Communication Avoiding Algorithms in Plasma and Magma

Bilel Hadri, Hatem Ltaief, Emmanuel Agullo, Fengguang Song, and Jack Dongarra

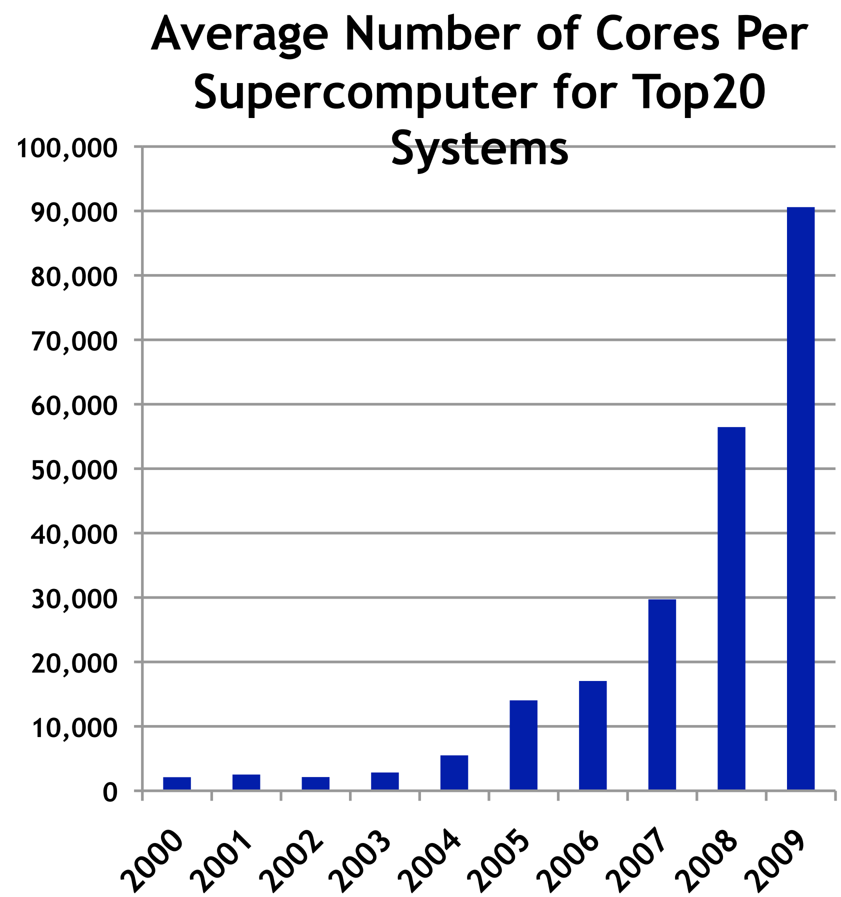
University of Tennessee
Oak Ridge National Laboratory

Potential System Architecture

Systems	2009	2018	Difference Today & 2018
System peak	**2 Pflop/s**	**1 Eflop/s**	**O(1000)**
Power	**6 MW**	**~20 MW**	
System memory	0.3 PB	32 - 64 PB **[.03 Bytes/Flop]**	O(100)
Node performance	125 GF	1,2 or 15TF	O(10) – O(100)
Node memory BW	25 GB/s	2 - 4TB/s **[.002 Bytes/Flop]**	O(100)
Node concurrency	12	O(1k) or 10k	O(100) – O(1000)
Total Node Interconnect BW	3.5 GB/s	200-400GB/s (1:4 or 1:8 from memory BW)	O(100)
System size (nodes)	18,700	O(100,000) or O(1M)	O(10) – O(100)
Total concurrency	225,000	O(billion) [O(10) to O(100) for latency hiding]	O(10,000)
Storage	15 PB	500-1000 PB (>10x system memory is min)	O(10) – O(100)
IO	0.2 TB	60 TB/s (how long to drain the machine)	O(100)
MTTI	days	O(1 day)	- O(10)

Factors that Necessitate Redesign of Our Software

- Steepness of the ascent from terascale to petascale to exascale
- Extreme parallelism and hybrid design
  - Preparing for million/billion way parallelism
- Tightening memory/bandwidth bottleneck
  - Limits on power/clock speed implication on multicore
  - Reducing communication will become much more intense
  - Memory per core changes, byte-to-flop ratio will change
- Necessary Fault Tolerance
  - MTTF will drop
  - Checkpoint/restart has limitations



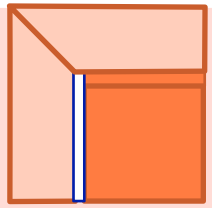
Software infrastructure does not exist today

Major Changes to Software

- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with cluster computing and message passing
  - Rethink and rewrite the applications, algorithms, and software
- Numerical libraries for example will change
  - For example, both LAPACK and ScaLAPACK will undergo major changes to accommodate this

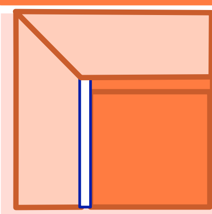
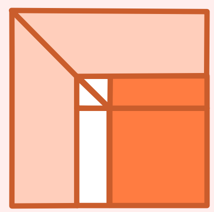
A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations

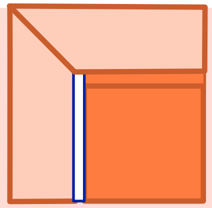
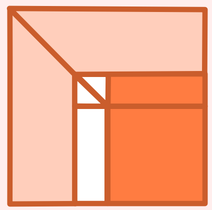
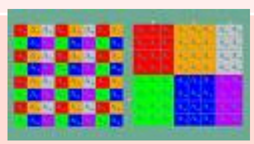
A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations

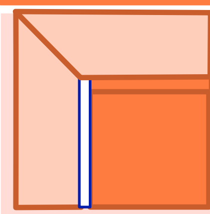
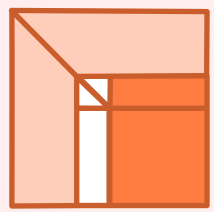
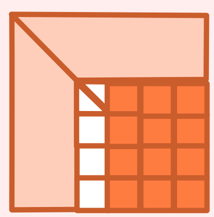
A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing

A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

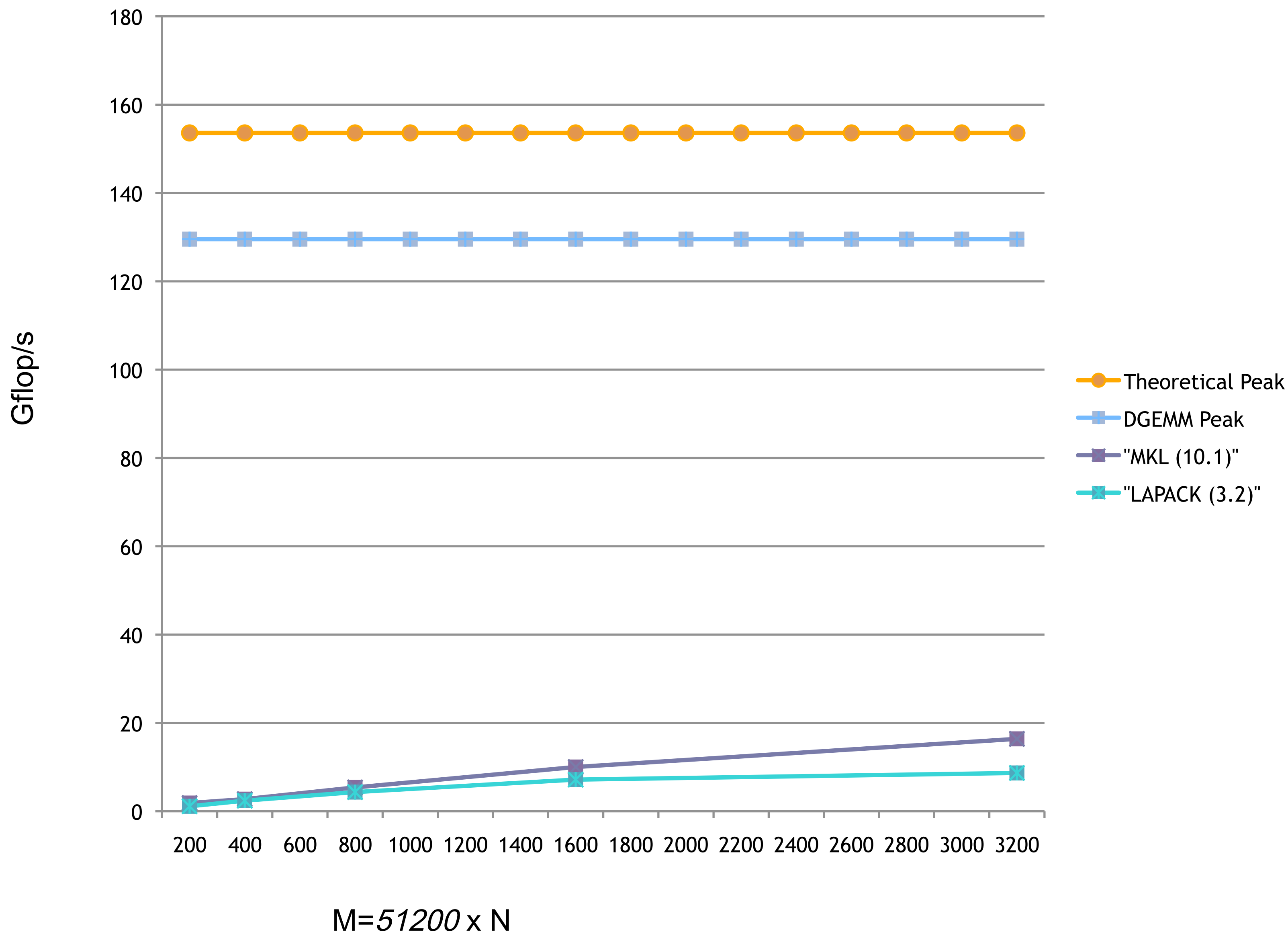
Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels

