jh200023-NAHI

Hierarchical Low-Rank Approximation Methods on Distributed Memory and GPUs

Tokyo Tech.: R. Yokota, K. Osawa, H. Naganuma, H. Otomo, S. Iwase, S. Deshmukh, P. Spalthoff, M.R. Apriansyah, Q. Ma, Y. Ueno, H. Nakta, L. Kaku, T. Spendlhoffer, M. Shibuya, T. Shohata, H. Hoshino, A. Li, S. Takashima, T. Ito, X. Zhang
U. Tokyo: A. Ida, K. Nakajima, T. Hanawa, T. Hoshino
Hokkaido U.: T. Iwashita, T. Fukaya
Nagoya U.: S. Ohshima
Kyoto U.: T. Hiraishi
Sandia NL: I. Yamazaki

JHPCN 12th Symposium
Online, 2020/07/09

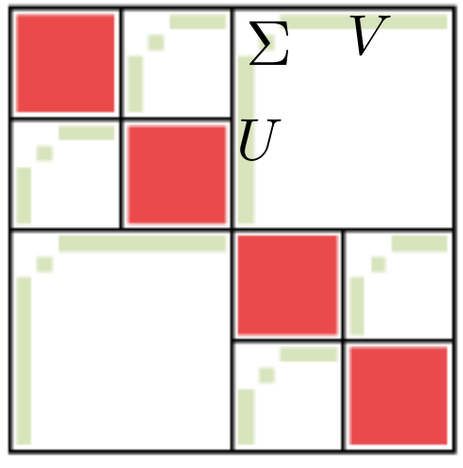
Why Hierarchical Low-Rank Approximation?

Replace dense linear algebra

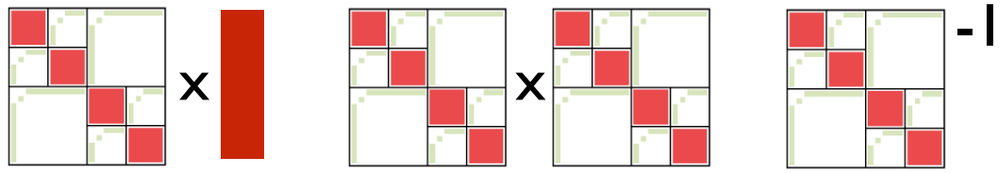
Compute : $\mathcal{O}(N^3) \longrightarrow \mathcal{O}(N \log^2 N)$

Memory : $\mathcal{O}(N^2) \longrightarrow \mathcal{O}(N)$

Hierarchical off-diagonal blocks
Approximated with low rank



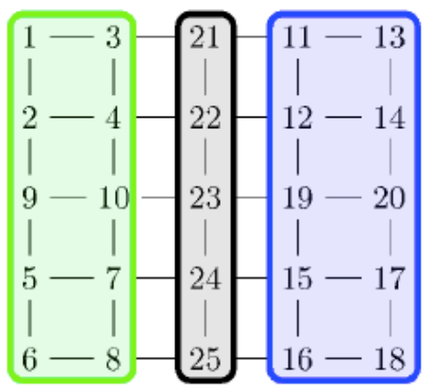
Augment sparse linear algebra

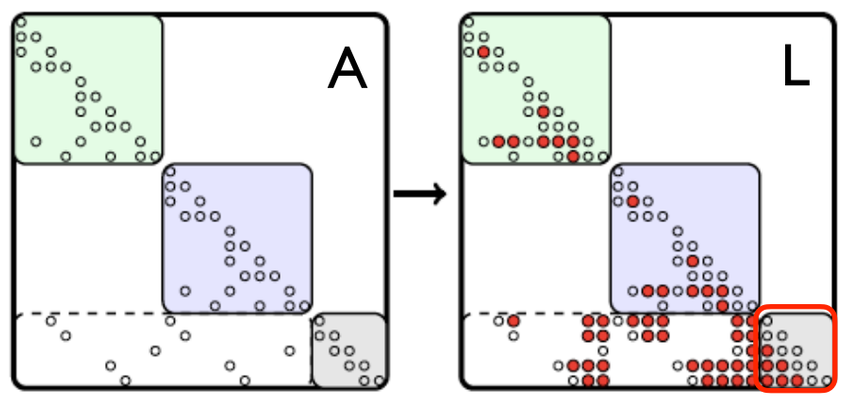


Sparse direct solvers

Schur complement (frontal matrix) is dense but numerically low-rank

Nested dissection





Schur complement

Iterative solvers

Use small rank to precondition

Less sensitive to matrix condition than multigrid

Replacing Exact Linear Algebra with Low-Rank

Exact

$\mathcal{O}(N^3)$

Application

ScaLAPACK | cuSolverMG

LAPACK PLASMA | MKL | cuSolverDN MAGMA

BLAS | CUBLAS

CPU
FP64

GPU
FP32

Distributed
QR
LU
MatMul
Mat-vec

Approximate

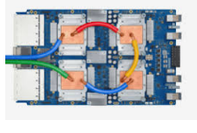
$\mathcal{O}(N \log^2 N)$

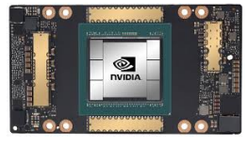
App.

