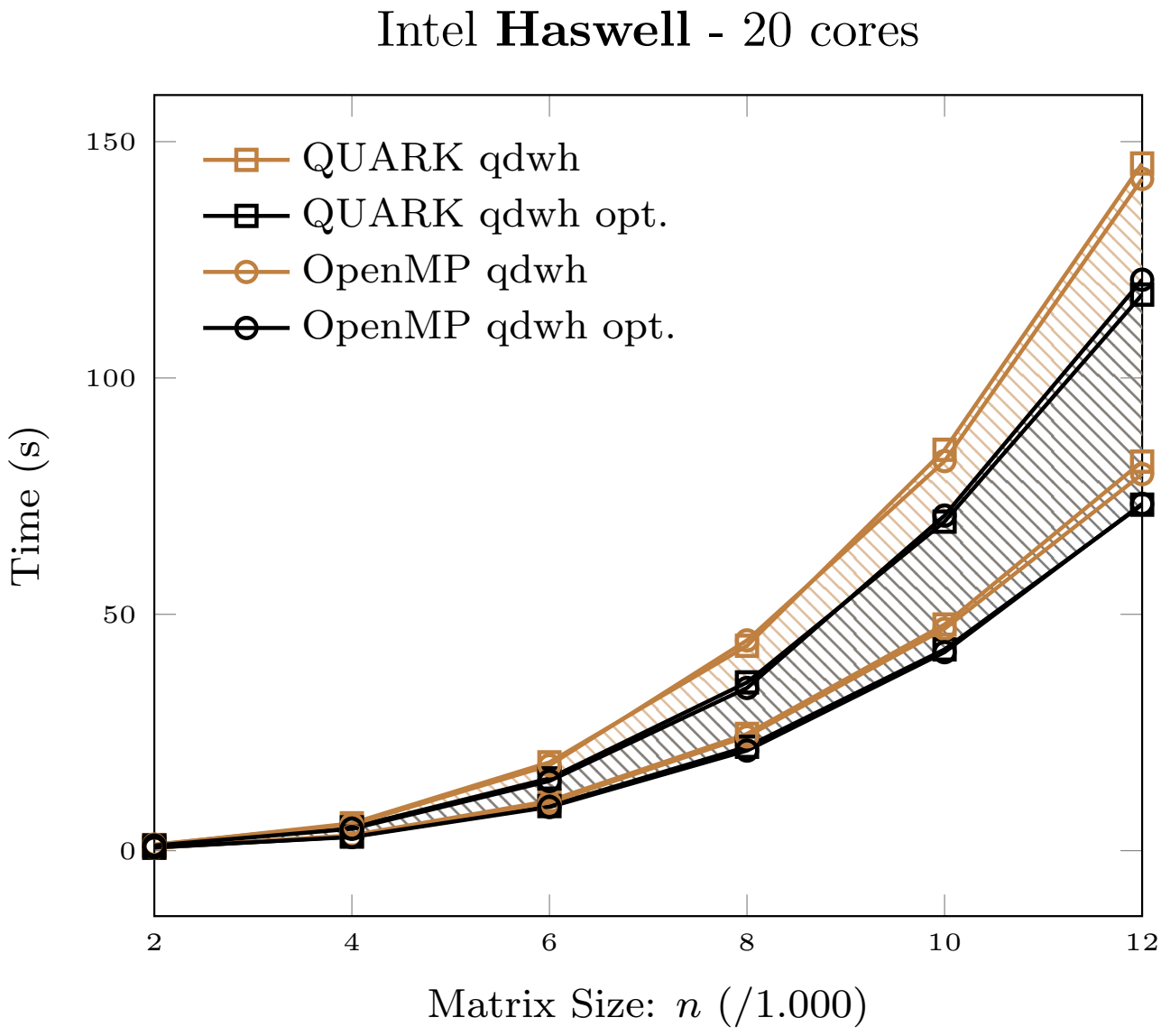
Computer architectures (Intel)