Those new algorithms
- have a very low granularity, they scale very well (multicore, petascale computing, …)
- removes a lots of dependencies among the tasks, (multicore, distributed computing)
- avoid latency (distributed computing, out-of-core)
- rely on fast kernels

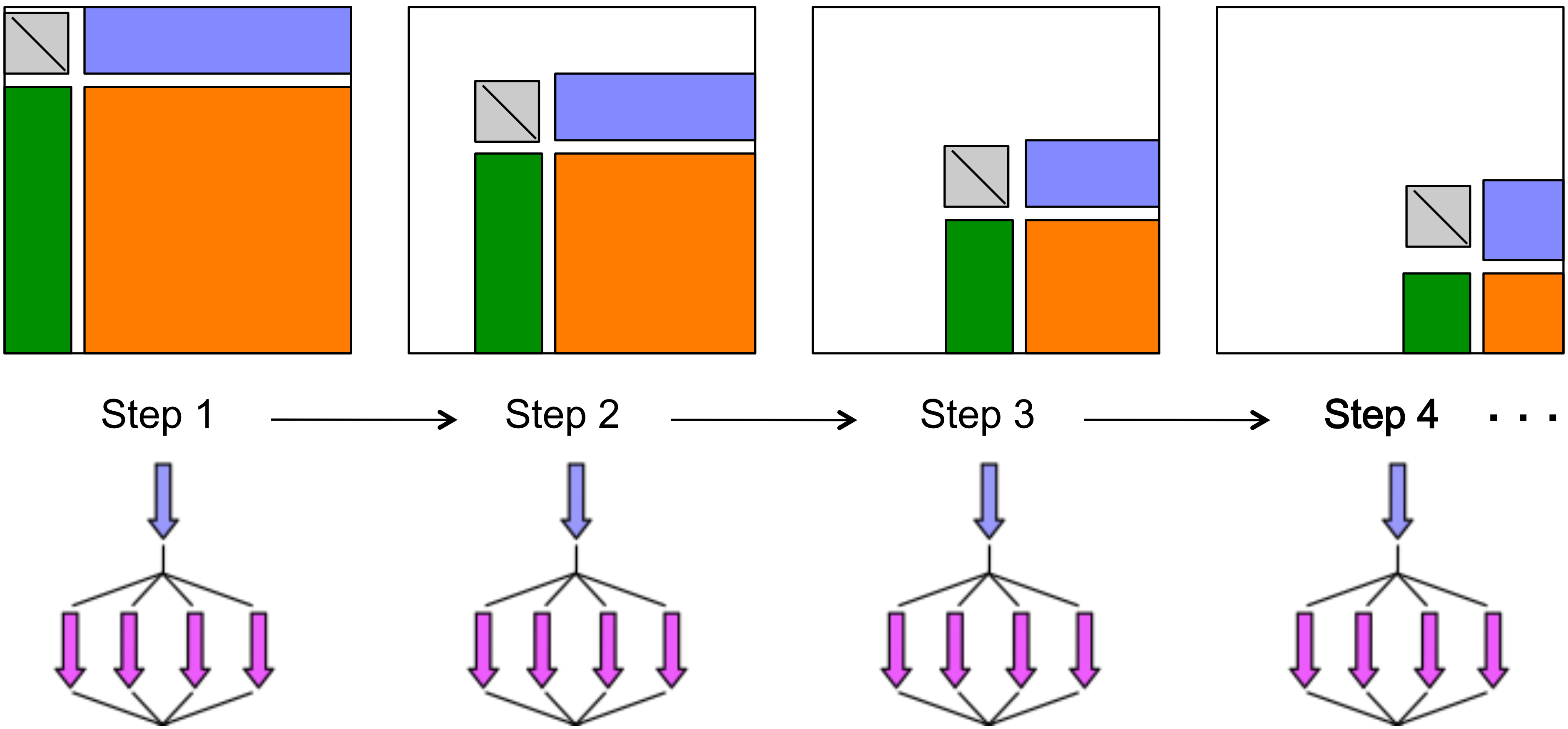
Those new algorithms need new kernels and rely on efficient scheduling algorithms.