HACApK
HiCMA

HBLAS

?PU
TF32, bfloat16





Groups

Germany

- Shared memory H-LU
  Kriemann (2014)
- Nested cross approximation
  Börm & Christophersen (2014)
- H^2-matrix for eigenvalues
  Berner et al. (2015)
- OmpSs H-LU
  Aliaga et al. (2017)
- GCA H^2-matrix
  Börm et al. (2018)

Berkeley

- HSS2D
  Xia (2014)
- HSS selected inversion
  Xia et al. (2015)
- Superfast DC eigenvalue
  Vogel, et al. (2016)
- Shared memory HSS MF
  Ghysels et al. (2016)
- Distributed HSS MF
  Rouet et al. (2016)

Japan

- Distributed H-matrix
  Ida et al. (2015)
- Distributed GPU H-matrix
  Yamazaki et al. (2018)
- Lattice H-matrix
  Ida (2018)
- GPU load-balance ACA
  Hoshino et al. (2018)
- GPU autotuning
  Ohshima et al. (2018)

EPFL

- HODLR QR
  Kressner et al. (2018)

Minnesota

- Multilevel Low-Rank
  Li & Saad (2013)
- DD Low-Rank
  Li & Saad (2014)
- Schur Low-Rank
  Li & Saad (2015)
- Multilevel Schur Low-Rank
  Xi et al. (2016)
- SMASH
  Cai et al. (2018)

Stanford(Ying)

- O(N) RS 2-D
  Corona (2015)
- HIF for PDEs
  Ho & Ying (2016)
- Distributed memory HIF
  Li & Ying (2016)
- RS for maximum likelihood
  Minden et al. (2016)
- RS with strong admissibility
  Minden et al. (2017)
- Quantized Tensor Train
  Corona et al. (2017)

Stanford(Darve)

- HODLR multifrontal
  Aminfar et al. (2016)
- IFMM precond. Stokes
  Coulier et al. (2017)
- IFMM precond. Helmholtz
  Takahashi et al. (2017)
- Extended sparsification IFMM
  Pouransari et al. (2017)
- Non-extensive sparsification
  Sushnikova et al. (2017)
- Sparsified Nested Dissection
  Cambier et al. (2019)

INRIA

- BLR multifrontal
  Amestoy et al. (2015)
- BLR multicore
  Amestoy et al. (2017)
- Multilevel BLR
  Amestoy et al. (2017)

KAUST

- BLR Cholesky
  Akbudak et al. (2017)
- Batched QR, SVD
  Boukaram et al. (2018)
- GPU MatVec
  Boukaram et al. (2019)

Texas (Biros)

- inv-ASKIT
  Yu et al. (2016)
- Distributed inv-ASKIT
  Yu et al. (2017)
- GOFMM
  Yu et al. (2017)
- Distributed GOFMM
  Yu et al. (2018)

Replacing Exact Linear Algebra with Low-Rank

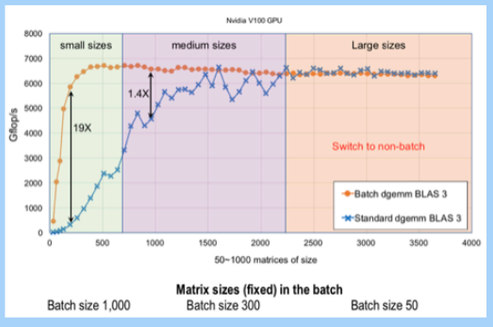
	CPU	GPU	MPI	MPI+GPU
Mat-vec	1999Hackbusch	2016JHPCN	2017JHPCN	2018JHPCN
LU	2014Kriemann	2019JHPCN	2018JHPCN	2020JHPCN
QR	2019JHPCN	x	x	x
EigenValue	2020JHPCN	x	x	x

