PLASMA INTERTWinING

Jakub Šístek

joint work with

Maksims Abalenkovs, George Bosilca, Jack Dongarra, Reazul Hoque,
Jakub Kurzak, Julien Langou, David Stevens, and Mawussi Zounon

University of Manchester, UK
Institute of Mathematics, Czech Academy of Sciences, Prague, Czechia
University of Tennessee, USA
University of Colorado Denver, USA

Innovative Computing Lab
Knoxville, USA, September 21, 2018

Outline

Introduction and motivation

PLASMA library and OpenMP

QR factorization in PLASMA

PLASMA + OmpSs

PLASMA + StarPU

Distributed Tasking in PLASMA

Hybrid programming with PaRSEC and OpenMP

Conclusions

Motivation

- heterogeneous computer architectures and imbalances (accelerators, distributed memory, ...)
- increasingly difficult to use the hybrid resources and balance load
- *less control for a programmer — more for a runtime system*

Algorithm as a sequence of tasks

- units of an algorithm as **tasks**
- data dependencies among tasks – directed acyclic graph (DAG)

Runtime systems

- dynamic scheduling of tasks on available cores
- QUARK, OmpSs, StarPU, PaRSEC, OpenMP, ...
- asynchronous execution

Motivation

- heterogeneous computer architectures and imbalances (accelerators, distributed memory, ...)
- increasingly difficult to use the hybrid resources and balance load
- *less control for a programmer — more for a runtime system*

Algorithm as a sequence of tasks

- units of an algorithm as **tasks**
- data dependencies among tasks – directed acyclic graph (DAG)

Runtime systems

- dynamic scheduling of tasks on available cores
- QUARK, OmpSs, StarPU, PaRSEC, OpenMP, . . .
- asynchronous execution

Task-based programming

Motivation

- heterogeneous computer architectures and imbalances (accelerators, distributed memory, ...)
- increasingly difficult to use the hybrid resources and balance load
- *less control for a programmer — more for a runtime system*

Algorithm as a sequence of tasks

- units of an algorithm as **tasks**
- data dependencies among tasks – directed acyclic graph (DAG)

Runtime systems

- dynamic scheduling of tasks on available cores
- QUARK, OmpSs, StarPU, PaRSEC, OpenMP, ...
- asynchronous execution

Outline

Parallel Linear Algebra Software for Multicore Architectures

- evolution of dense linear algebra libraries
- shared memory multicore and manycore environment
- tile-based algorithms
- task-based implementation
- dynamic scheduling of asynchronous kernels

Underlying runtime system

- version 2.8.0 (December 2015) — QUARK (QUeuing And Runtime for Kernels)
- version 17.1 (January 2017) — **OpenMP v4.5**

A. Buttari, J. Langou, J. Kurzak, J. Dongarra.
A class of parallel tiled linear algebra algorithms for multicore architectures.
Parallel Computing 35, 38–53, 2009

PLASMA library

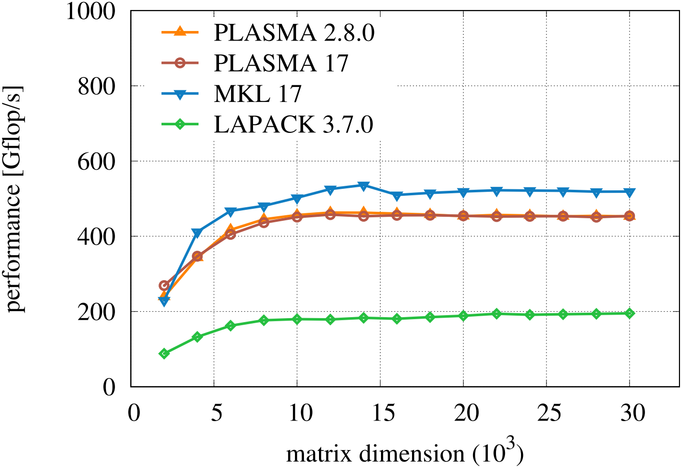
Parallel Linear Algebra Software for Multicore Architectures

- evolution of dense linear algebra libraries
- shared memory multicore and manycore environment
- tile-based algorithms
- task-based implementation
- dynamic scheduling of asynchronous kernels

Underlying runtime system

- version 2.8.0 (December 2015) — QUARK (QUeuing And Runtime for Kernels)
- version 17.1 (January 2017) — **OpenMP v4.5**

A. Buttari, J. Langou, J. Kurzak, J. Dongarra.
A class of parallel tiled linear algebra algorithms for multicore architectures.
Parallel Computing 35, 38–53, 2009



QR factorization, 2x Haswell, 24 cores

Porting PLASMA to OpenMP

- Portability – any compiler with OpenMP v4.5 support
- No loss of performance compared to QUARK
- High performance compared to some optimized vendor libraries

A. YarKhan, J. Kurzak, P. Luszczek, and J. Dongarra.
Porting the PLASMA Numerical Library to the OpenMP Standard.
International Journal of Parallel Programming, 2016.

J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, P. Wu, I. Yamazaki, A. YarKhan, M. Abalenkovs, N. Bagherpour, S. Hammarling, J. Šístek, D. Stevens, M. Zounon, and S.D. Relton.
PLASMA: Parallel linear algebra software for multicore using OpenMP.
To appear in *ACM Transactions on Mathematical Software*.

Outline

Introduction and motivation

PLASMA library and OpenMP

QR factorization in PLASMA

PLASMA + OmpSs

PLASMA + StarPU

Distributed Tasking in PLASMA

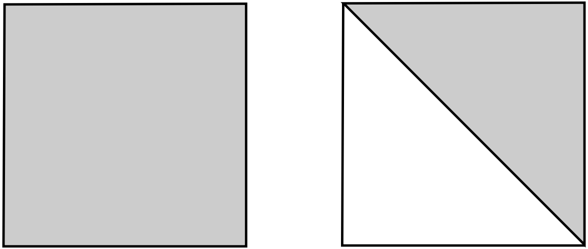
Hybrid programming with PaRSEC and OpenMP

Conclusions

Matrix factorization

$$A = QR$$

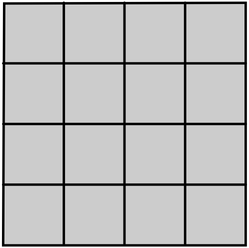
- A — general matrix
- Q — orthogonal matrix, $Q^T Q = I$
- R — upper trapezoidal

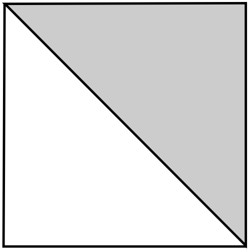


Matrix factorization

$$A = QR$$

- A — general matrix
- Q — orthogonal matrix, $Q^T Q = I$
- R — upper trapezoidal





Sequential kernels

QR decomposition of a tile

$$A_1 = Q_1 R_1$$

- `GEQRT`

Application of Q^T

$$Q_1^T X = Y$$

- `DORMQR`

QR of Triangle-on-top-of-Square

$$\begin{bmatrix} R_1 \\ A_2 \end{bmatrix} = Q_2 \begin{bmatrix} R_{12} \\ \end{bmatrix}$$

- `TSQRT`

Application of a two-tile matrix Q

$$Q_2^T \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

- `TSMQR`

A. Abdelfattah, H. Anzt, J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, I. Yamazaki, and A. YarKhan
Linear algebra software for large-scale accelerated multicore computing.
Acta Numerica 25:1–160, 2016

QR decomposition of a tile

$$A_1 = Q_1 R_1$$

- `GEQRT`

Application of Q^T

$$Q_1^T X = Y$$

- `DORMQR`

QR of Triangle-on-top-of-Square

$$\begin{bmatrix} R_1 \\ A_2 \end{bmatrix} = Q_2 \begin{bmatrix} R_{12} \\ \end{bmatrix}$$

- `TSQRT`

Application of a two-tile matrix Q

$$Q_2^T \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

- `TSMQR`

A. Abdelfattah, H. Anzt, J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, I. Yamazaki, and A. YarKhan
Linear algebra software for large-scale accelerated multicore computing.
Acta Numerica 25:1–160, 2016

QR decomposition of a tile

$$A_1 = Q_1 R_1$$

- GEQRT

Application of Q^T

$$Q_1^T X = Y$$

- DORMQR

QR of Triangle-on-top-of-Square

$$\begin{bmatrix} R_1 \\ A_2 \end{bmatrix} = Q_2 \begin{bmatrix} R_{12} \\ \end{bmatrix}$$

- TSQRT

Application of a two-tile matrix Q

$$Q_2^T \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

- TSMQR

A. Abdelfattah, H. Anzt, J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, I. Yamazaki, and A. YarKhan
Linear algebra software for large-scale accelerated multicore computing.
Acta Numerica 25:1–160, 2016

Sequential kernels

QR decomposition of a tile

$$A_1 = Q_1 R_1$$

- GEQRT

Application of Q^T

$$Q_1^T X = Y$$

- DORMQR

QR of Triangle-on-top-of-Square

$$\begin{bmatrix} R_1 \\ A_2 \end{bmatrix} = Q_2 \begin{bmatrix} R_{12} \\ \\ \end{bmatrix}$$

- TSQRT

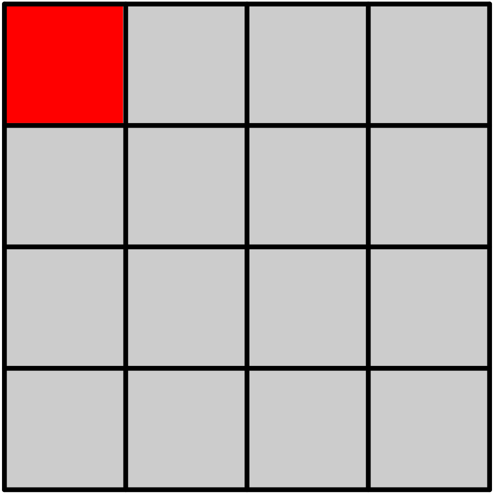
Application of a two-tile matrix Q

$$Q_2^T \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

- TSMQR

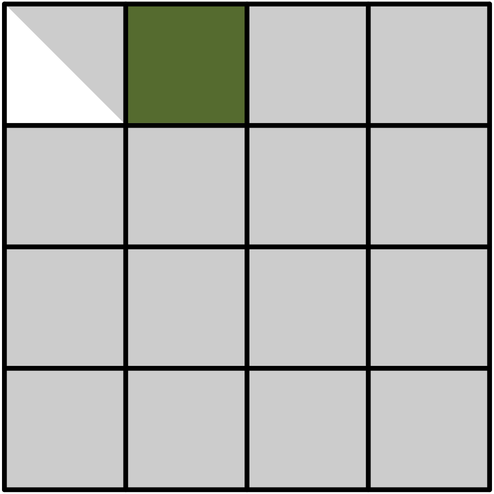
A. Abdelfattah, H. Anzt, J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, I. Yamazaki, and A. YarKhan
Linear algebra software for large-scale accelerated multicore computing.
Acta Numerica 25:1–160, 2016