QR Factorization Intel 16 cores

Tall Skinny Matrices

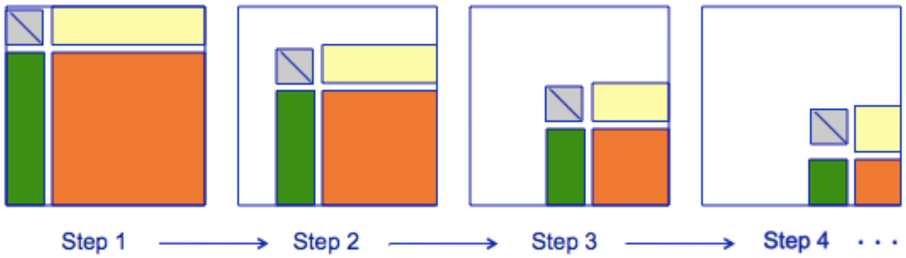


LAPACK QR

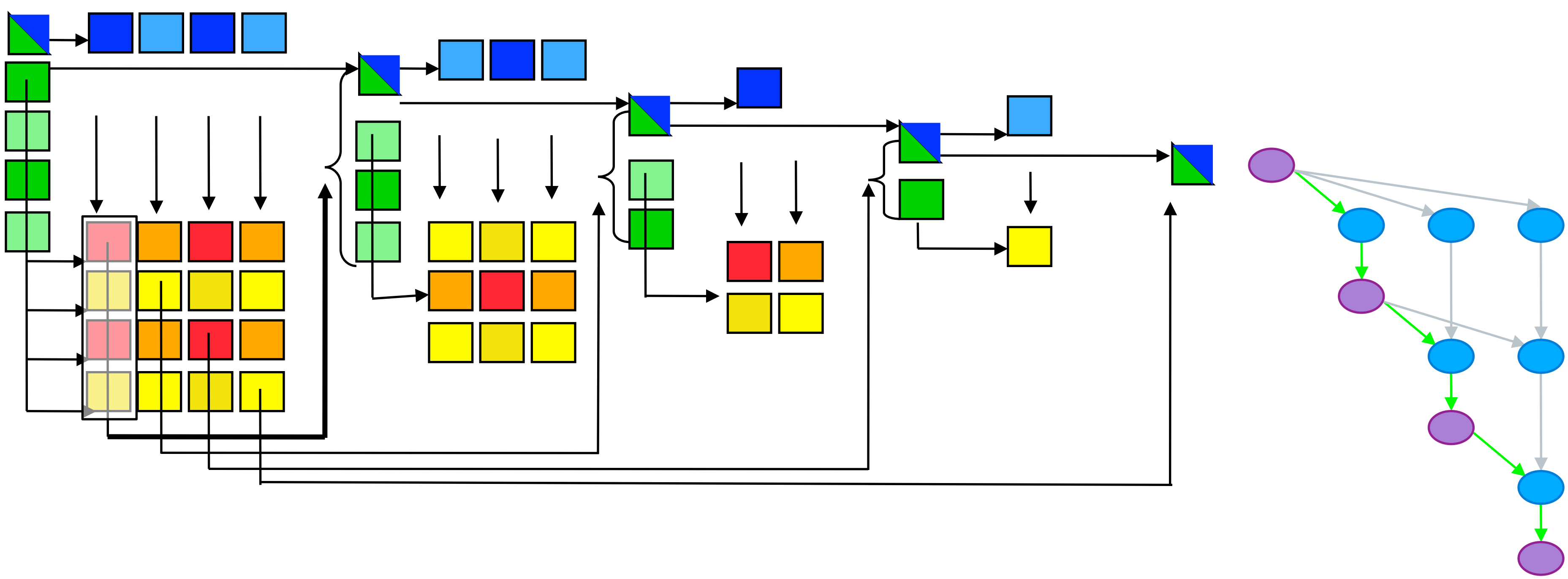


- **Fork-join, bulk synchronous processing**

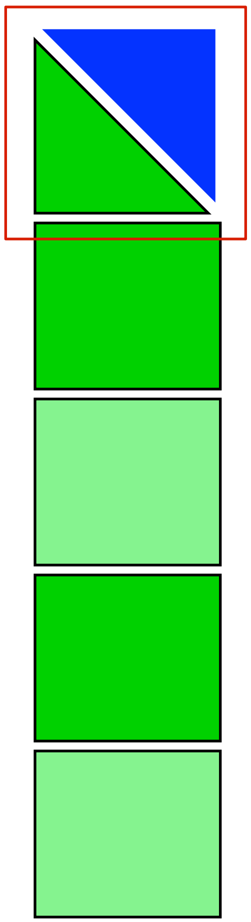
Parallel Tasks in QR



- Break into smaller tasks and remove dependencies

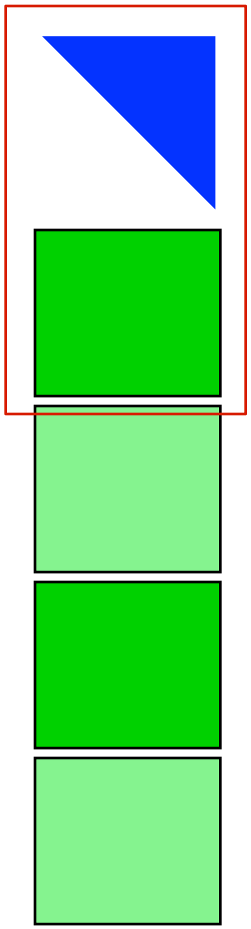


Parallel Tasks in QR



Step 1: QR of block 1,1

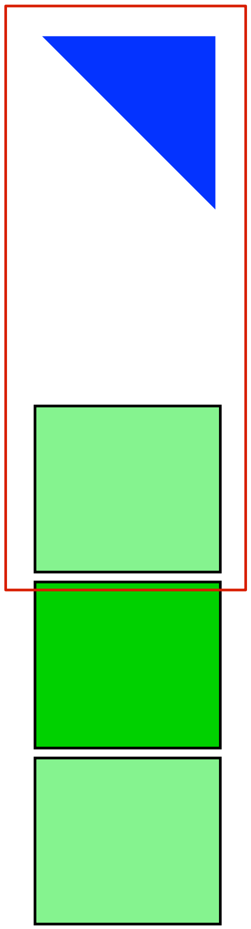
Parallel Tasks in QR



Step 1: QR of block 1,1

Step 2: Use R to zero $A_{1,2}$

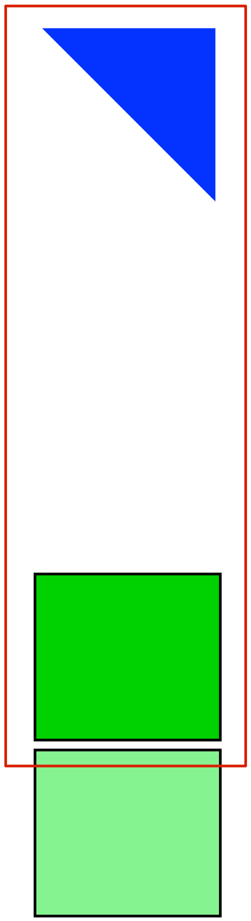
Parallel Tasks in QR



Step 1: QR of block 1,1

Step 2: Use R to zero $A_{1,2}$

Parallel Tasks in QR



Step 1: QR of block 1,1

Step 2: Use R to zero $A_{1,2}$

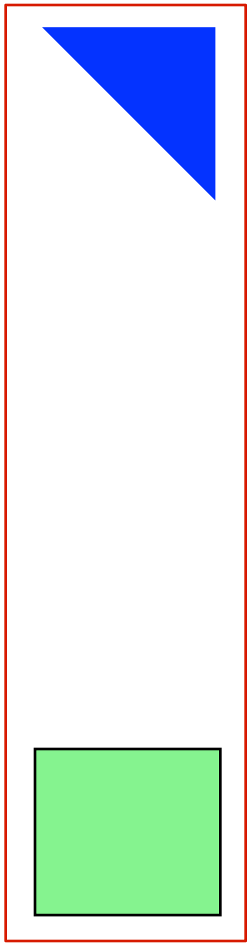
Step3: Use R to zero $A_{1,3}$

.

.

.

Parallel Tasks in QR



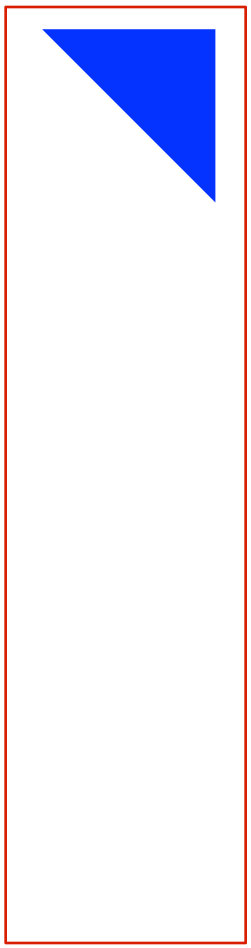
Step 1: QR of block 1,1

Step 2: Use R to zero $A_{1,2}$

Step3: Use R to zero $A_{1,3}$

.
.
.