Batched BLAS

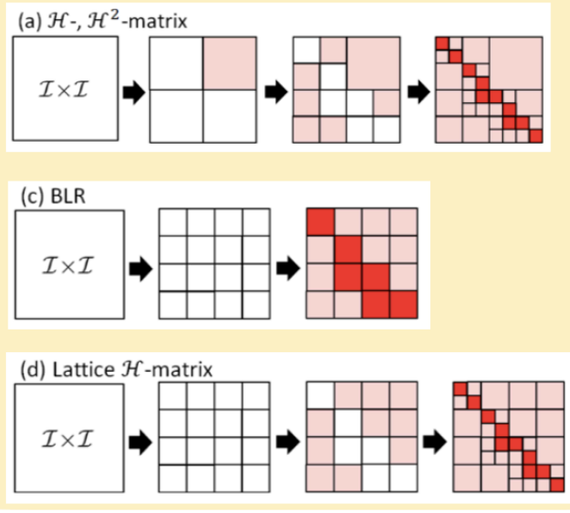
batch GEMM
batch GEMV

Intel Math Kernel Library

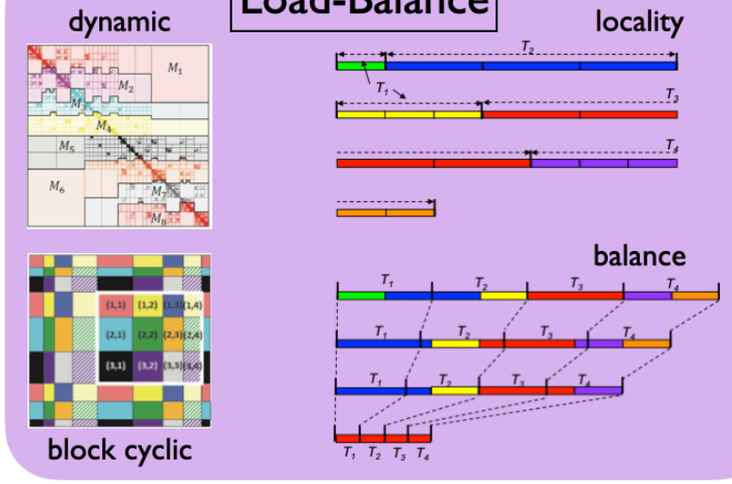
MAGMA



BLR + H-matrix

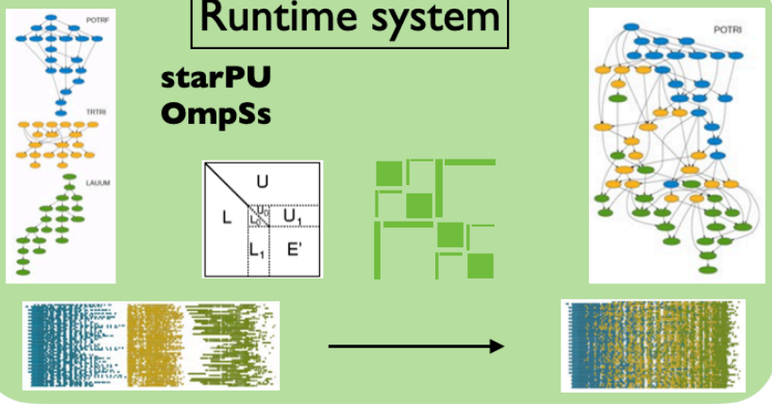


Load-Balance



Runtime system

starPU
OmpSs



Approximate

$$\mathcal{O}(N \log^2 N)$$

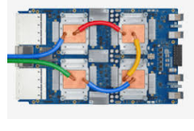
Distributed
QR
LU
MatMul
Mat-vec

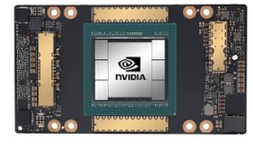
App.

HACApK
HiCMA

HBLAS

?PU
TF32, bfloat16





Future Plans for FY2020

1) Use of uniform basis as a nested basis block low-rank (BLR) matrix

2) QR factorization on TensorCores with refinement

3) Eigenvalue computation based on BLR-QR

4) GPU implementation of lattice H-matrix

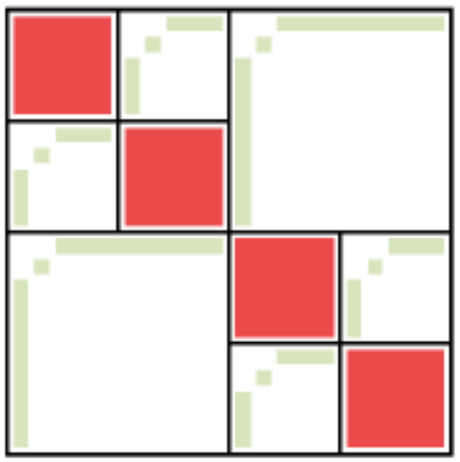
	CPU	GPU	MPI	MPI+GPU
Mat-vec	1999Hackbusch	2016JHPCN	2017JHPCN	2018JHPCN
LU	2014Kriemann	2019JHPCN	2018JHPCN	2020JHPCN
QR	2019JHPCN	x	x	x
EigenValue	2020JHPCN	x	x	x

Use of uniform basis as a nested basis block low-rank (BLR) matrix

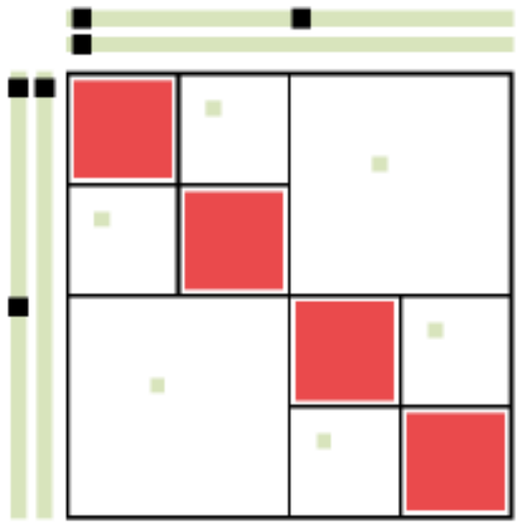
	Nested Basis	Admissibility
H-matrix	No	Strong
H^2-matrix	Yes	Strong
HODLR	No	Weak
HSS	Yes	Weak
BLR	No	non-hierarchical