Tile QR factorization



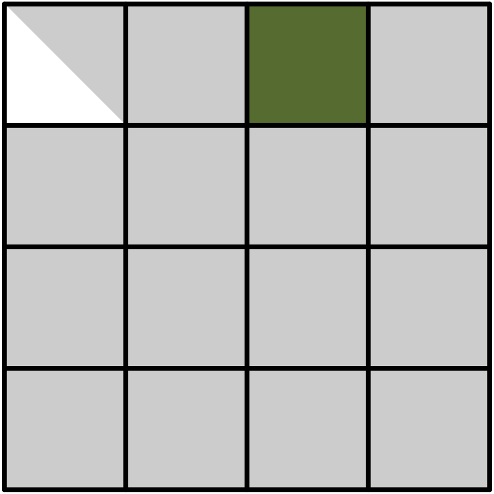
GEQRT

Tile QR factorization



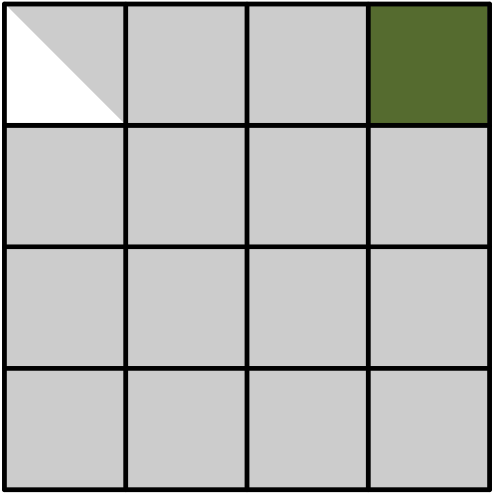
DORMQR

Tile QR factorization



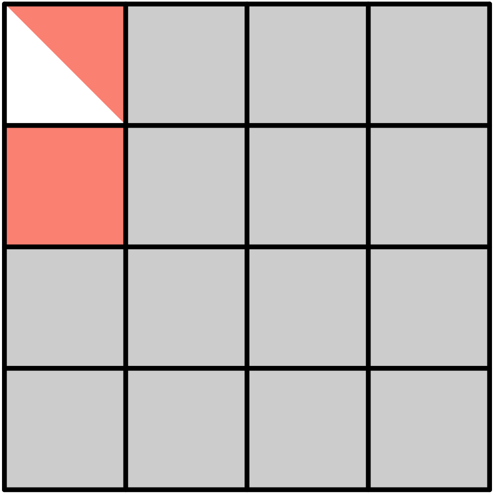
DORMQR

Tile QR factorization



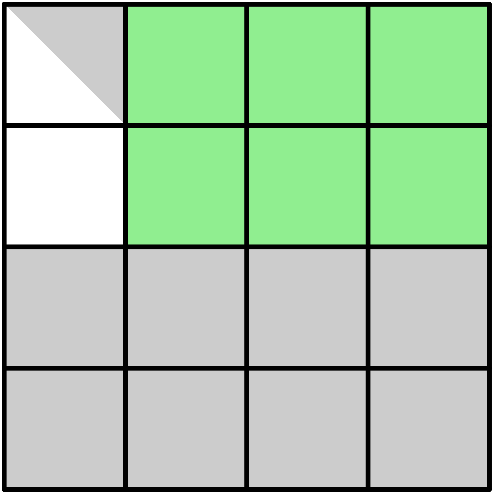
DORMQR

Tile QR factorization



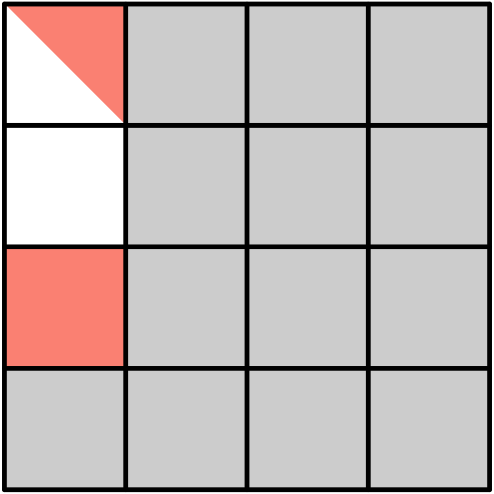
TSQRT

Tile QR factorization



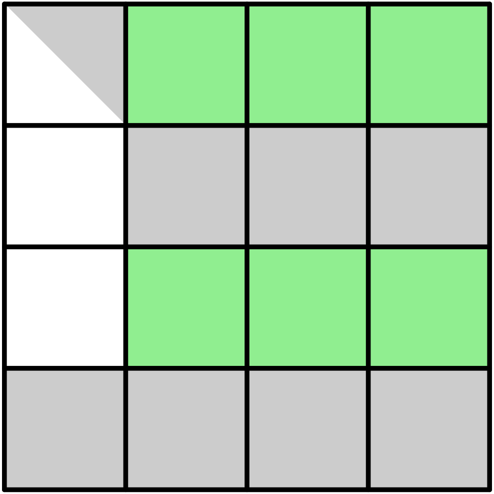
TSMQR

Tile QR factorization

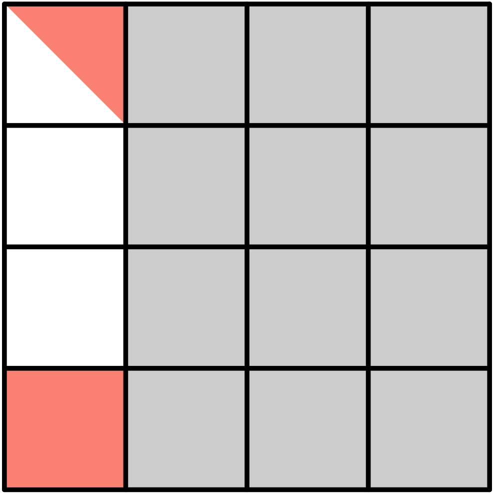


TSQRT

Tile QR factorization

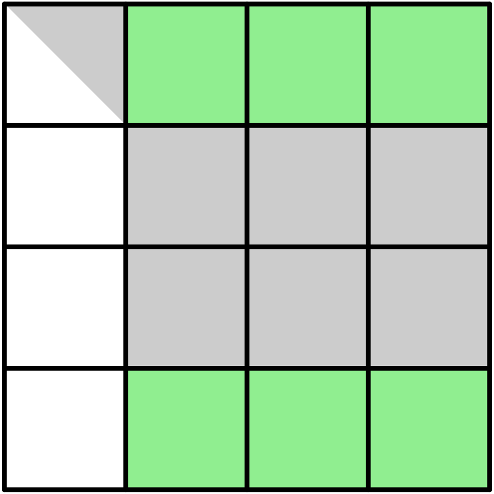


TSMQR



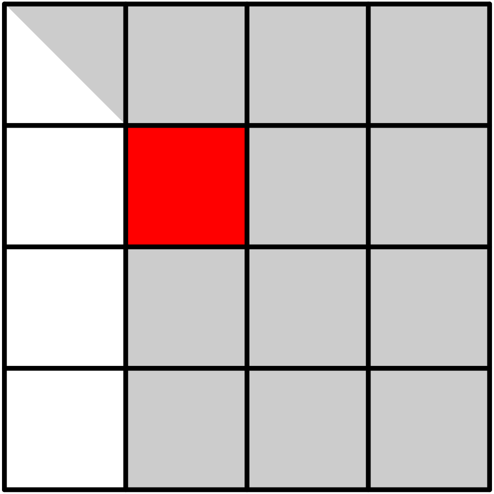
TSQRT

Tile QR factorization



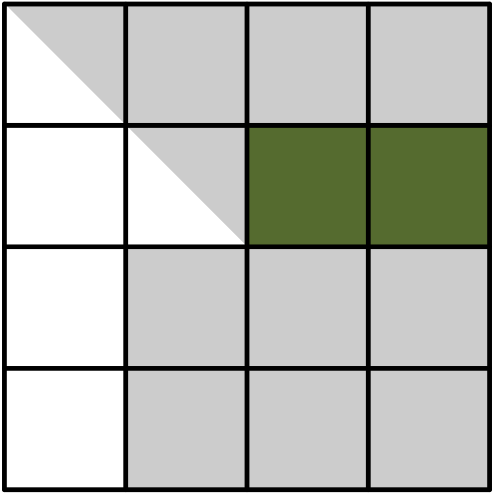
TSMQR

Tile QR factorization



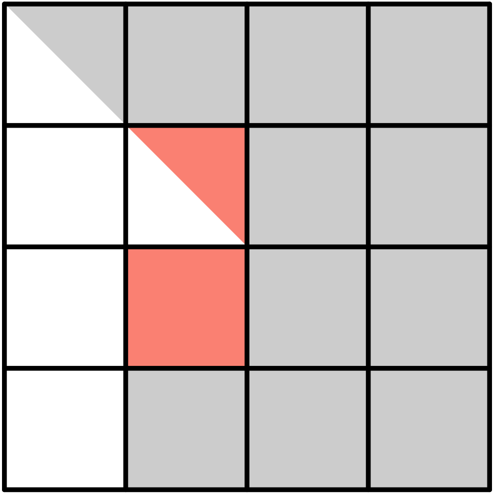
GEQRT

Tile QR factorization



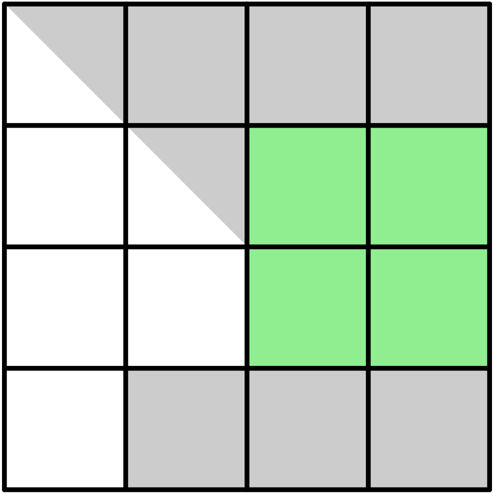
DORMQR

Tile QR factorization



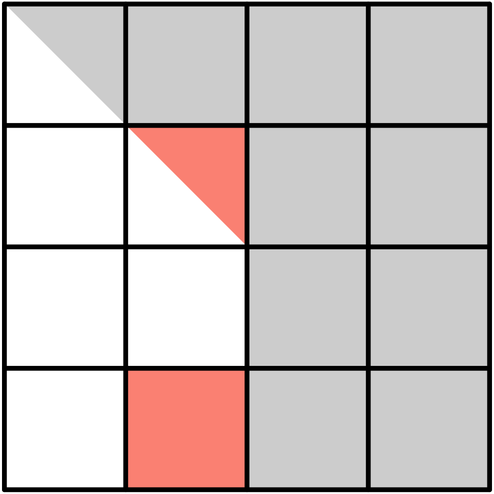
TSQRT

Tile QR factorization



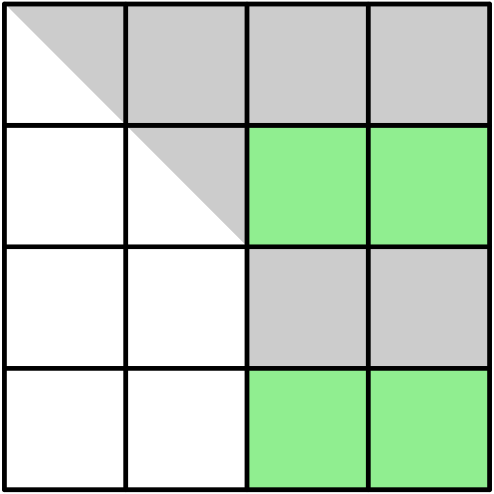
TSMQR

Tile QR factorization



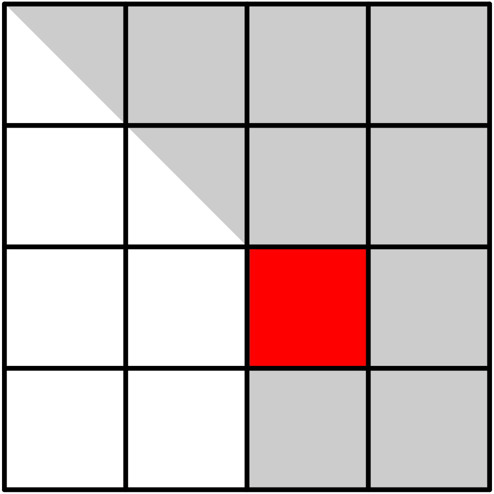
TSQRT

Tile QR factorization



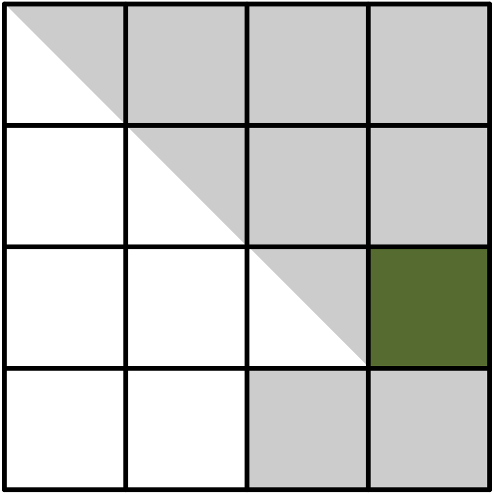
TSMQR

Tile QR factorization