Parallel Tasks in QR



Step 1: QR of block 1,1

Step 2: Use R to zero $A_{1,2}$

Step3: Use R to zero $A_{1,3}$

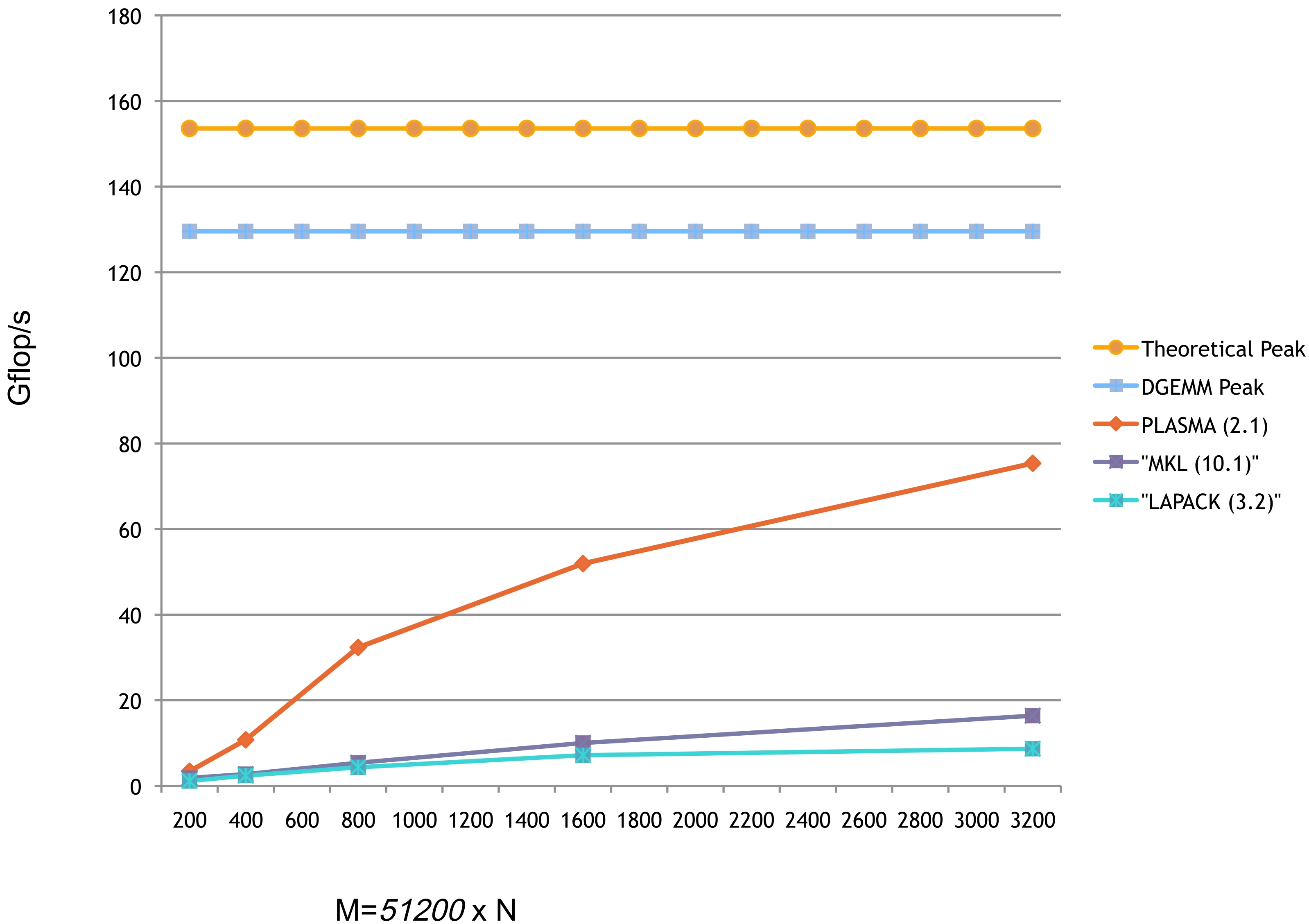
.

.

.

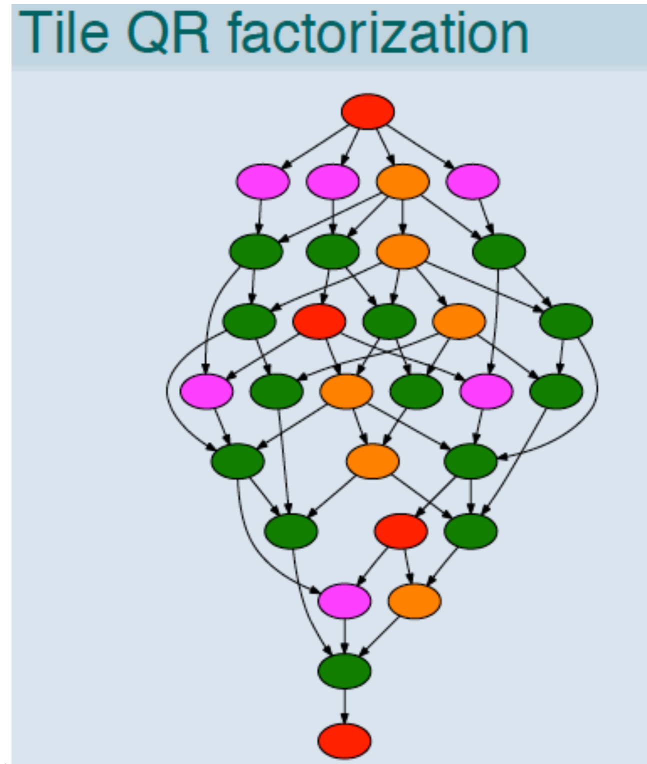
QR Factorization Intel 16 cores

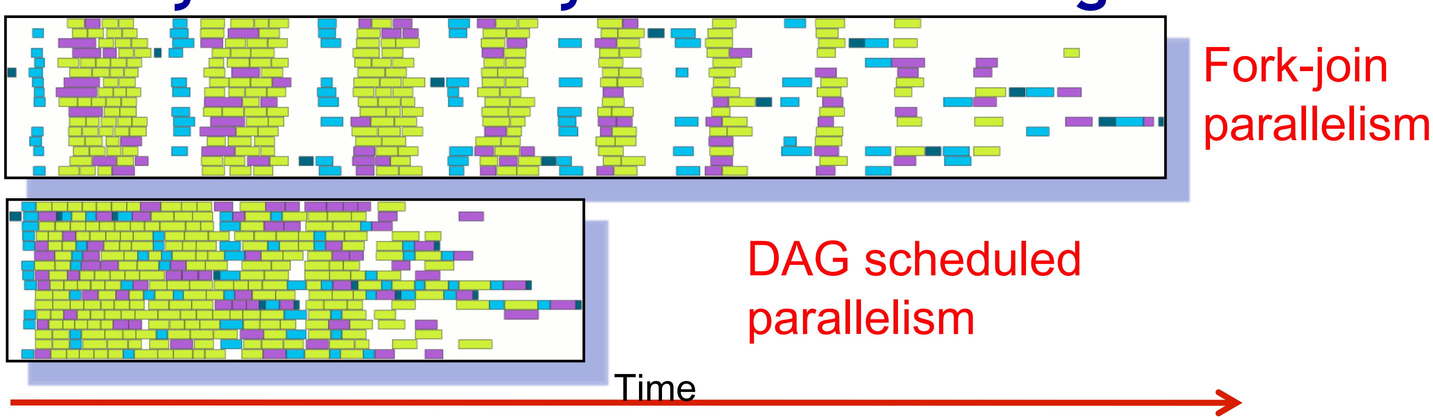
Tall Skinny Matrices



PLASMA: Parallel Linear Algebra s/w for Multicore Architectures

- **Objectives**
  - high utilization of each core
  - scaling to large number of cores
  - shared or distributed memory
- **Methodology**
  - DAG scheduling
  - explicit parallelism
  - implicit communication
  - Fine granularity / block data layout
- **Arbitrary DAG with dynamic scheduling**





Communication Avoiding Algorithms

- Goal: Algorithms that communicate as little as possible
- Jim Demmel and company have been working on algorithms that obtain a provable minimum communication.
- Direct methods (BLAS, LU, QR, SVD, other decompositions)
  - Communication lower bounds for *all* these problems
  - Algorithms that attain them (*all* dense linear algebra, some sparse)
    - Mostly not in LAPACK or ScaLAPACK (yet)
- Iterative methods – Krylov subspace methods for Ax=b, Ax=λx
  - Communication lower bounds, and algorithms that attain them (depending on sparsity structure)
    - Not in any libraries (yet)
- For QR Factorization they can show:

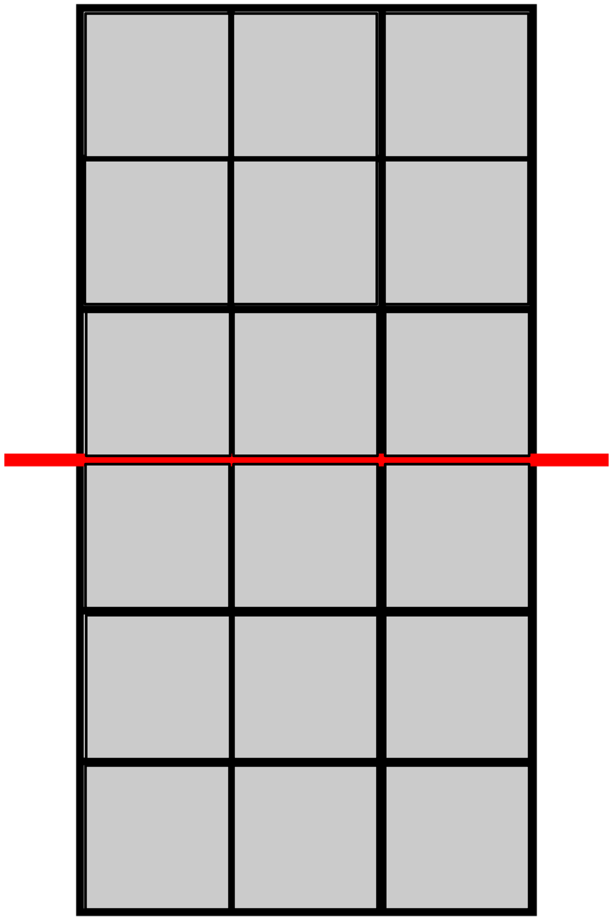
	Lower bound
# flops	$\Theta(mn^2)$
# words	$\Theta(\frac{mn^2}{\sqrt{W}})$
# messages	$\Theta(\frac{mn^2}{W^{3/2}})$