Nested Basis

Non-nested

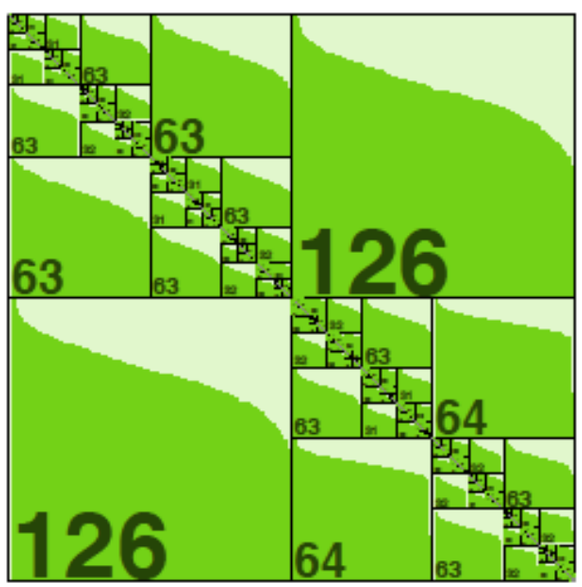


Nested

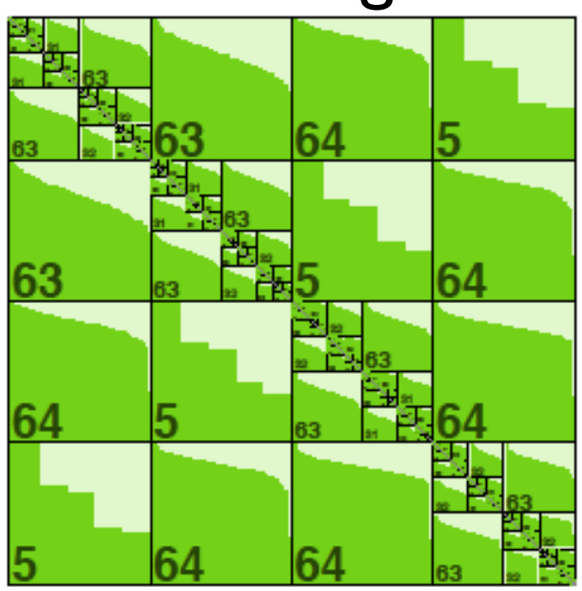


Admissibility

Weak

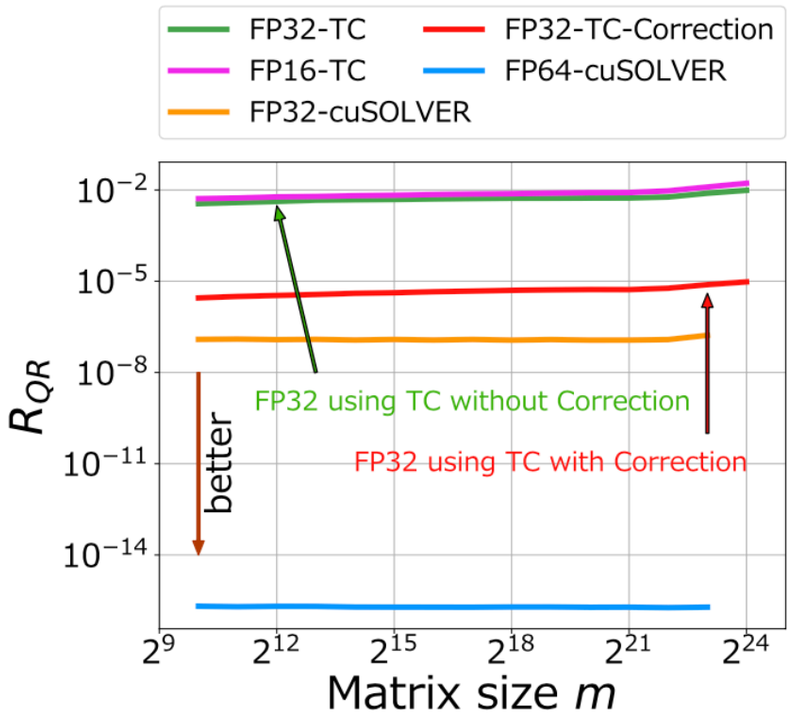


Strong

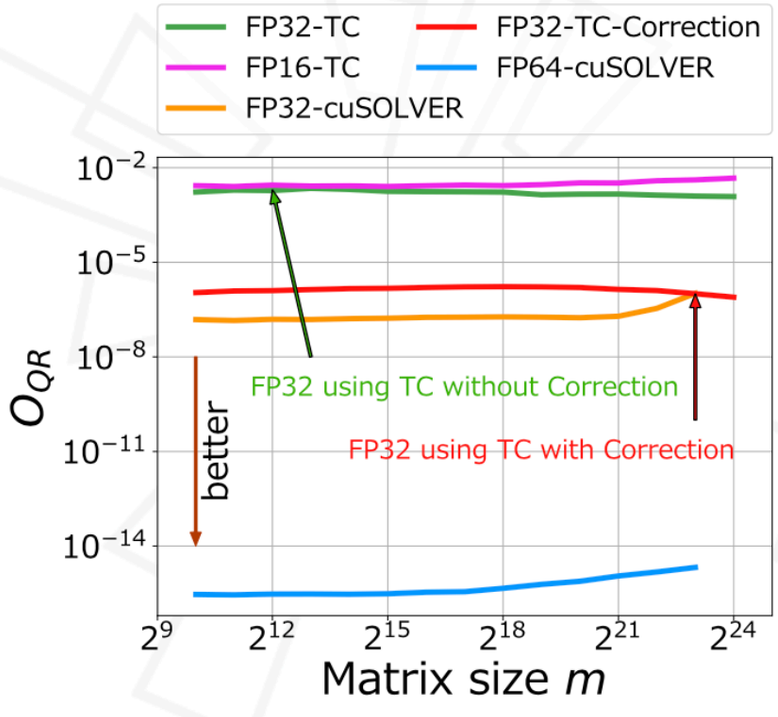


QR factorization on TensorCores with refinement

Residual



Orthogonality



FP16 FP32