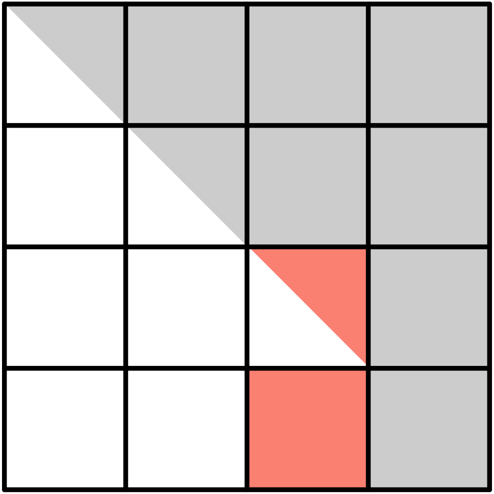
GEQRT

Tile QR factorization

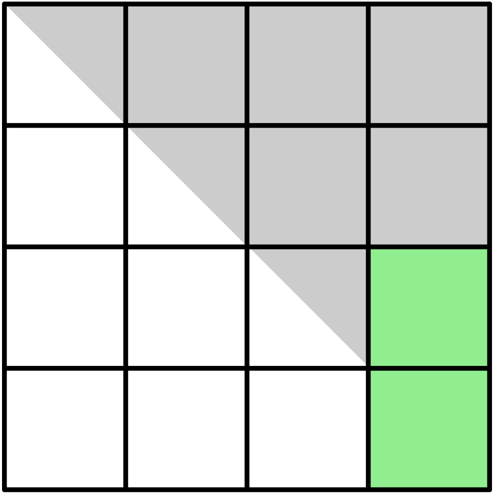


DORMQR

Tile QR factorization

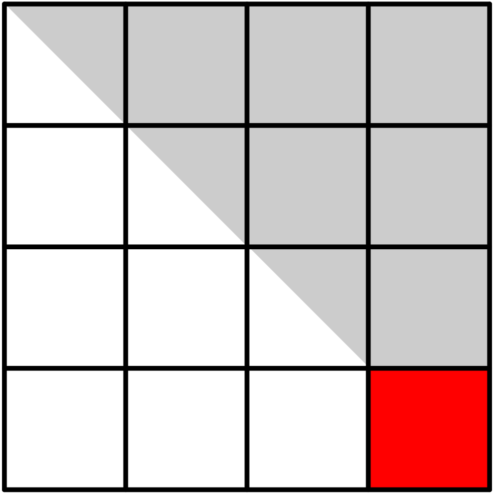


TSQRT



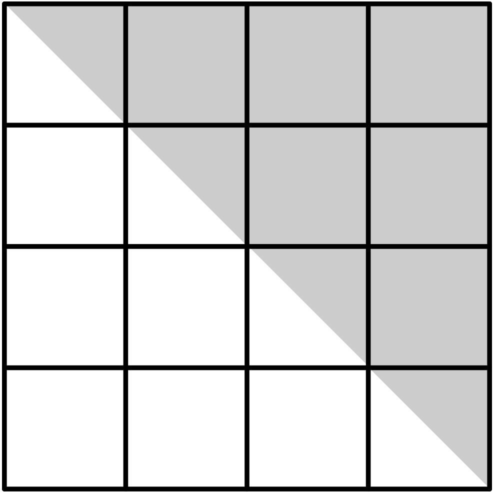
TSMQR

Tile QR factorization



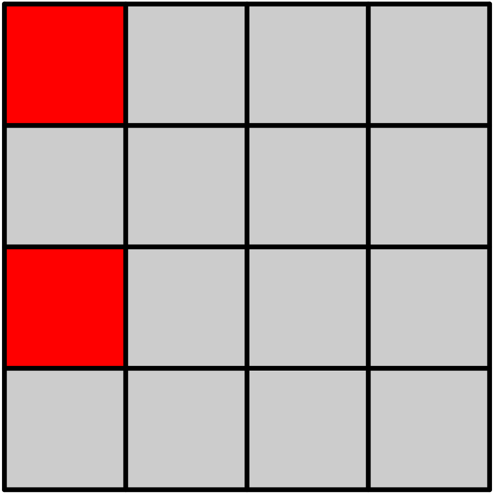
GEQRT

Tile QR factorization



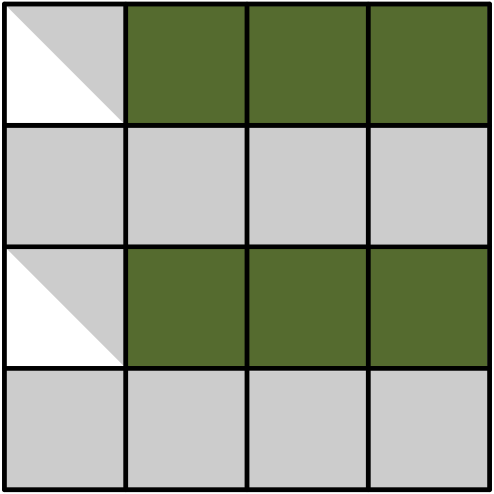
Final form

Tile QR factorization — tree-based



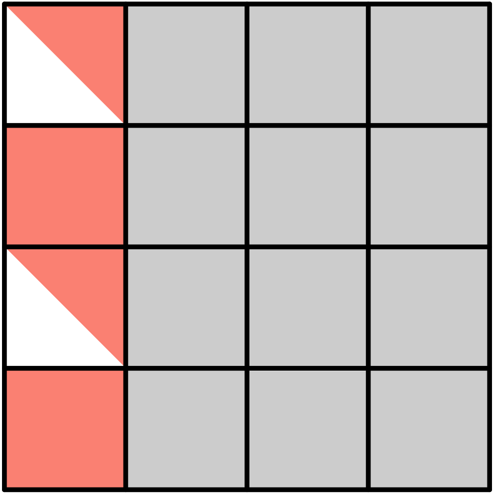
GEQRT

Tile QR factorization — tree-based



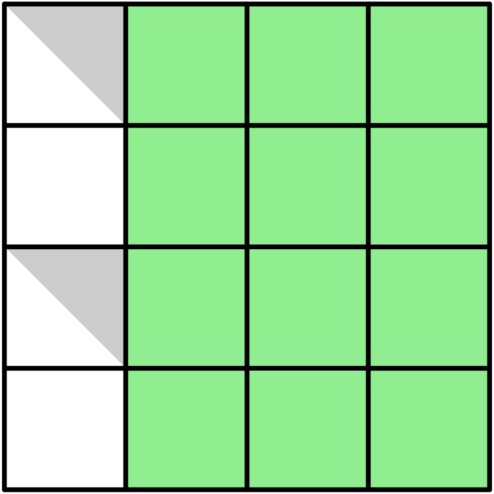
DORMQR

Tile QR factorization — tree-based



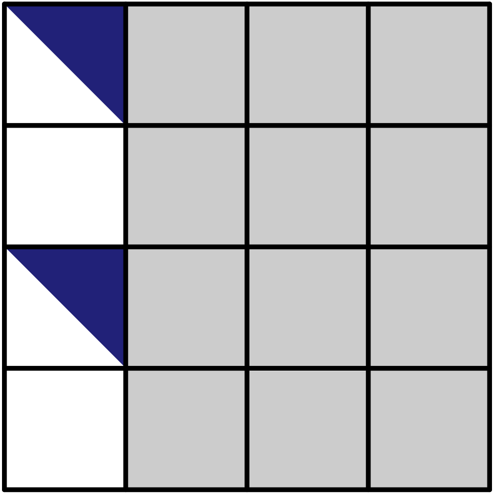
TSQRT

Tile QR factorization — tree-based



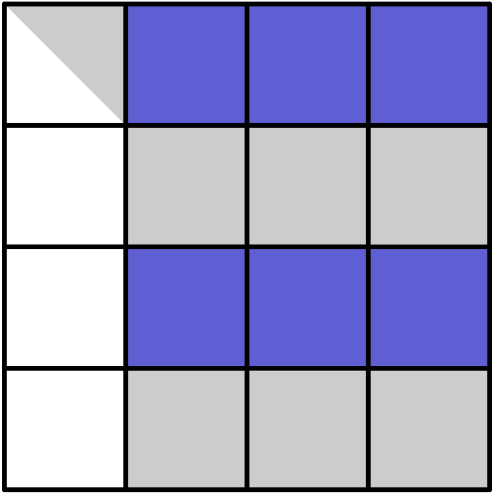
TSMQR

Tile QR factorization — tree-based



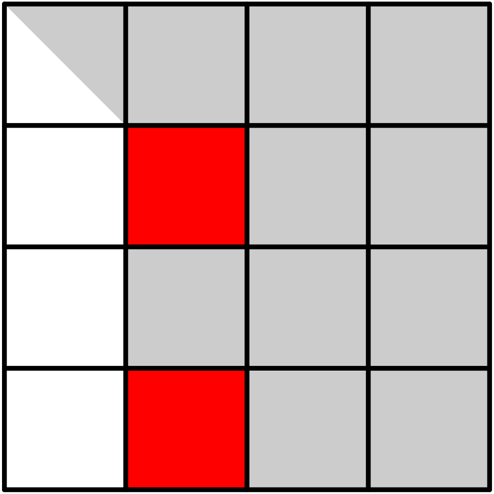
TTQRT

Tile QR factorization — tree-based



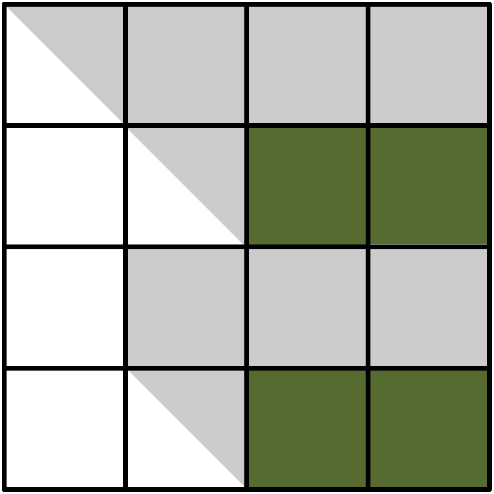
TTMQR

Tile QR factorization — tree-based



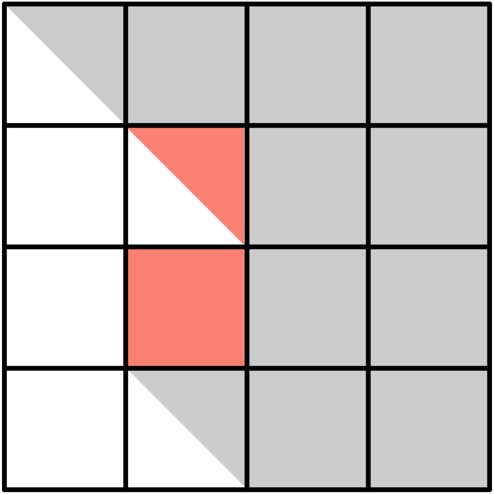
GEQRT

Tile QR factorization — tree-based



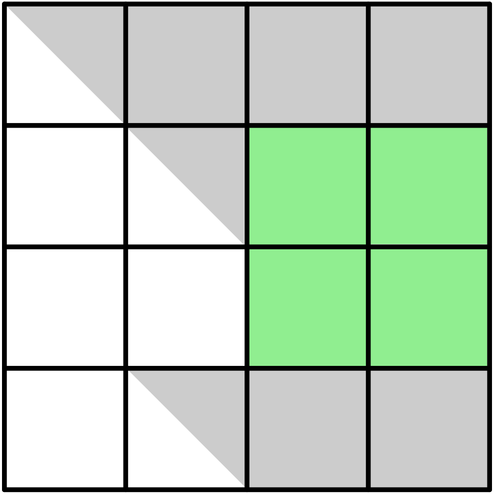
DORMQR

Tile QR factorization — tree-based



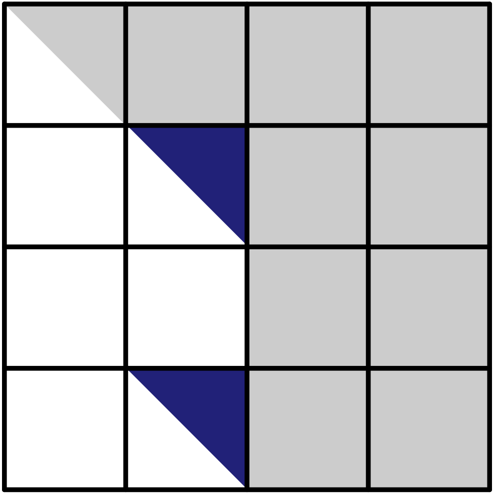