Communication Reducing QR Factorization

TS matrix
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- merge the domains

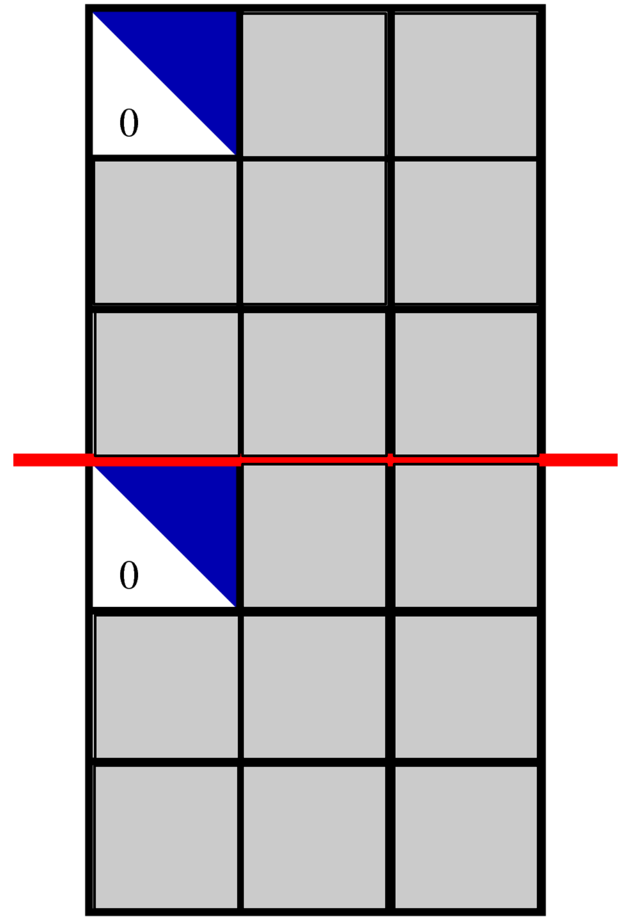


Communication Reducing QR Factorization

TS matrix
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- **panel factorization**
- updating the trailing submatrix
- merge the domains



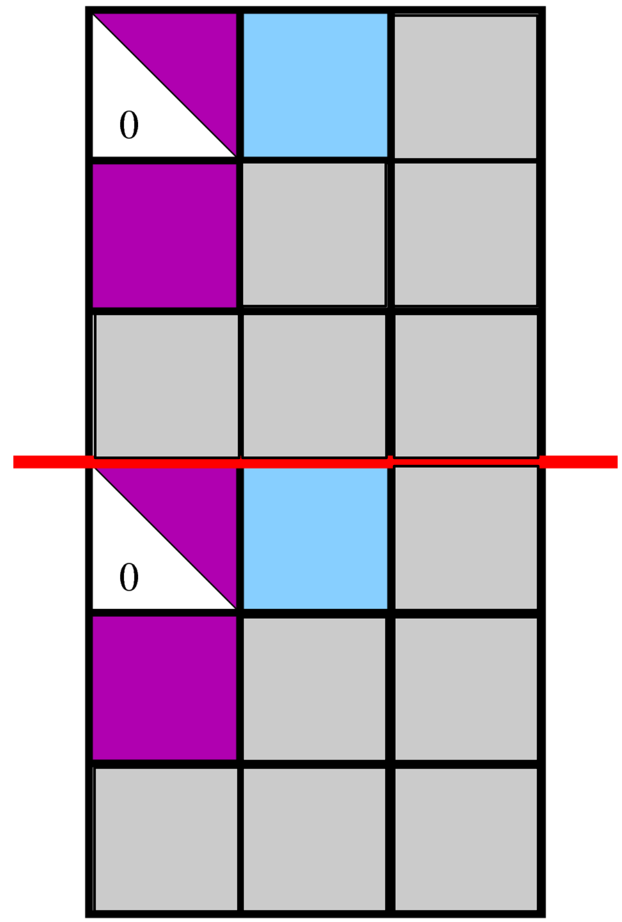
Communication Reducing QR Factorization

TS matrix

- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- **panel factorization**
- **updating the trailing submatrix**
- merge the domains

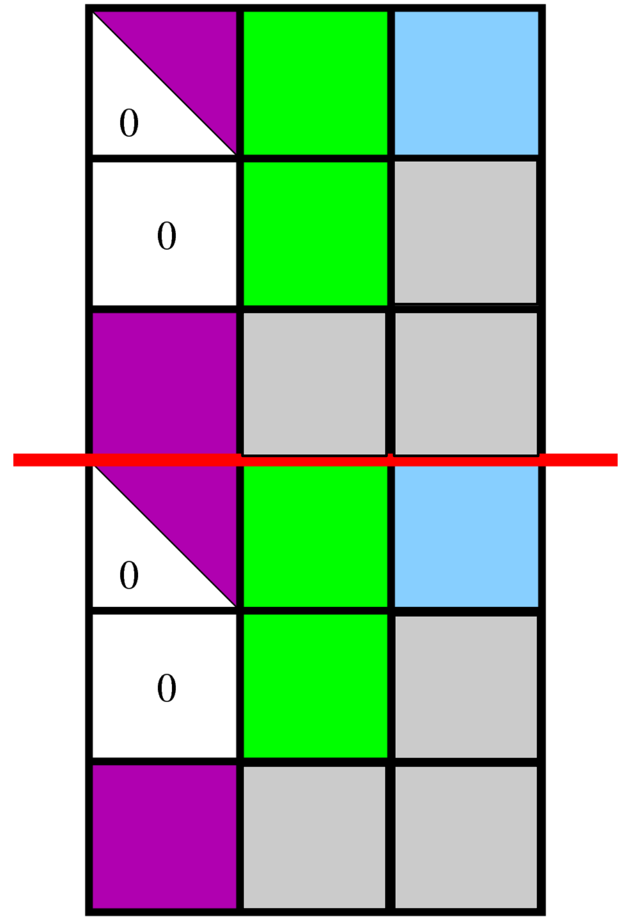


Communication Reducing QR Factorization

TS matrix
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- **panel factorization**
- **updating the trailing submatrix**
- merge the domains