$$\mathbf{M}_{\mathrm{FP16}} \xleftarrow{\text{narrowing}} \mathbf{M}_{\mathrm{FP32}}$$

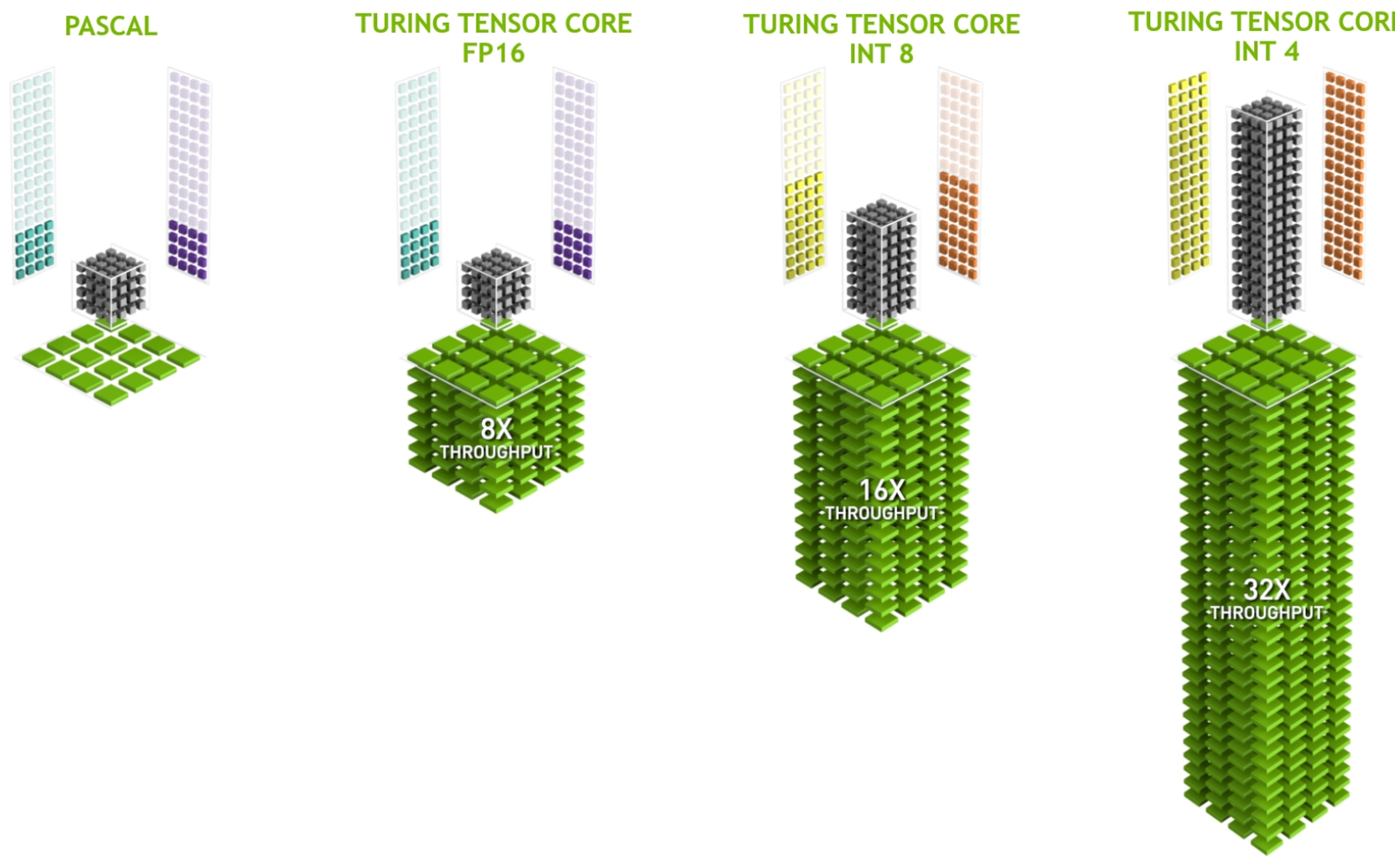
Without correction

$$\mathbf{Q}_{\mathrm{FP32}} \leftarrow \mathbf{H}_{\mathrm{FP16}}\,\mathbf{Q}_{\mathrm{FP16}}$$
$$\mathbf{R}_{\mathrm{FP32}} \leftarrow \mathbf{H}_{\mathrm{FP16}}\,\mathbf{R}_{\mathrm{FP16}}$$