TSQRT

Tile QR factorization — tree-based



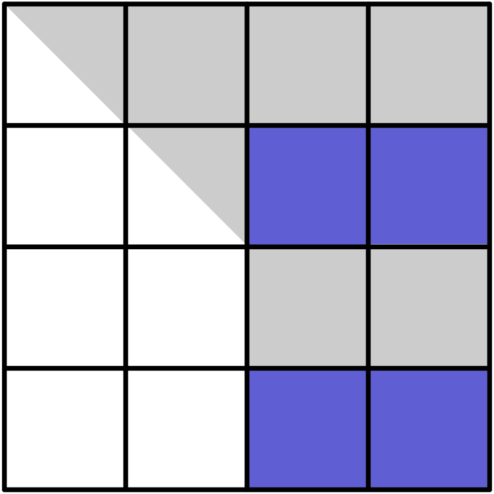
TSMQR

Tile QR factorization — tree-based



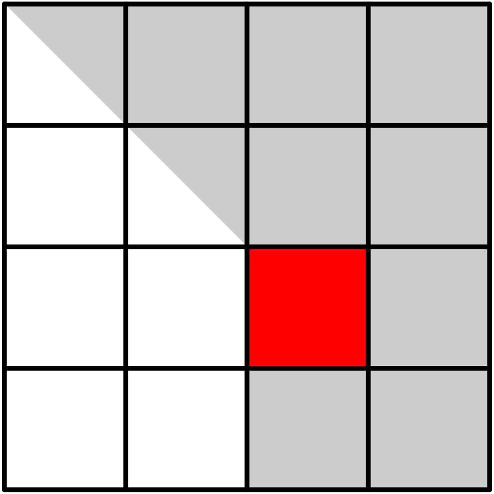
$TTQRT$

Tile QR factorization — tree-based



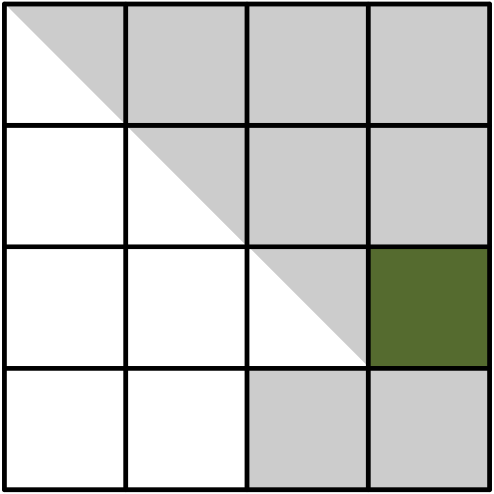
TTMQR

Tile QR factorization — tree-based



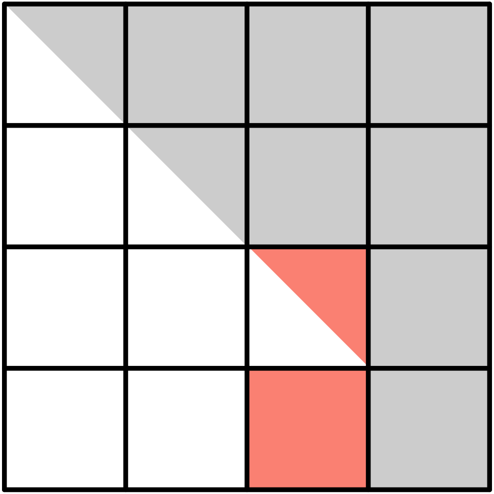
GEQRT

Tile QR factorization — tree-based

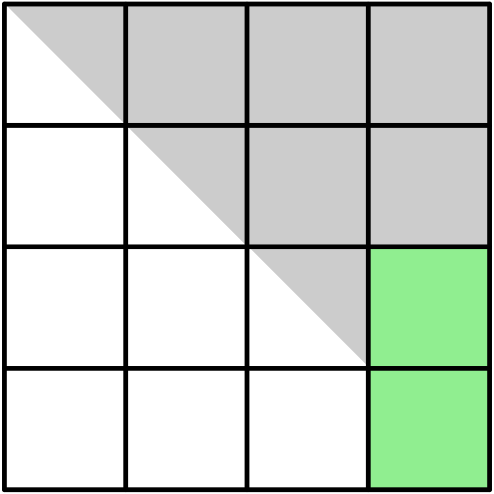


DORMQR

Tile QR factorization — tree-based

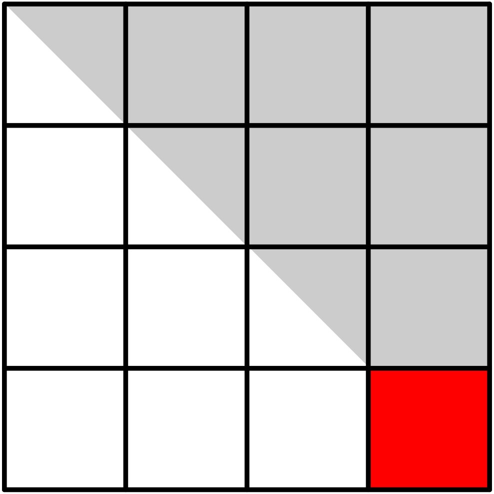


TSQRT



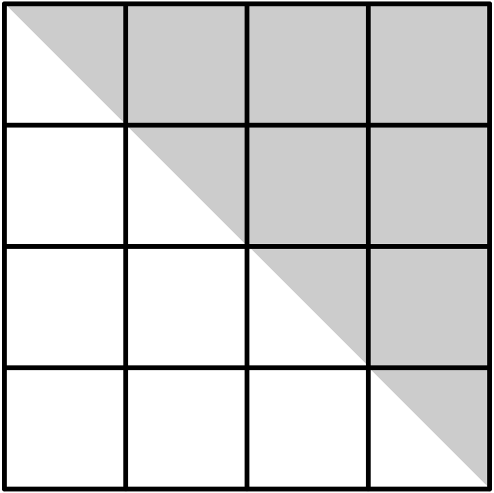
TSMQR

Tile QR factorization — tree-based



GEQRT

Tile QR factorization — tree-based



Final form

The Triangle-on-top-of-Triangle (TT) kernels

QR for a two-triangle matrix

$$\begin{bmatrix} R_1 \\ R_2 \end{bmatrix} = Q_{12} \begin{bmatrix} R_{12} \\ \end{bmatrix}$$

- `TTQRT`

Application of a two-tile matrix Q

$$Q_{12}^T \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

- `TTMQR`

J. W. Demmel, L. Grigori, M. F. Hoemmen, and J. Langou. Communication-optimal parallel and sequential QR and LU factorizations. Technical Report 204, LAPACK Working Note, August 2008.