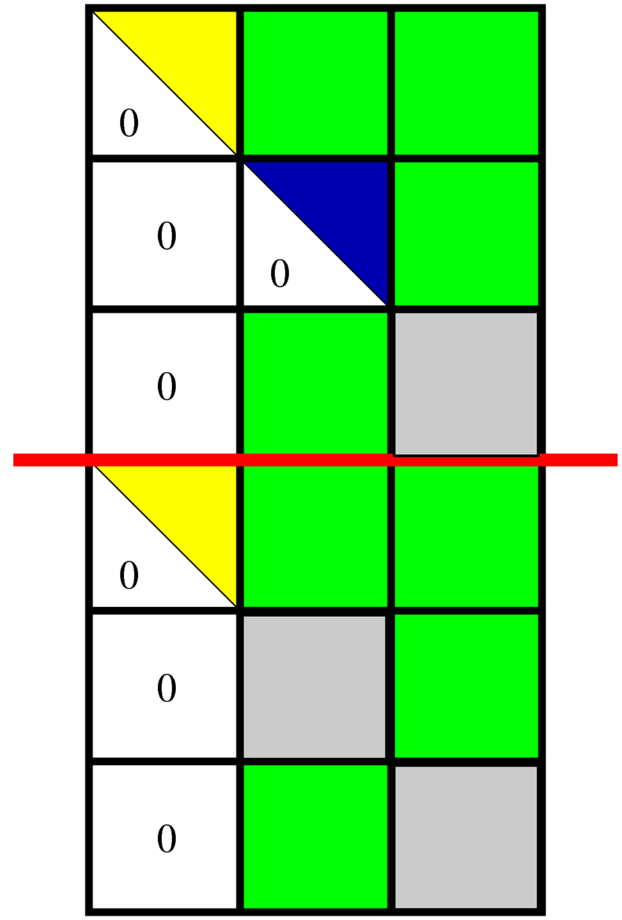
Communication Reducing QR Factorization

TS matrix

- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- merge the domains

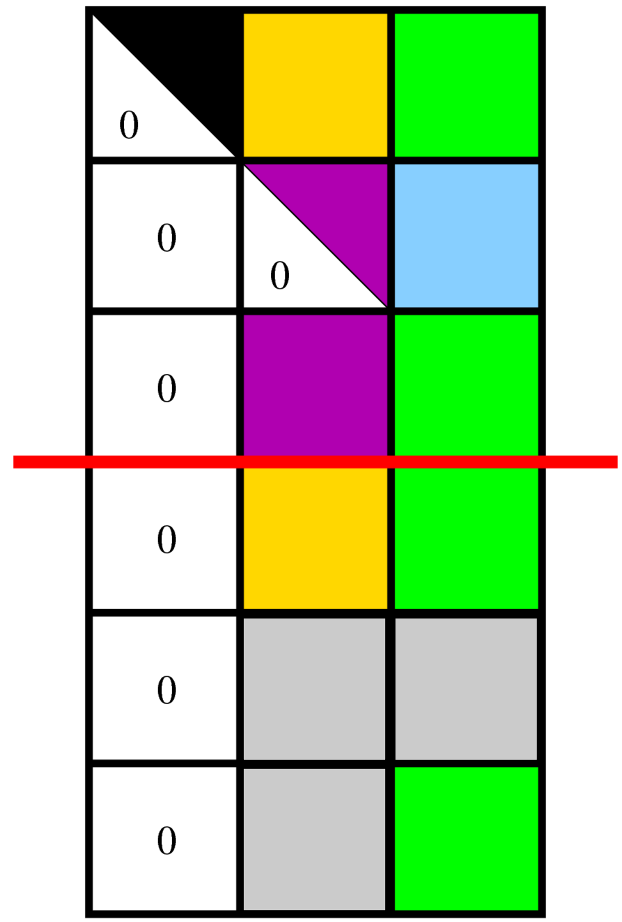


Communication Reducing QR Factorization

TS matrix
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- merge the domains



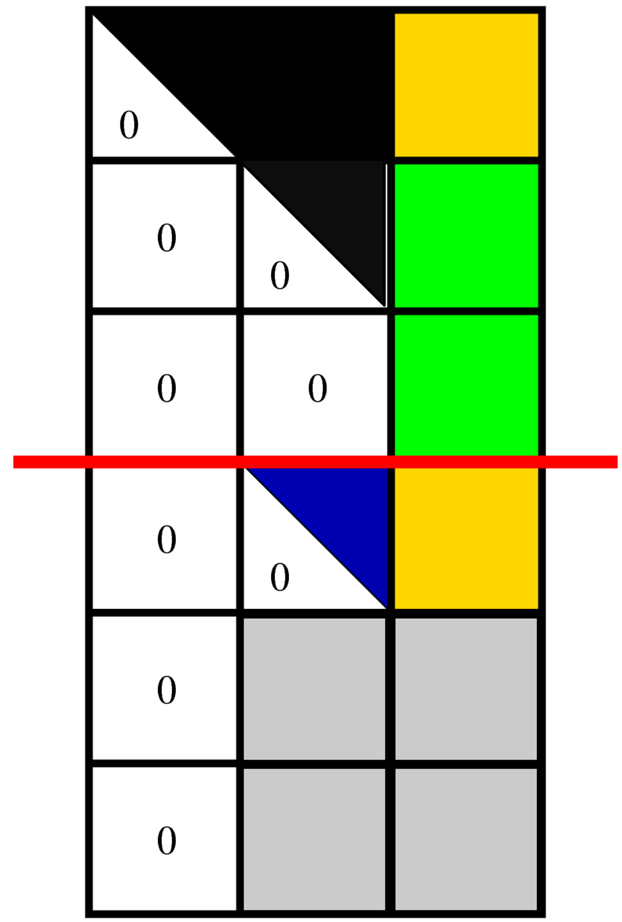
Communication Reducing QR Factorization

TS matrix

- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- merge the domains

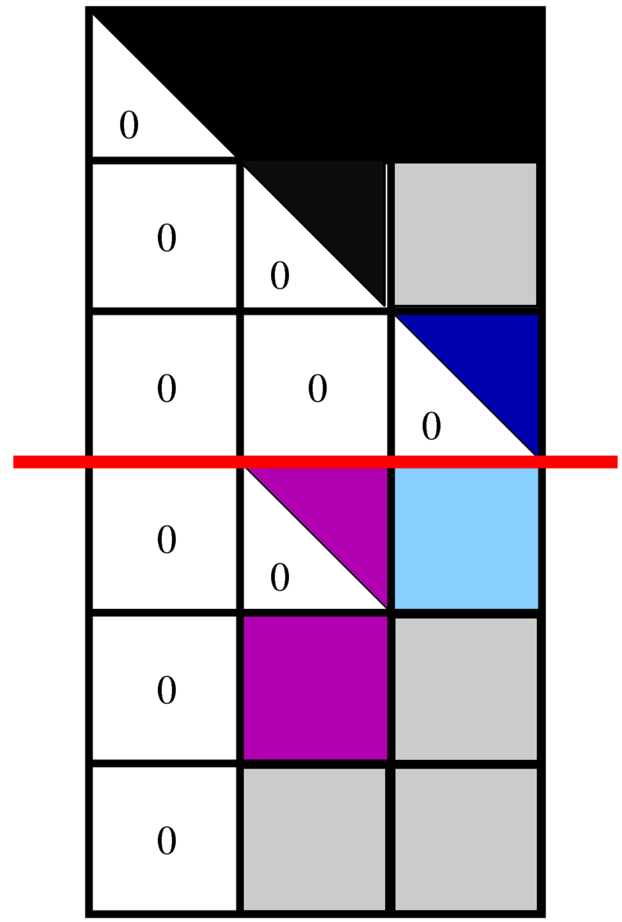


Communication Reducing QR Factorization

TS matrix
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- merge the domains



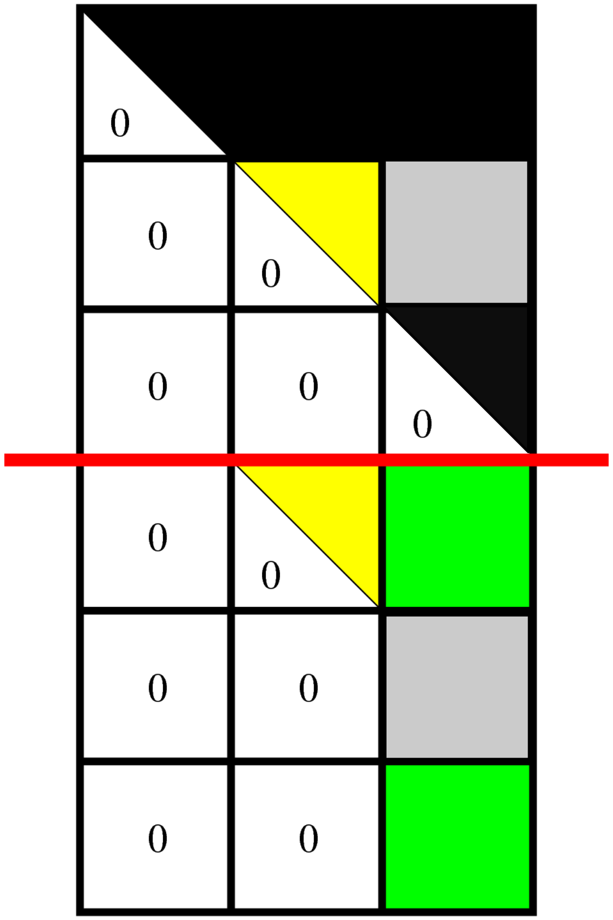
Communication Reducing QR Factorization

TS matrix

- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- **updating the trailing submatrix**
- **merge the domains**