With correction

$$\mathbf{Q}_{\mathrm{FP32}} \leftarrow \mathbf{H}_{\mathrm{FP16}}\,\mathbf{Q}_{\mathrm{FP16}} + \Delta\mathbf{H}_{\mathrm{FP16}}\,\mathbf{Q}_{\mathrm{FP16}} + \mathbf{H}_{\mathrm{FP16}}\,\Delta\mathbf{Q}_{\mathrm{FP16}}$$
$$\mathbf{R}_{\mathrm{FP32}} \leftarrow \mathbf{H}_{\mathrm{FP16}}\,\mathbf{R}_{\mathrm{FP16}} + \underbrace{\Delta\mathbf{H}_{\mathrm{FP16}}\,\mathbf{R}_{\mathrm{FP16}} + \mathbf{H}_{\mathrm{FP16}}\,\Delta\mathbf{R}_{\mathrm{FP16}}}_{\text{Correction terms}}$$

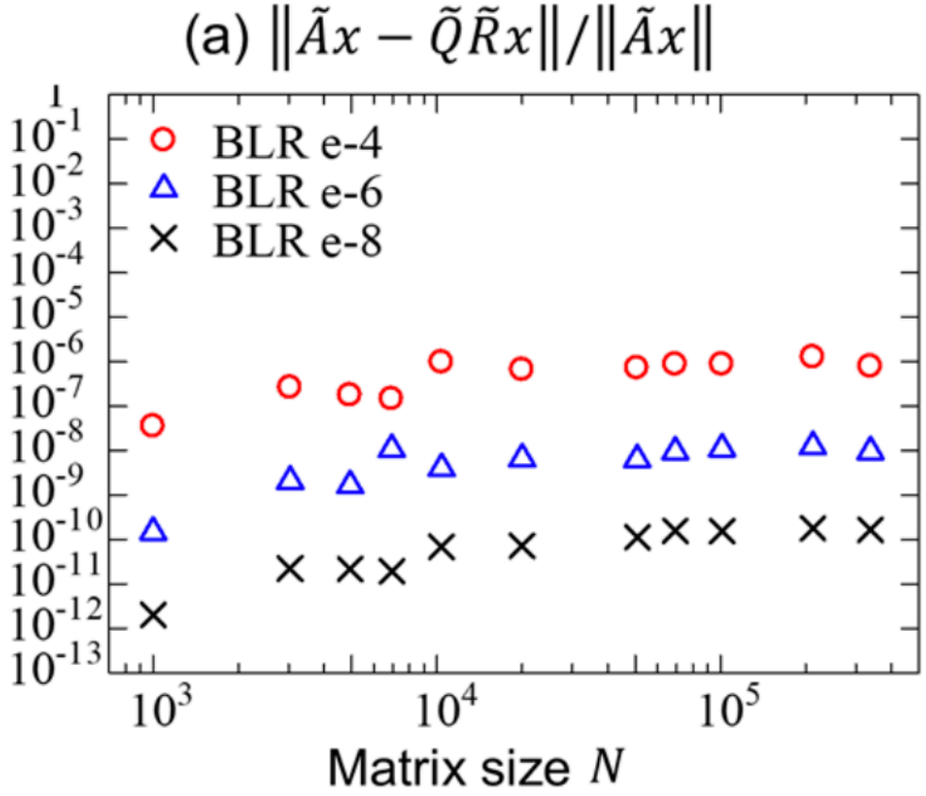
where $\Delta\mathbf{M}_{\mathrm{FP16}} \xleftarrow{\text{narrowing}} \mathbf{M}_{\mathrm{FP32}} - \mathbf{M}_{\mathrm{FP16}}$