The Triangle-on-top-of-Triangle (TT) kernels

QR for a two-triangle matrix

$$\begin{bmatrix} R_1 \\ R_2 \end{bmatrix} = Q_{12} \begin{bmatrix} R_{12} \\ \end{bmatrix}$$

- TTQRT

Application of a two-tile matrix Q

$$Q_{12}^T \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

- TTMQR

J. W. Demmel, L. Grigori, M. F. Hoemmen, and J. Langou. Communication-optimal parallel and sequential QR and LU factorizations. Technical Report 204, LAPACK Working Note, August 2008.

Standard tile QR factorization

- like block Householder reflections
- parallelism from update
- minimal number of visits of tiles (data locality)

Tree-based QR factorization

- like block Givens rotations
- better parallelism
- some tiles visited more times (worse data locality)

J. Dongarra, M. Faverge, T. Herault, M. Jacquelin, J. Langou, and Y. Robert.
Hierarchical QR factorization algorithms for multi-core clusters.
Parallel Computing, 39(4–5):212–232, 2013.

Standard tile QR factorization

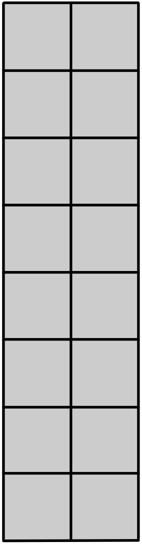
- like block Householder reflections
- parallelism from update
- minimal number of visits of tiles (data locality)

Tree-based QR factorization

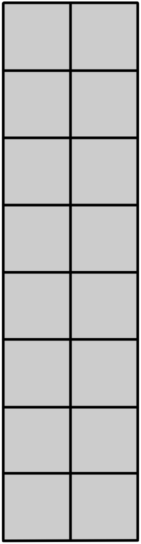
- like block Givens rotations
- better parallelism
- some tiles visited more times (worse data locality)

J. Dongarra, M. Faverge, T. Herault, M. Jacquelin, J. Langou, and Y. Robert.
Hierarchical QR factorization algorithms for multi-core clusters.
Parallel Computing, 39(4–5):212–232, 2013.

Order of tile eliminations

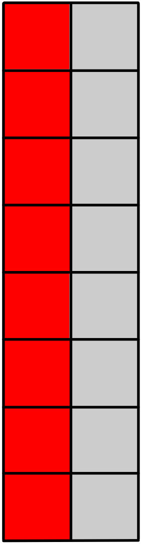


Binary tree pairing

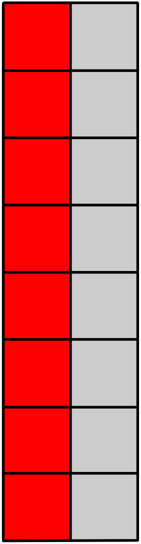


Greedy pairing

Order of tile eliminations

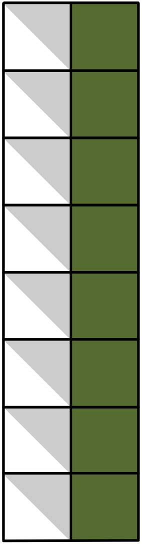


Binary tree pairing

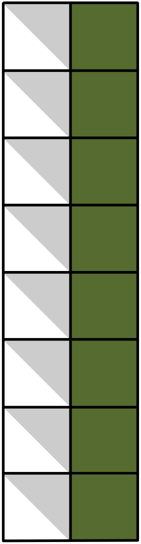


Greedy pairing

Order of tile eliminations

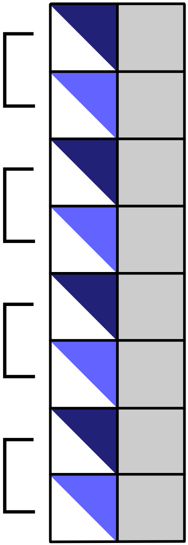


Binary tree pairing

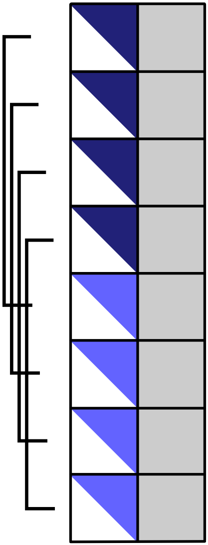


Greedy pairing

Order of tile eliminations

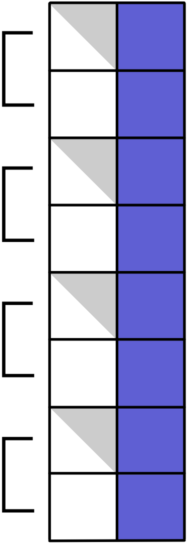


Binary tree pairing

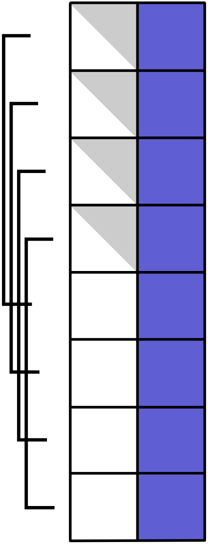


Greedy pairing

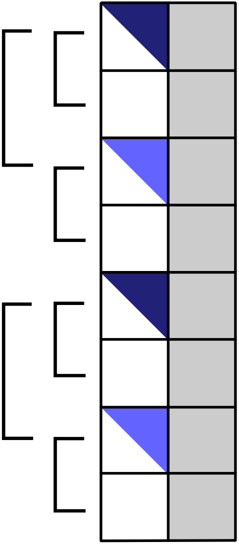
Order of tile eliminations



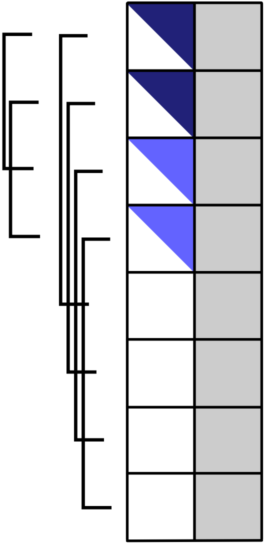
Binary tree pairing



Greedy pairing

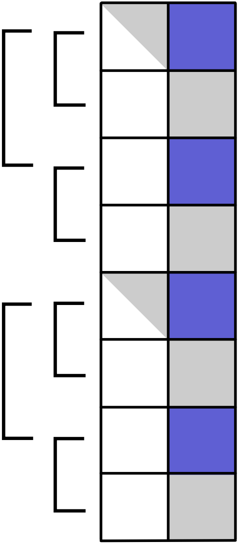


Binary tree pairing

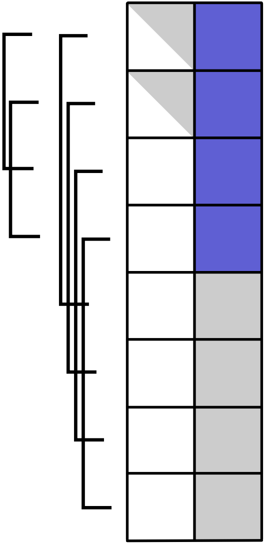


Greedy pairing

Order of tile eliminations

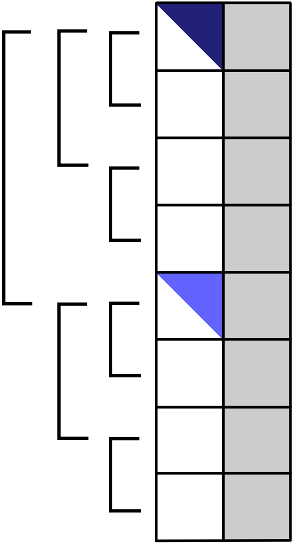


Binary tree pairing

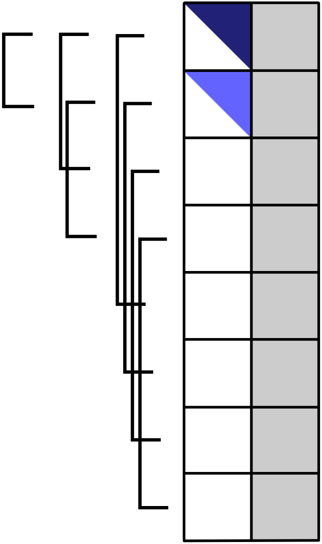


Greedy pairing

Order of tile eliminations

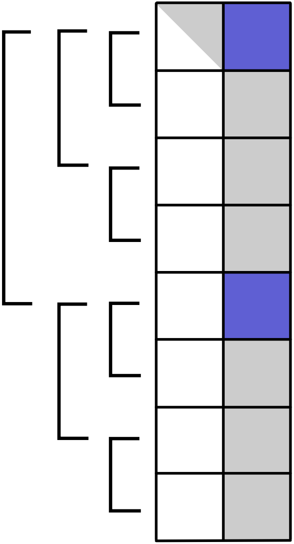


Binary tree pairing

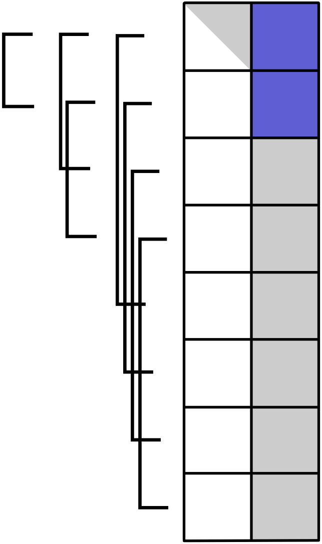


Greedy pairing

Order of tile eliminations

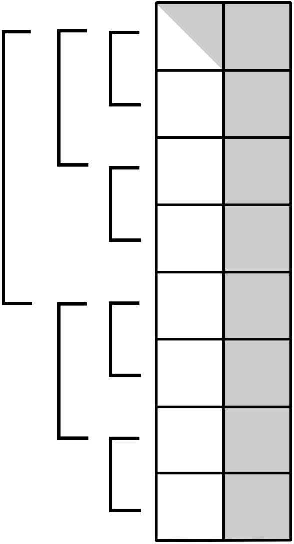


Binary tree pairing

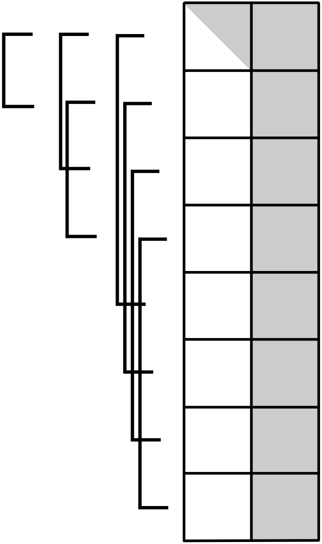


Greedy pairing

Order of tile eliminations



Binary tree pairing



Greedy pairing

Order of tile eliminations

Binary tree

- organises eliminations within the panel in a binary-tree fashion

Greedy algorithm

- organises eliminations within the panel from the bottom upwards
- enables parallelism across tile columns

H. Bouwmeester, M. Jacquelin, J. Langou and Y. Robert. Tiled QR factorization algorithms. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Seatle, WA, 2011, pp. 1-11.