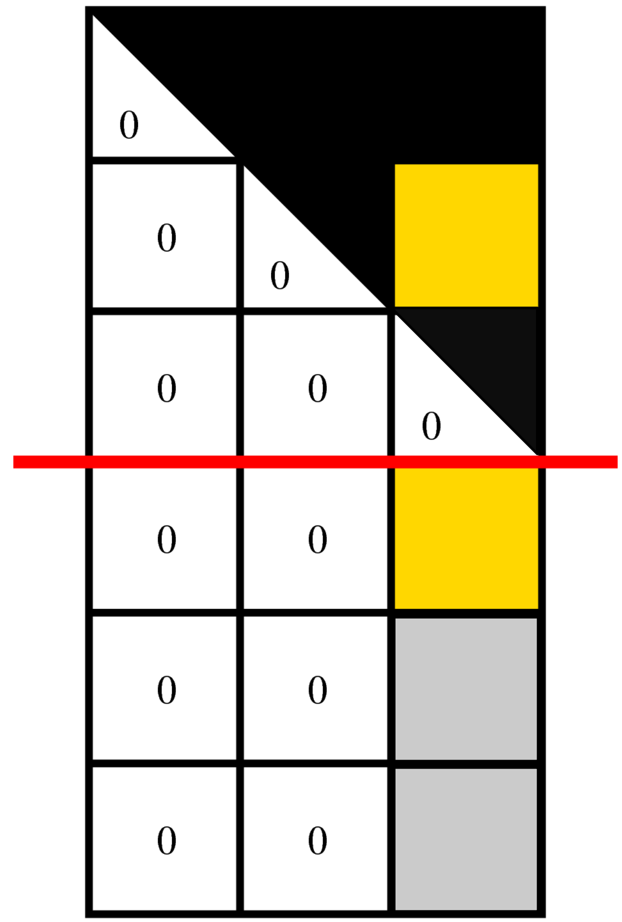


Communication Reducing QR Factorization

TS matrix
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- **merge the domains**



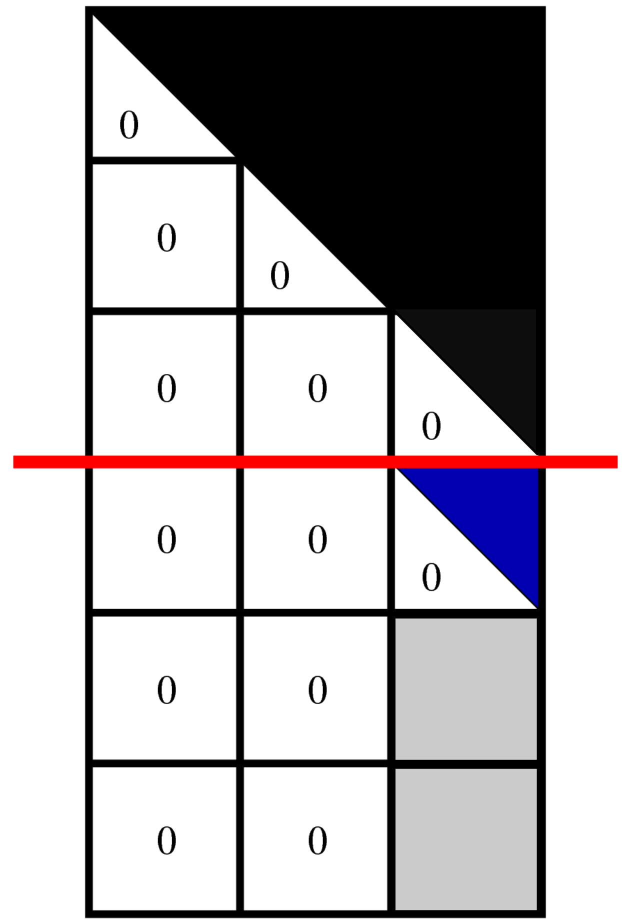
Communication Reducing QR Factorization

TS matrix

- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- merge the domains

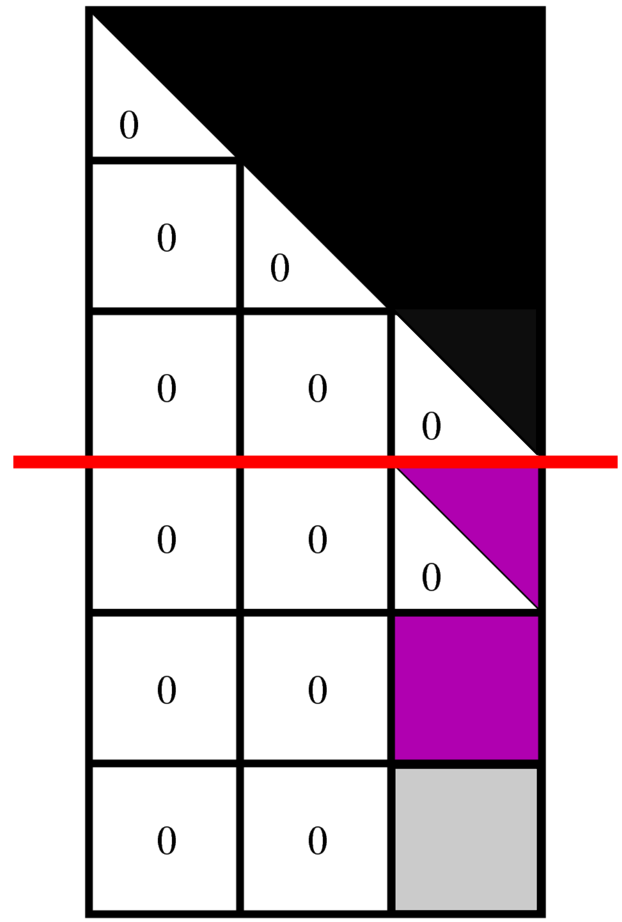


Communication Reducing QR Factorization

TS matrix
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- **panel factorization**
- updating the trailing submatrix
- merge the domains



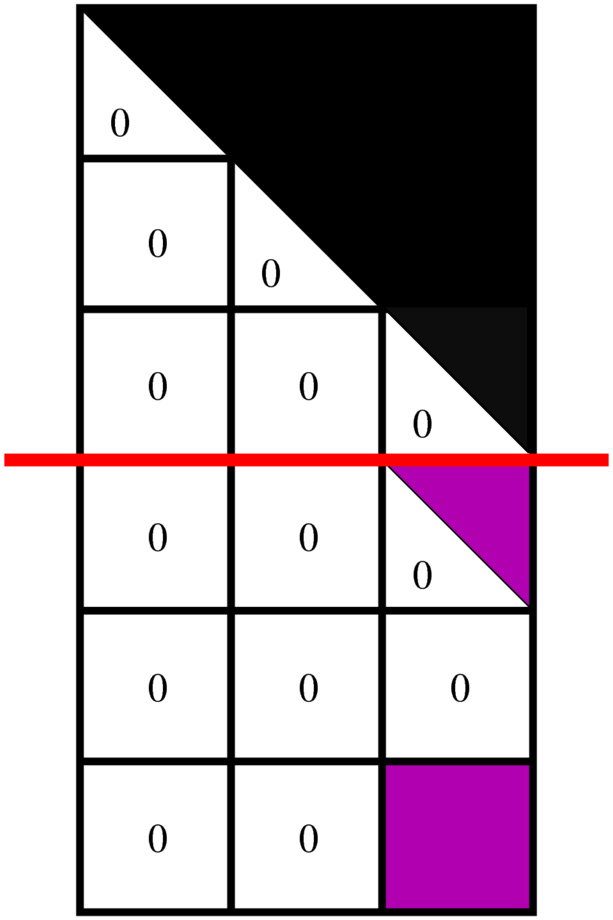
Communication Reducing QR Factorization

TS matrix

- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- **panel factorization**
- updating the trailing submatrix
- merge the domains