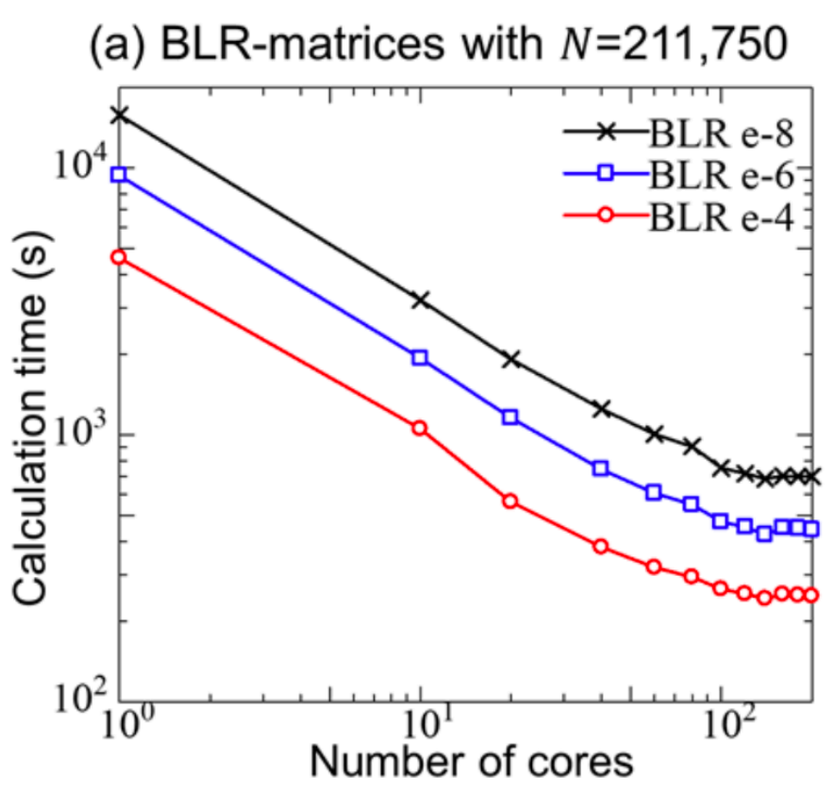


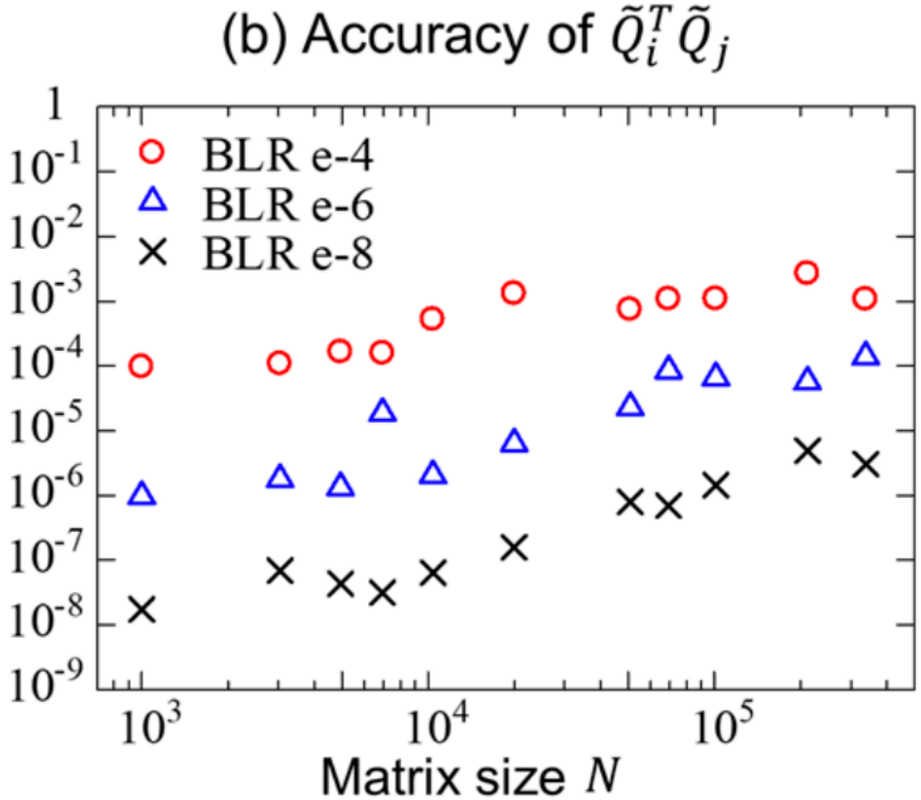
Eigenvalue computation based on BLR-QR

QR algorithm for computing eigenvalues

$$A_{k+1} = R_k Q_k = Q_k^{-1} Q_k R_k Q_k = Q_k^{-1} A_k Q_k = Q_k^{\mathsf{T}} A_k Q_k$$