- Haswell: 20 cores
- Sandy Bridge: 16 cores
- KNL: 68 cores

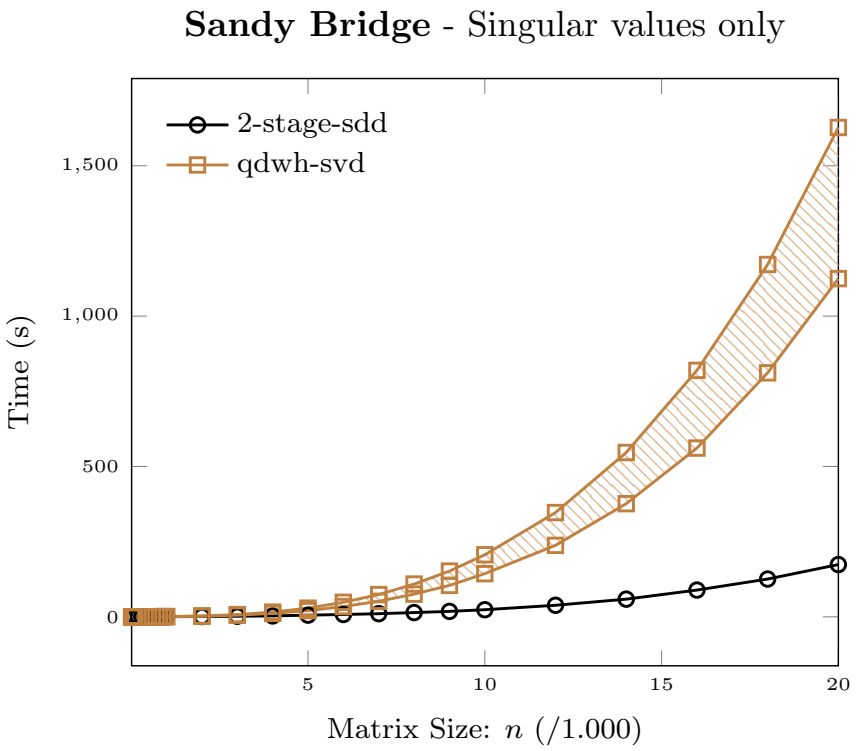
Runtime systems

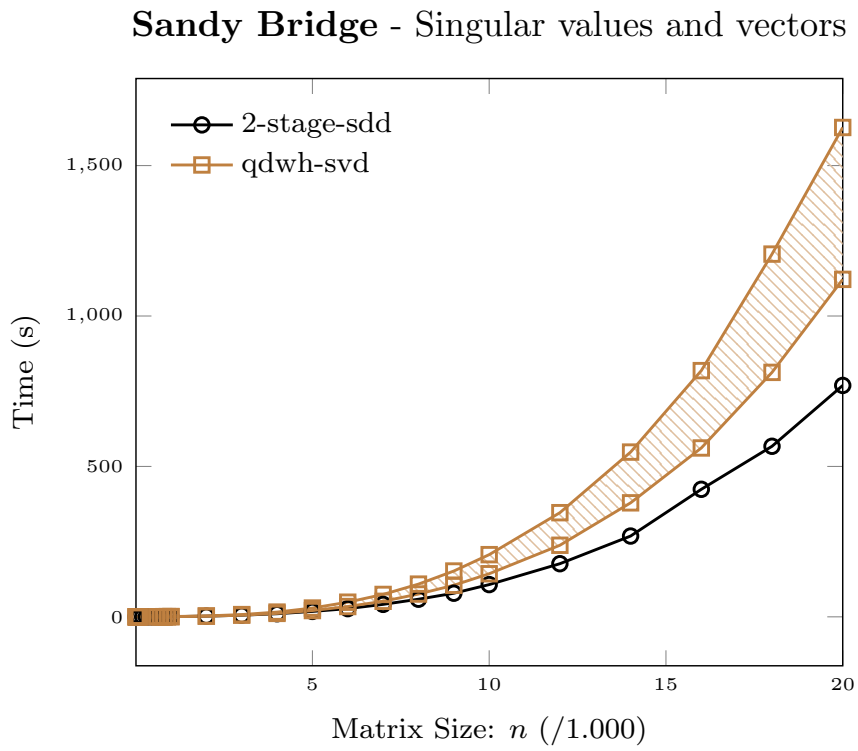
- Quark (Plasma-2.8)
- **NEW** OpenMP (Plasma-17)