Order of tile eliminations

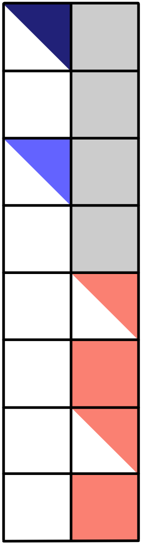
Binary tree

- organises eliminations within the panel in a binary-tree fashion

Greedy algorithm

- organises eliminations within the panel from the bottom upwards
- enables parallelism across tile columns

H. Bouwmeester, M. Jacquelin, J. Langou and Y. Robert. Tiled QR factorization algorithms. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Seatle, WA, 2011, pp. 1-11.



Block greedy algorithm

- greedy algorithm on top blocks eliminated with TS kernels

Adjusting the size of the flat trees

$$bs = \frac{m_t\left(n_t^2/2 + n_t/2\right)}{4\,p}$$

- bs . . . block size of the flat trees
- m_t . . . number of tile rows
- n_t . . . number of tile columns
- p . . . hardware parallelism (number of OpenMP threads)

Inspired by

M. Faverge, J. Langou, Y. Robert, J. Dongarra. Bidiagonalization and R-Bidiagonalization: Parallel Tiled Algorithms, Critical Paths and Distributed-Memory Implementation. IPDPS'17 - 31st IEEE International Parallel and Distributed Processing Symposium, May 2017, Orlando, United States, 2017.

$$A = QR$$

Archer

- 2 x Intel Xeon CPU E5-2697 v2 @ 2.70GHz
- 2 x 12 cores
- 64GB RAM
- theoretical peak performance 518 Gflop/s

Intel Knights Landing (KNL)

- Intel Xeon Phi 7250
- 68 cores
- 16GB MCDRAM
- theoretical peak performance 3046 Gflop/s

$$A = QR$$

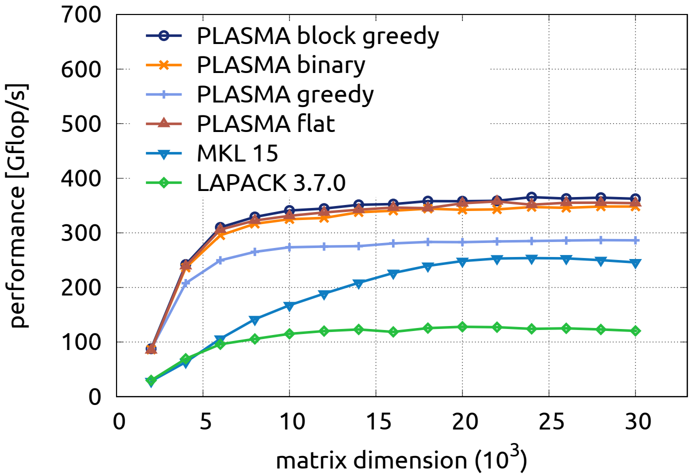
Archer

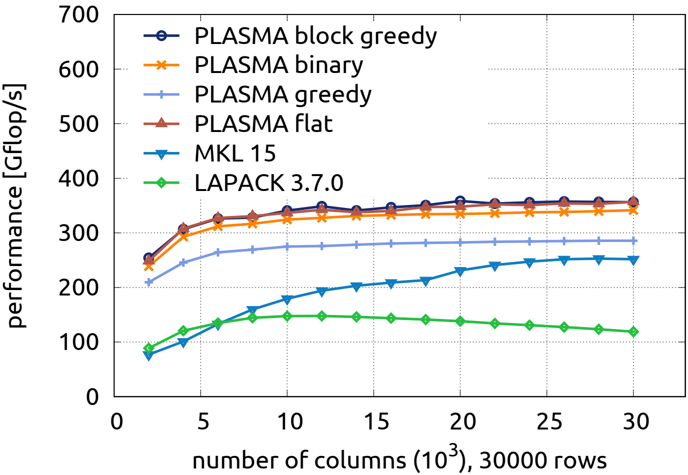
- 2 x Intel Xeon CPU E5-2697 v2 @ 2.70GHz
- 2 x 12 cores
- 64GB RAM
- theoretical peak performance 518 Gflop/s

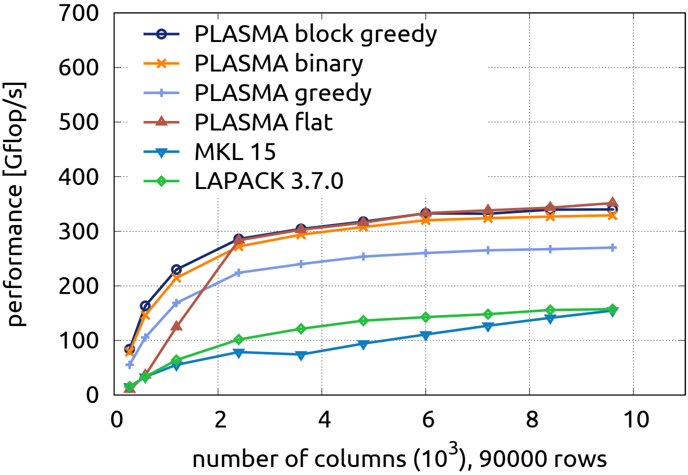
Intel Knights Landing (KNL)

- Intel Xeon Phi 7250
- 68 cores
- 16GB MCDRAM
- theoretical peak performance 3046 Gflop/s

QR factorization — square matrices, Archer

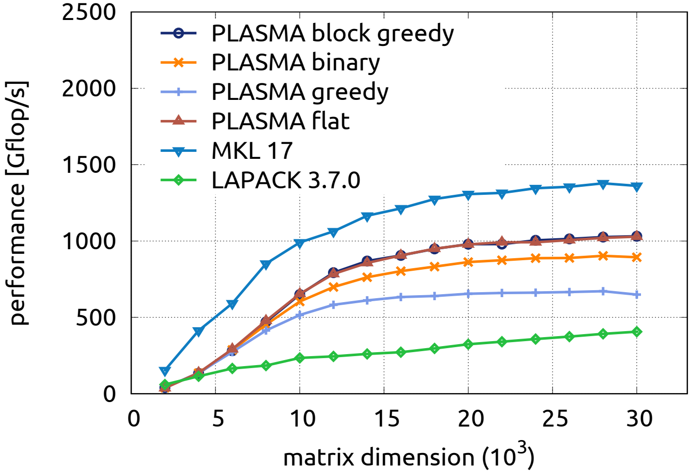



PLASMA block greedy
PLASMA binary
PLASMA greedy
PLASMA flat
MKL 15
LAPACK 3.7.0
performance [Gflop/s]
number of columns (10^3), 30000 rows
0
100
200
300
400
500
600
700
0
5
10
15
20
25
30

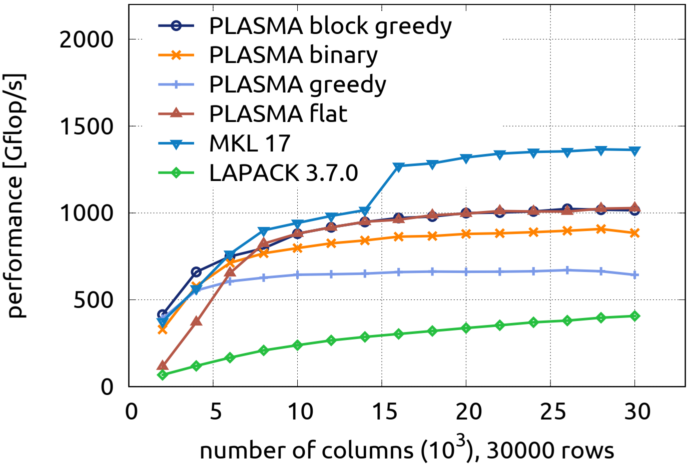


PLASMA block greedy
PLASMA binary
PLASMA greedy
PLASMA flat
MKL 15
LAPACK 3.7.0
performance [Gflop/s]
number of columns (10^3), 90000 rows
0
100
200
300
400
500
600
700
0
2
4
6
8
10

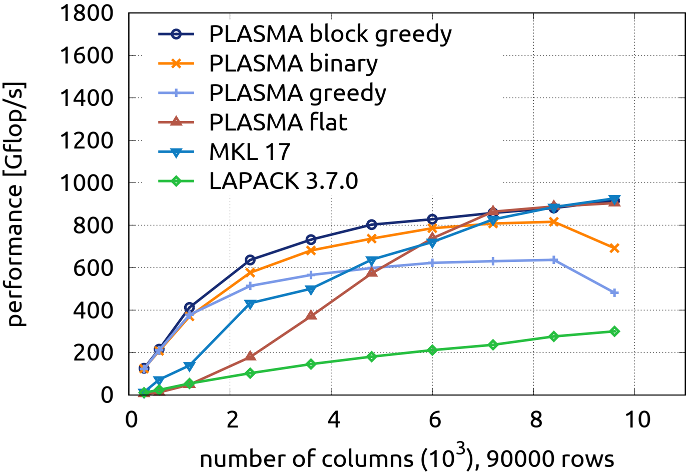
QR factorization — square matrices, KNL



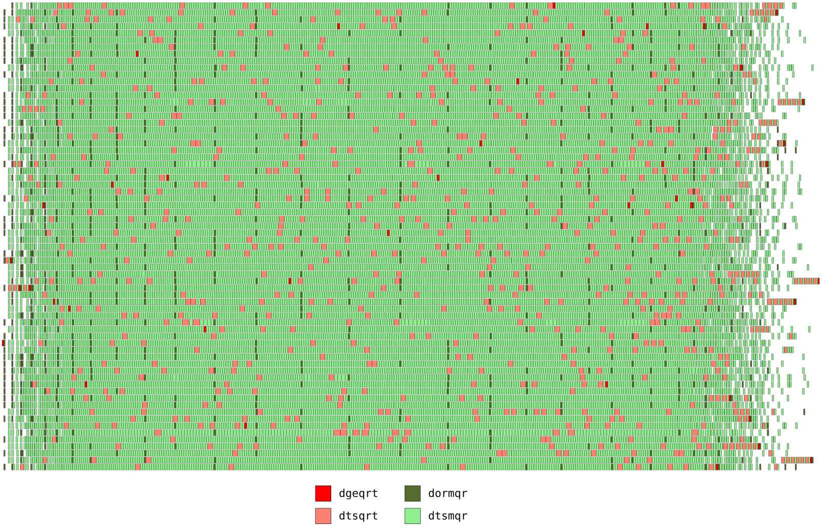
QR factorization — rectangular matrices, KNL