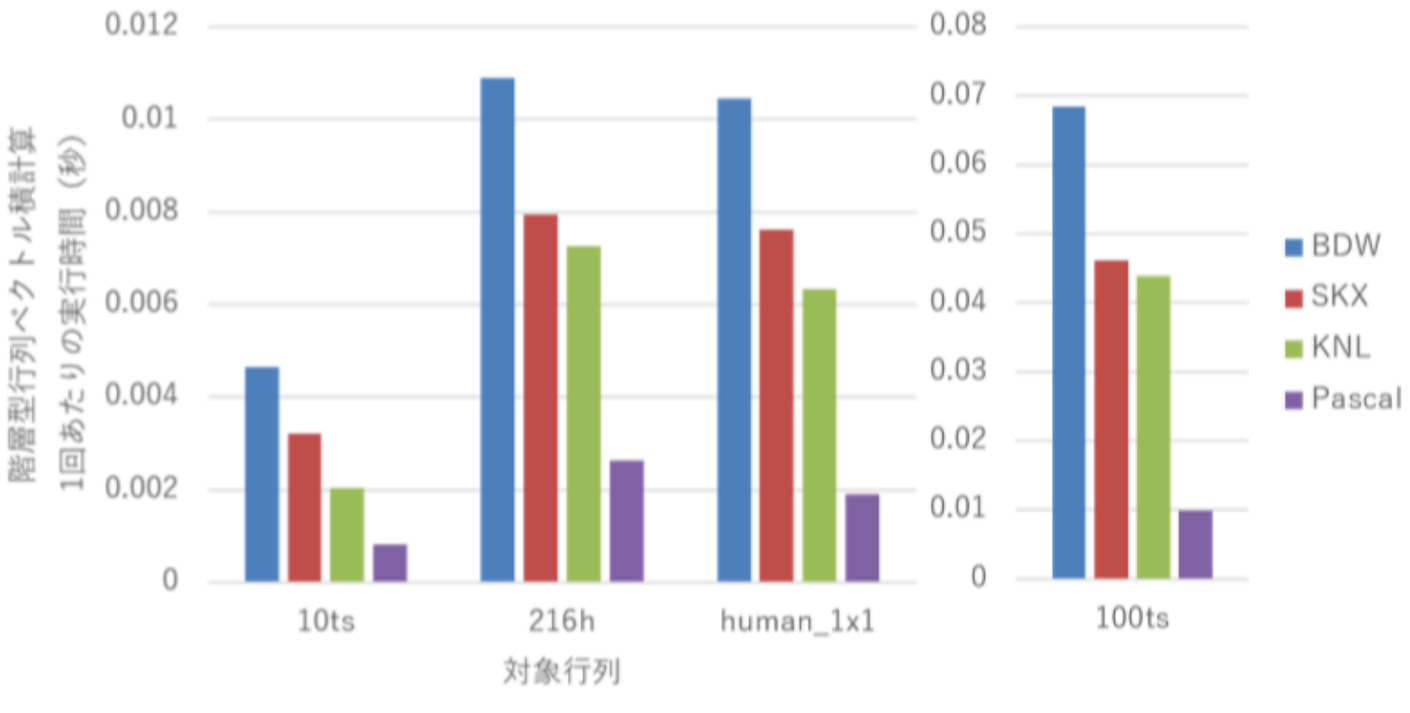




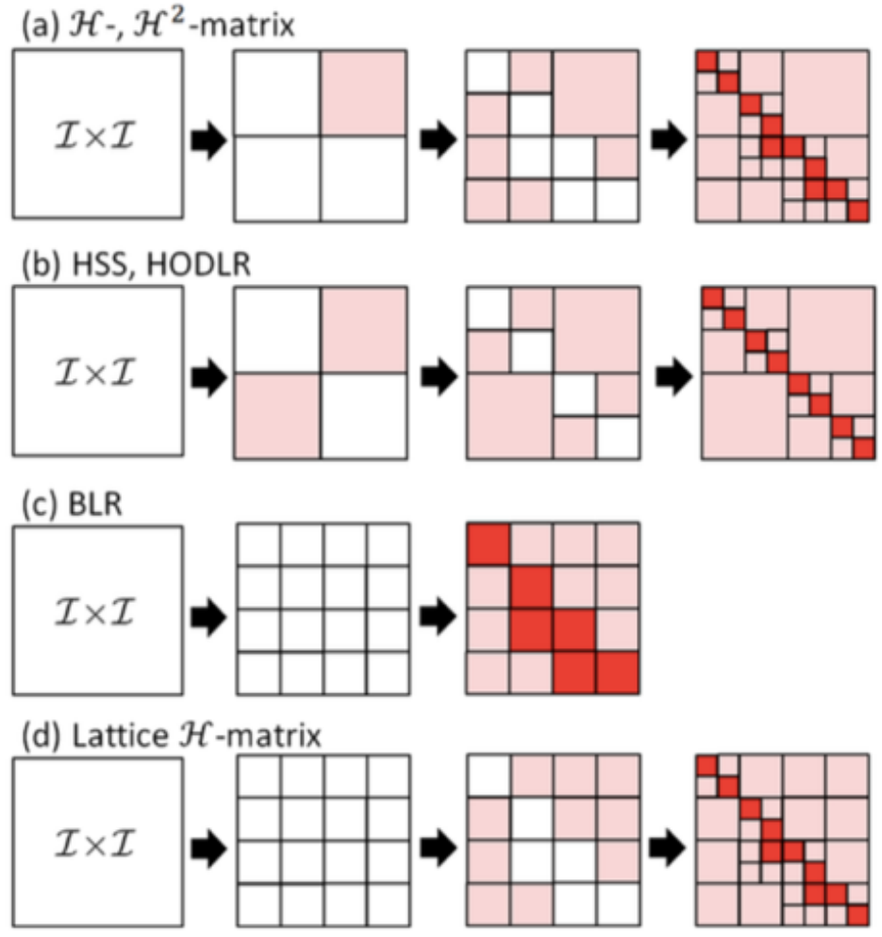


A. Ida, H. Nakashima, T. Hiraishi, I. Yamazaki, R. Yokota, and T. Iwashita, QR Factorization of Block Low-Rank Matrices with Weak Admissibility Condition, IPSJ Trans. ACS (2019).

GPU implementation of lattice H-matrix



	型番	HPL性能 GFLOPS	STREAM Triad性能 GB/s
BDW (Broadwell-EP)	Intel Xeon E5-2695 v4	570	65
SKX (Skylake-SP)	Intel Xeon Gold 6140	900	95
KNL (Knights Landing)	Intel Xeon Phi 7150	1800	495
Pascal	NVIDIA Tesla P100	4000	550



Using batched GEMV for lattice H-matrix

Batched MAGMA on GPUs