QDWH on runtime systems



QDWH-SVD: Sandy Bridge



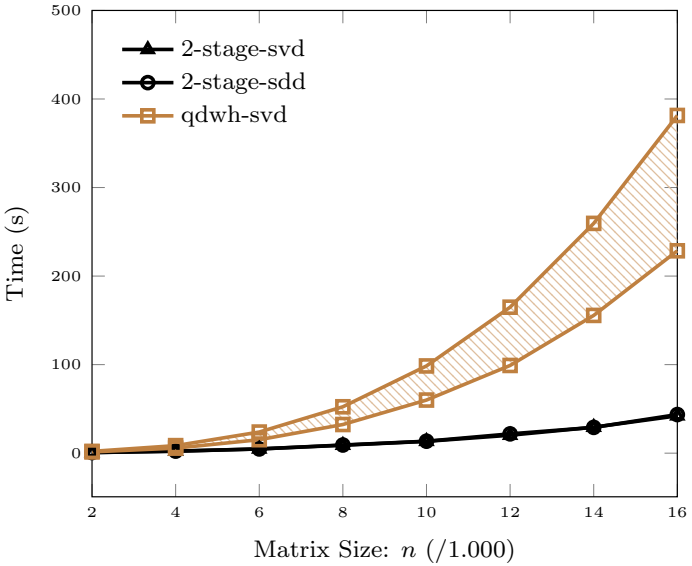


κ_2	$\frac{\#flops(QDWH-SVD)}{\#flops(SDD)}$	SandyBridge	Haswell	KNL
1	7.8	1.5	1.4	$\frac{1}{1.6}$
10^{16}	13	2.5	2.3	$\frac{1}{1.1}$

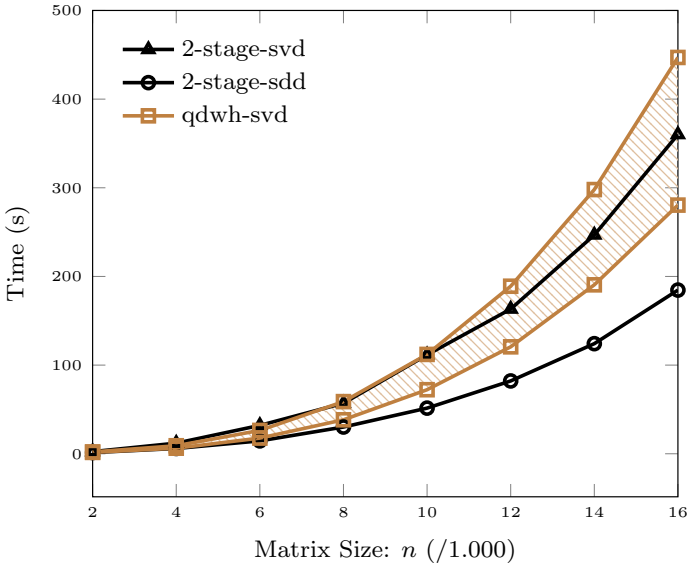
Table 3: Flop ratio vs speedup for QDWH-SVD (with vectors) and $n = 14.000$.