QR factorization — tall matrices, KNL

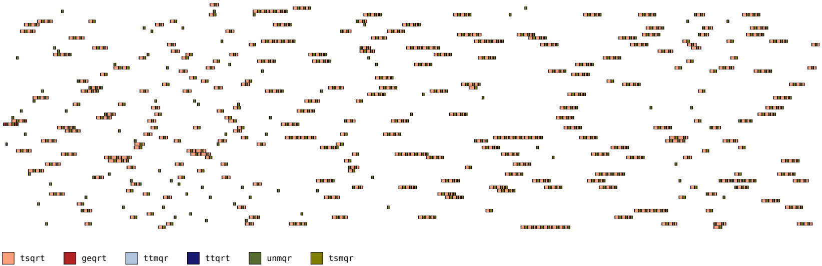


KNL parallel trace – square matrix



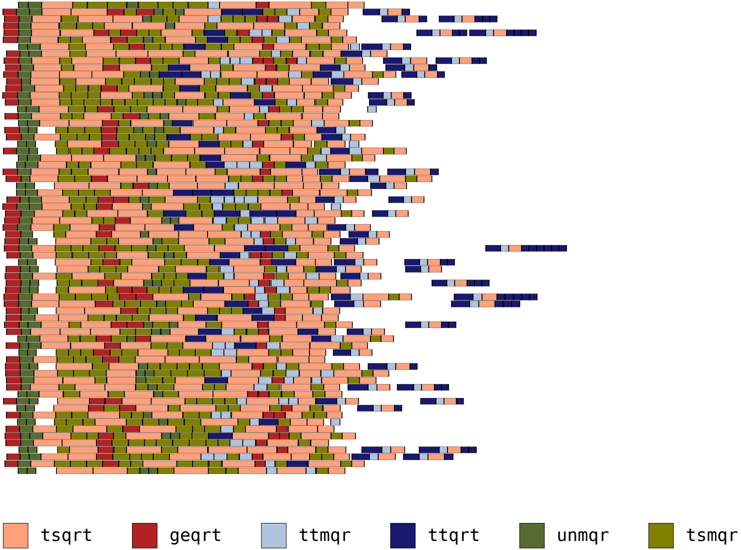
matrix 20 000×20 000, tile size 500×500, 68 cores

KNL parallel trace — flat tree



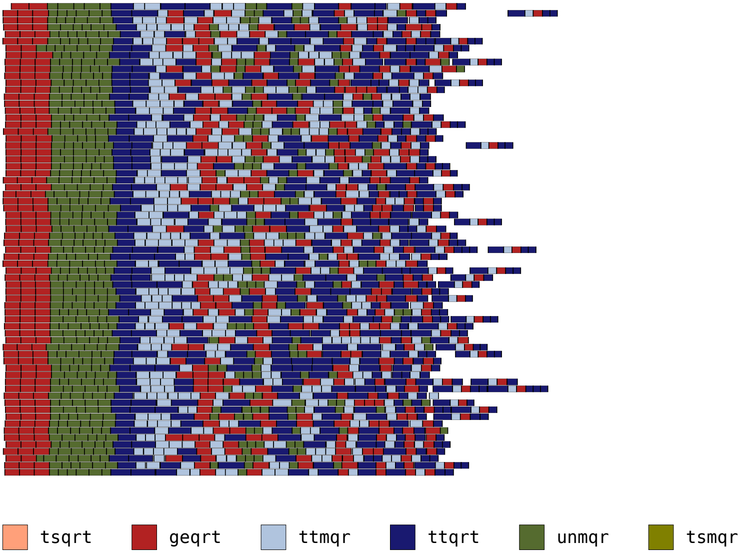
matrix 90 000×1 200, tile size 448, 68 cores

KNL parallel trace — binary tree



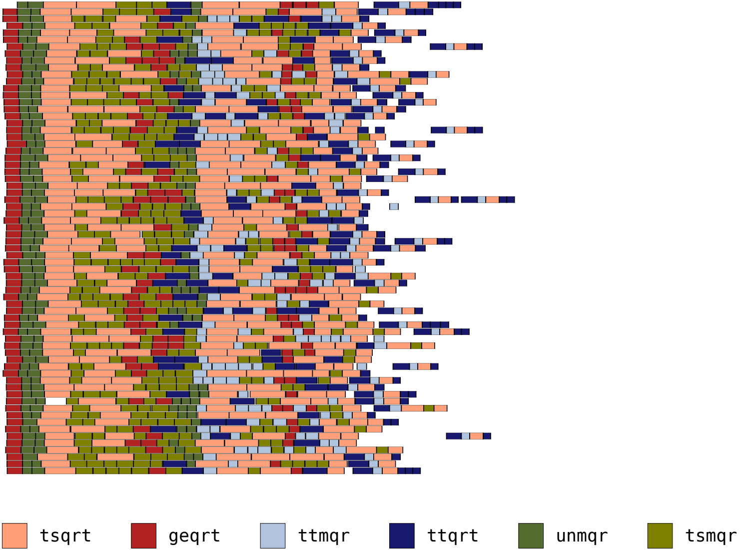
matrix 90 000×1 200, tile size 448, 68 cores

KNL parallel trace — greedy tree



matrix 90 000×1 200, tile size 448, 68 cores

KNL parallel trace — block greedy tree



matrix 90 000×1 200, tile size 448, 68 cores

Outline

Motivation

- allow PLASMA to use INTERTWinE **Resource Manager**
- potential to improve performance

Task in OpenMP

```
void core_omp_zgeqrt(m, n, A, lda, T, ib, ...)
{
    #pragma omp task depend(inout:A[0:lda*n]) \
                     depend(out:T[0:ib*n])
    {
        // Call the sequential kernel.
        int info = core_zgeqrt(A, T, ...);
    }
}
```

Task in OmpSs

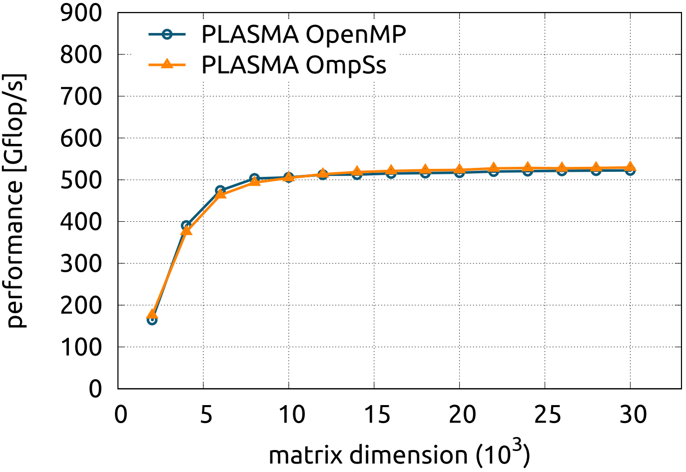
```
void core_ompss_zgeqrt(m, n, A, lda, T, ib, ...)
{
    #pragma oss task inout(A[0;lda*n]) \
                     out(T[0;ib*n]) \
                     no_copy_deps
    {
        // Call the sequential kernel.
        int info = core_zgeqrt(A, T, ...);
    }
}
```

Task in OpenMP

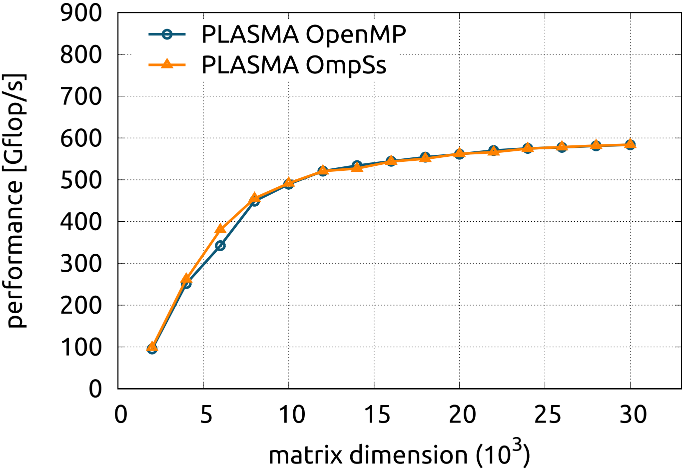
```
void core_omp_zgeqrt(m, n, A, lda, T, ib, ...)
{
    #pragma omp task depend(inout:A[0:lda*n]) \
                     depend(out:T[0:ib*n])
    {
        // Call the sequential kernel.
        int info = core_zgeqrt(A, T, ...);
    }
}
```

Task in OmpSs

```
void core_ompss_zgeqrt(m, n, A, lda, T, ib, ...)
{
    #pragma oss task inout(A[0;lda*n]) \
                     out(T[0;ib*n]) \
                     no_copy_deps
    {
        // Call the sequential kernel.
        int info = core_zgeqrt(A, T, ...);
    }
}
```