QDWH-SVD: Haswell vs KNL

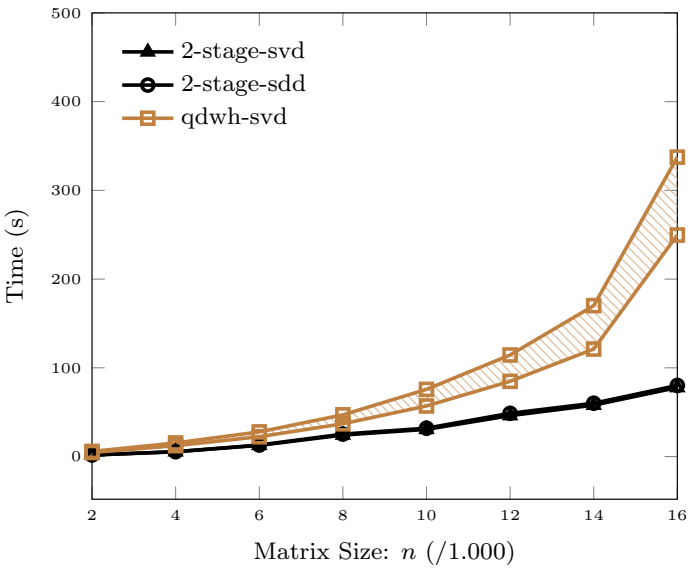
Haswell - Singular values only



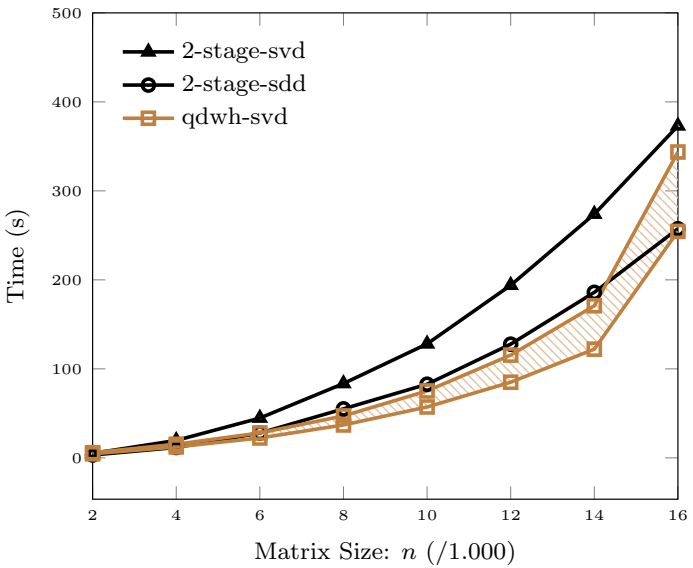
Haswell - Singular values and vectors



KNL - Singular values only

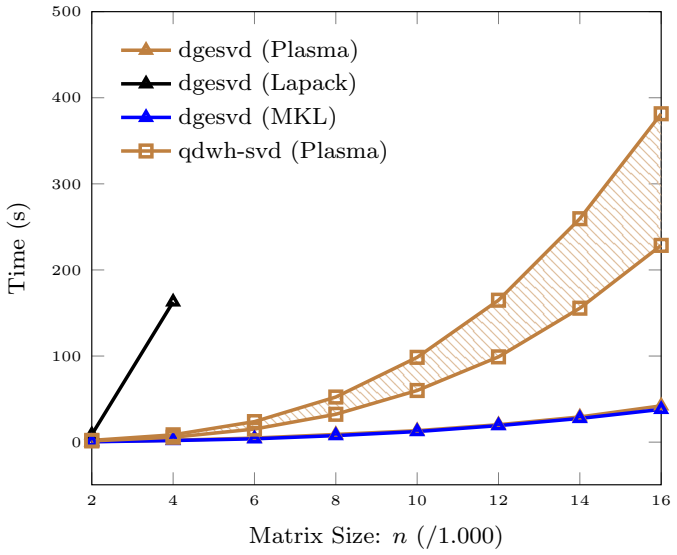


KNL - Singular values and vectors

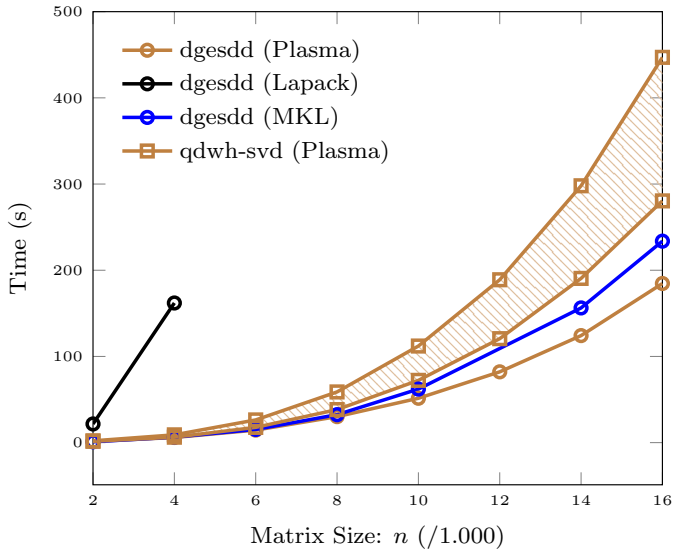


QDWH-SVD: Plasma vs MKL

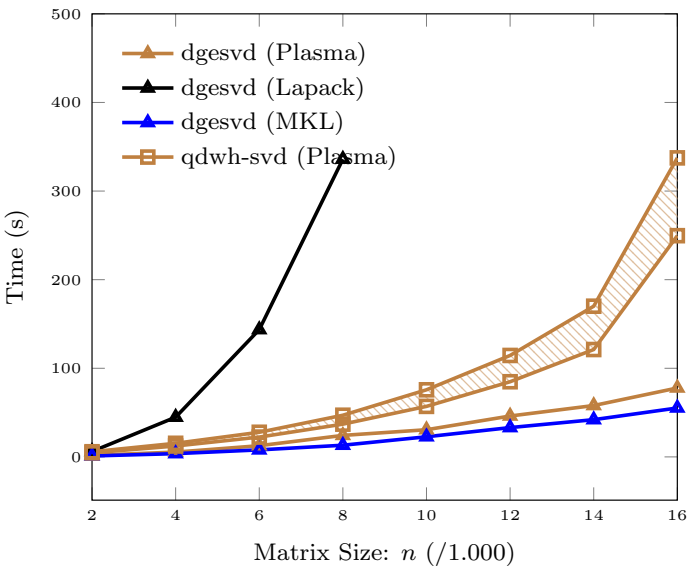
Haswell - Singular values only



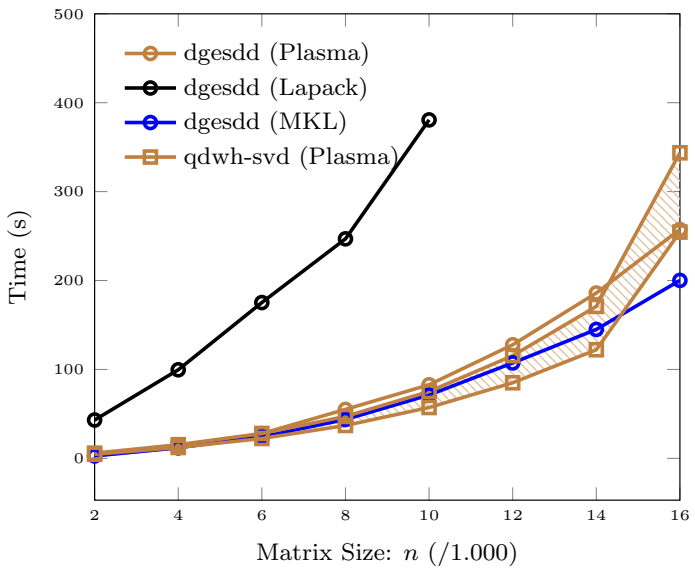
Haswell - Singular values and vectors



KNL - Singular values only



KNL - Singular values and vectors



Ongoing work & Perspectives

Ongoing work & Perspectives

StarPU implementation

- Heterogenous architectures
- Distributed memory version
- Accelerators (NVIDIA GPUs)

Applications

- Multidimensional Scaling
  - QDWH-PD and Eig
  - Single/Half Precision
- Matrix p-roots

Algorithms

- Mixed-precision
  - Multiplications, Cholesky and QR
  - on high-end GPUs (with Tensor-Core features)
- Zolotarev PD
  - 2 iterations
  - Extra flops embarassingly parallel
  - Larger memory footprint

Questions

Thank you for your attention.

References i

N. Higham. Computing the polar decomposition with applications. *SIAM Journal on Scientific and Statistical Computing*, 1986. ISSN 0196-5204. doi: 10.1137/0907079.