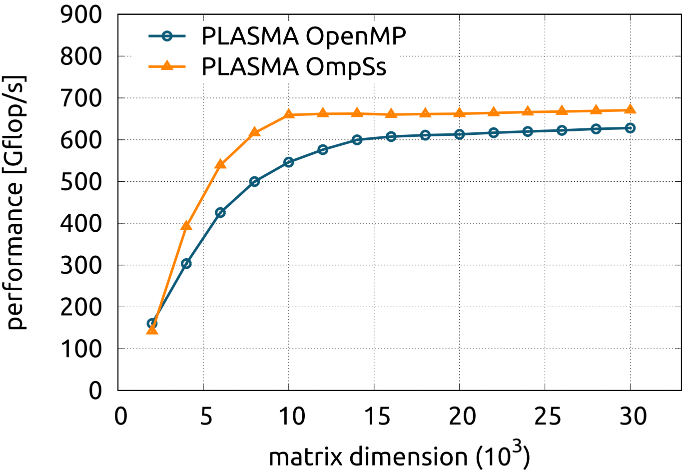


QR factorization, 2x Haswell, 24 cores



LU factorization, 2x Haswell, 24 cores

PLASMA + OmpSs

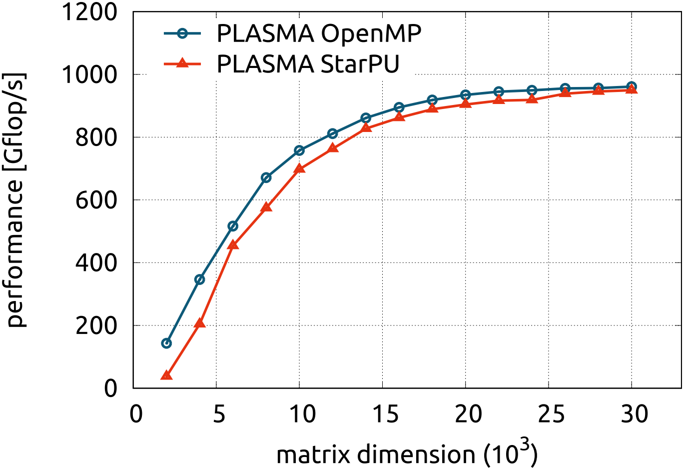


Cholesky factorization, 2x Haswell, 24 cores

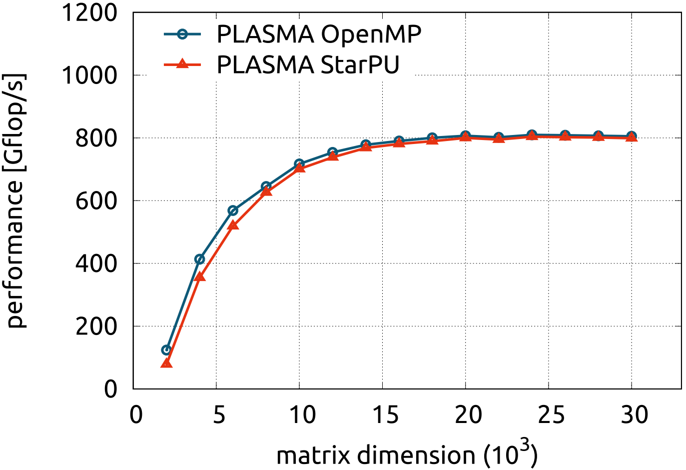
Outline

Motivation

- allow PLASMA to use INTERTWinE **Resource Manager**
- potential to switch to distributed memory
- potential to use GPUs



Cholesky factorization, 2x Broadwell, 28 cores



QR factorization, 2x Broadwell, 28 cores

Outline

Manchester Numerical Analysis

Distributed memory tasking in PLASMA

Motivation

- test the INTERTWinE **Directory/Cache memory manager**
- experience transition from shared memory to distributed memory tasks
- hybrid programming of PaRSEC + OpenMP

Tested runtime systems

- **StarPU**
- **PaRSEC-DTD**

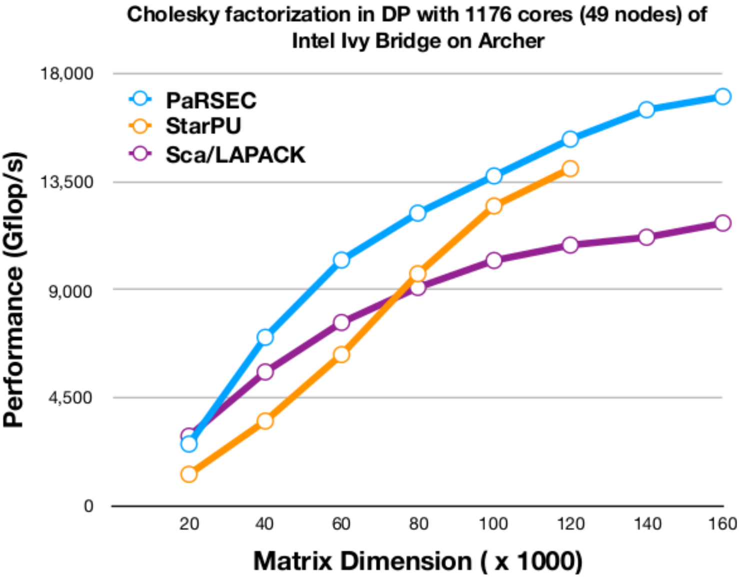
Motivation

- test the INTERTWinE **Directory/Cache memory manager**
- experience transition from shared memory to distributed memory tasks
- hybrid programming of PaRSEC + OpenMP

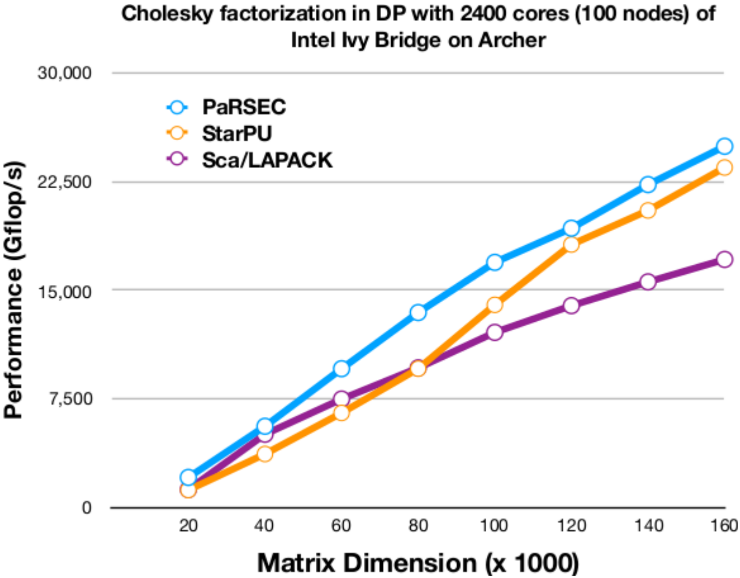
Tested runtime systems

- **StarPU**
- **PaRSEC-DTD**

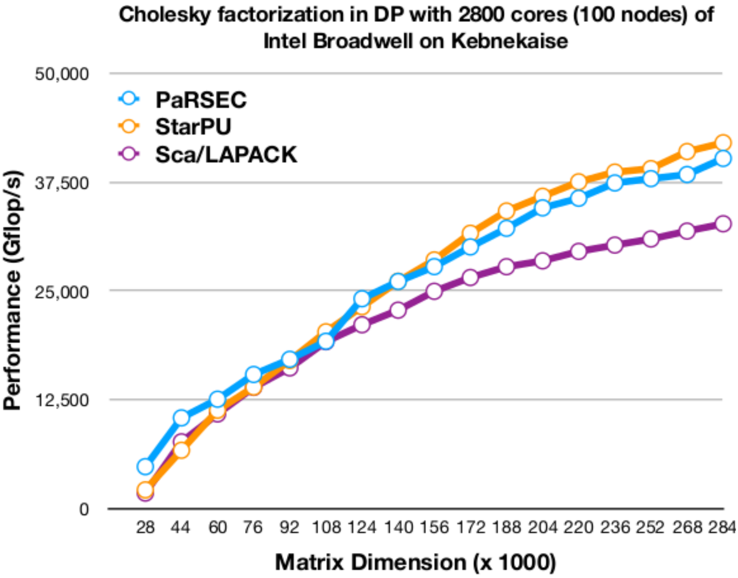
Preliminary results compared to ScaLAPACK



Preliminary results compared to ScaLAPACK



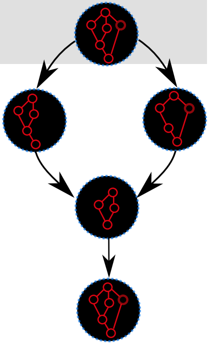
Preliminary results compared to ScaLAPACK

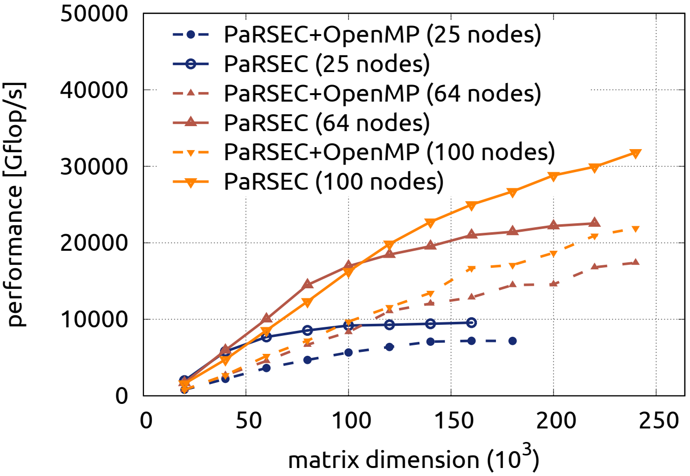


Outline

Motivation

- investigate interoperability of PaRSEC and OpenMP
- running PaRSEC inter-node, OpenMP intra-node





Cholesky factorization, Archer, Haswell nodes, 24 cores each

Outline

Introduction and motivation

PLASMA library and OpenMP

QR factorization in PLASMA

PLASMA + OmpSs

PLASMA + StarPU

Distributed Tasking in PLASMA

Hybrid programming with PaRSEC and OpenMP

Conclusions

Conclusions

QR in PLASMA

- implementation allowing different elimination trees for QR and LQ
- block greedy algorithm
- need to create the right amount of parallelism

Experiments with shared memory

- PLASMA + OpenMP
- PLASMA + OmpSs
- PLASMA + StarPU

Experiments with distributed memory

- PLASMA + StarPU
- PLASMA + PaRSEC-DTD
- hybrid PLASMA + PaRSEC-DTD + OpenMP

Conclusions

QR in PLASMA

- implementation allowing different elimination trees for QR and LQ
- block greedy algorithm
- need to create the right amount of parallelism

Experiments with shared memory

- PLASMA + OpenMP
- PLASMA + OmpSs
- PLASMA + StarPU

Experiments with distributed memory

- PLASMA + StarPU
- PLASMA + PaRSEC-DTD
- hybrid PLASMA + PaRSEC-DTD + OpenMP

Conclusions

QR in PLASMA

- implementation allowing different elimination trees for QR and LQ
- block greedy algorithm
- need to create the right amount of parallelism

Experiments with shared memory

- PLASMA + OpenMP
- PLASMA + OmpSs
- PLASMA + StarPU

Experiments with distributed memory

- PLASMA + StarPU
- PLASMA + PaRSEC-DTD
- hybrid PLASMA + PaRSEC-DTD + OpenMP

Thank you for your attention . . . and collaboration.

Jakub Šístek
sistek@math.cas.cz
http://users.math.cas.cz/~sistek/

PLASMA library
https://bitbucket.org/icl/plasma

INTERTWinE project
Programming Model INTERoperability ToWards Exascale
http://www.intertwine-project.eu

NLAFET project
Numerical Linear Algebra for Future Extreme Scale Systems
http://www.nlafet.eu