N. J. Higham. *Functions of Matrices: Theory and Computation*. 2008. ISBN 978-0-898716-46-7. doi: 10.1137/1.9780898717778.

N. J. Higham and F. Tisseur. A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. 21(4): 1185–1201, 2000. doi: 10.1137/S0895479899356080.

S. Li, J. Liu, and Y. Du. A new high performance and scalable svd algorithm on distributed memory systems. *CoRR*, abs/1806.06204, 2018.

H. Ltaief, D. E. Sukkari, A. Esposito, Y. Nakatsukasa, and D. E. Keyes. Massively parallel polar decomposition on distributed-memory systems. 2018.

Y. Nakatsukasa and R. W. Freund. Using zolotarevs rational approximation for computing the polar, symmetric eigenvalue, and singular value decompositions. *SIAM Rev. To appear*, 2014.

Y. Nakatsukasa and N. J. Higham. Backward stability of iterations for computing the polar decomposition. 33(2):460–479, 2012. doi: 10.1137/110857544.

Y. Nakatsukasa and N. J. Higham. Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD. *SIAM Journal on Scientific Computing*, 35(3):A1325–A1349, jan 2013. doi: 10.1137/120876605. URL https://doi.org/10.1137/120876605.

Y. Nakatsukasa, Z. Bai, and F. Gygi. Optimizing halley's iteration for computing the matrix polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2700–2720, jan 2010. doi: 10.1137/090774999. URL https://doi.org/10.1137/090774999.