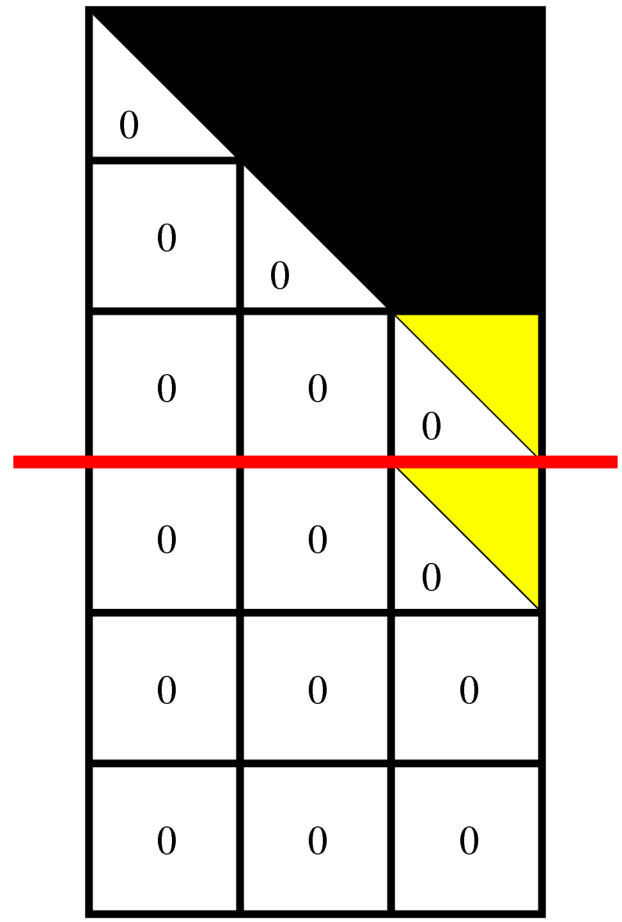


Communication Reducing QR Factorization

TS matrix
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- **merge the domains**



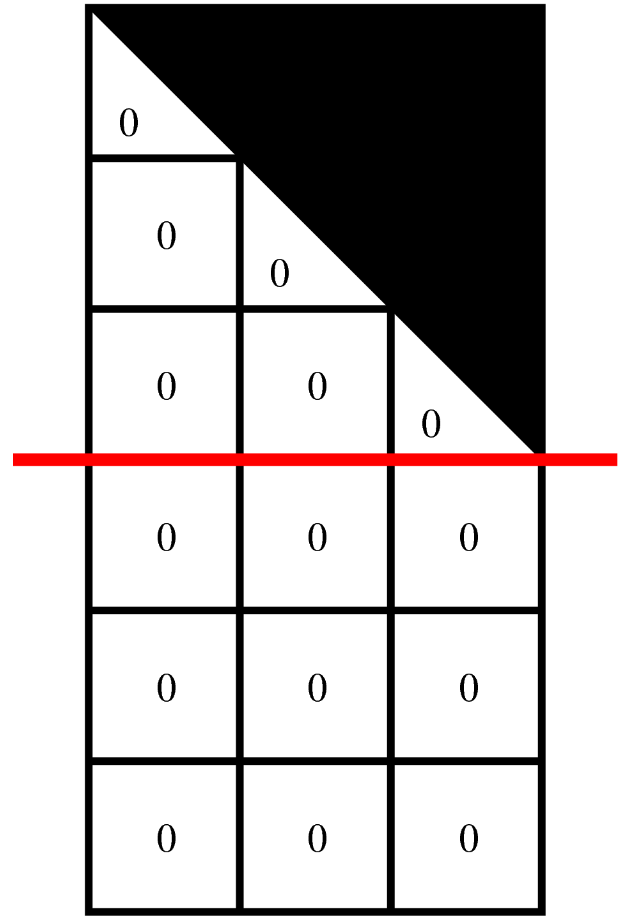
Communication Reducing QR Factorization

TS matrix

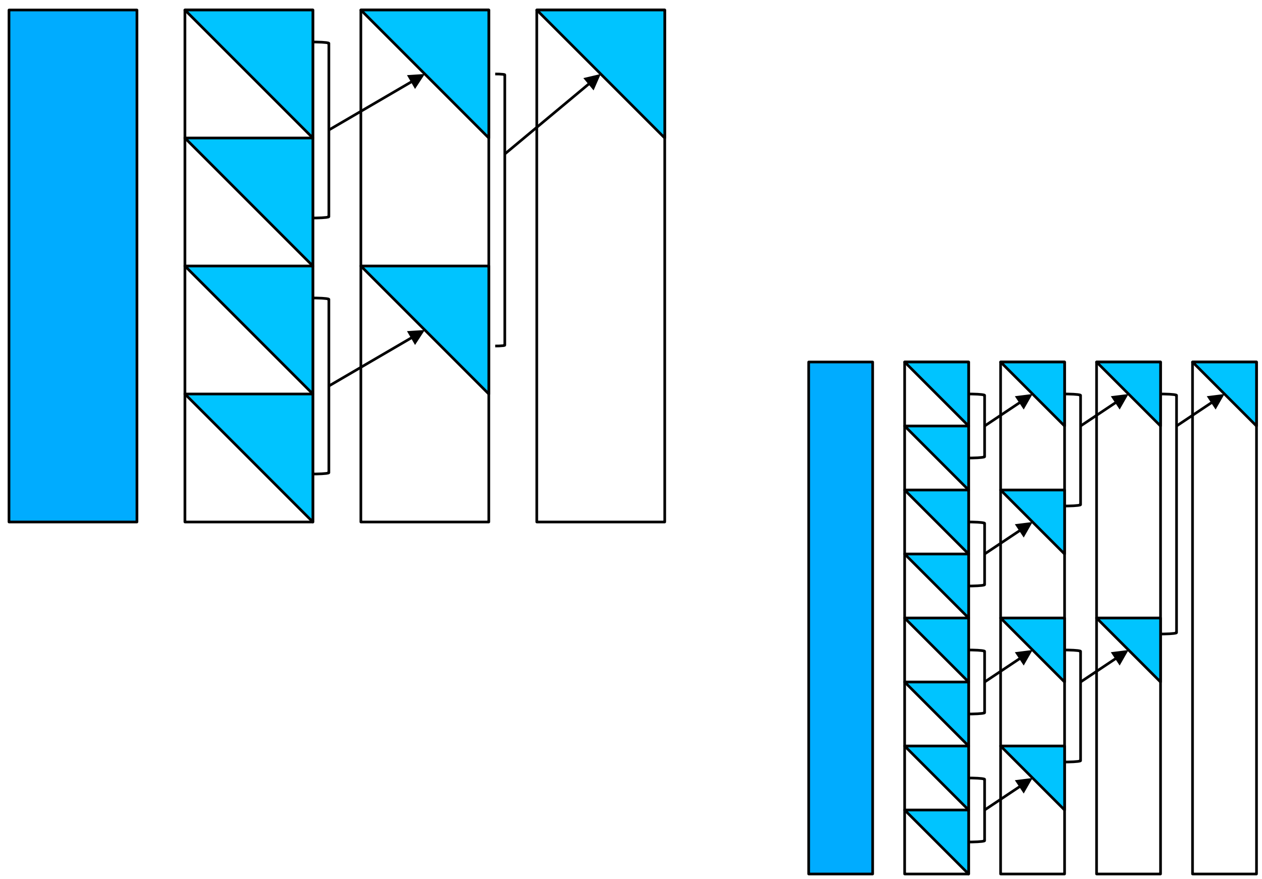
- MT=6 and NT=3
- split into 2 domains

3 overlapped steps

- panel factorization
- updating the trailing submatrix
- merge the domains
- Final R computed



Example with 4 and 8 Domains



A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Execution Trace

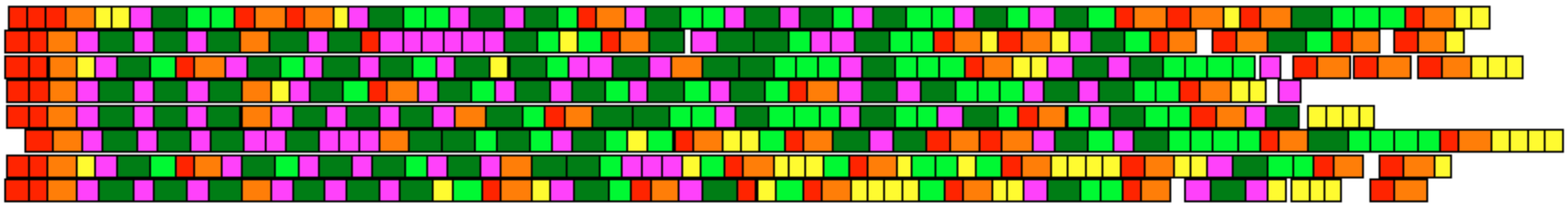


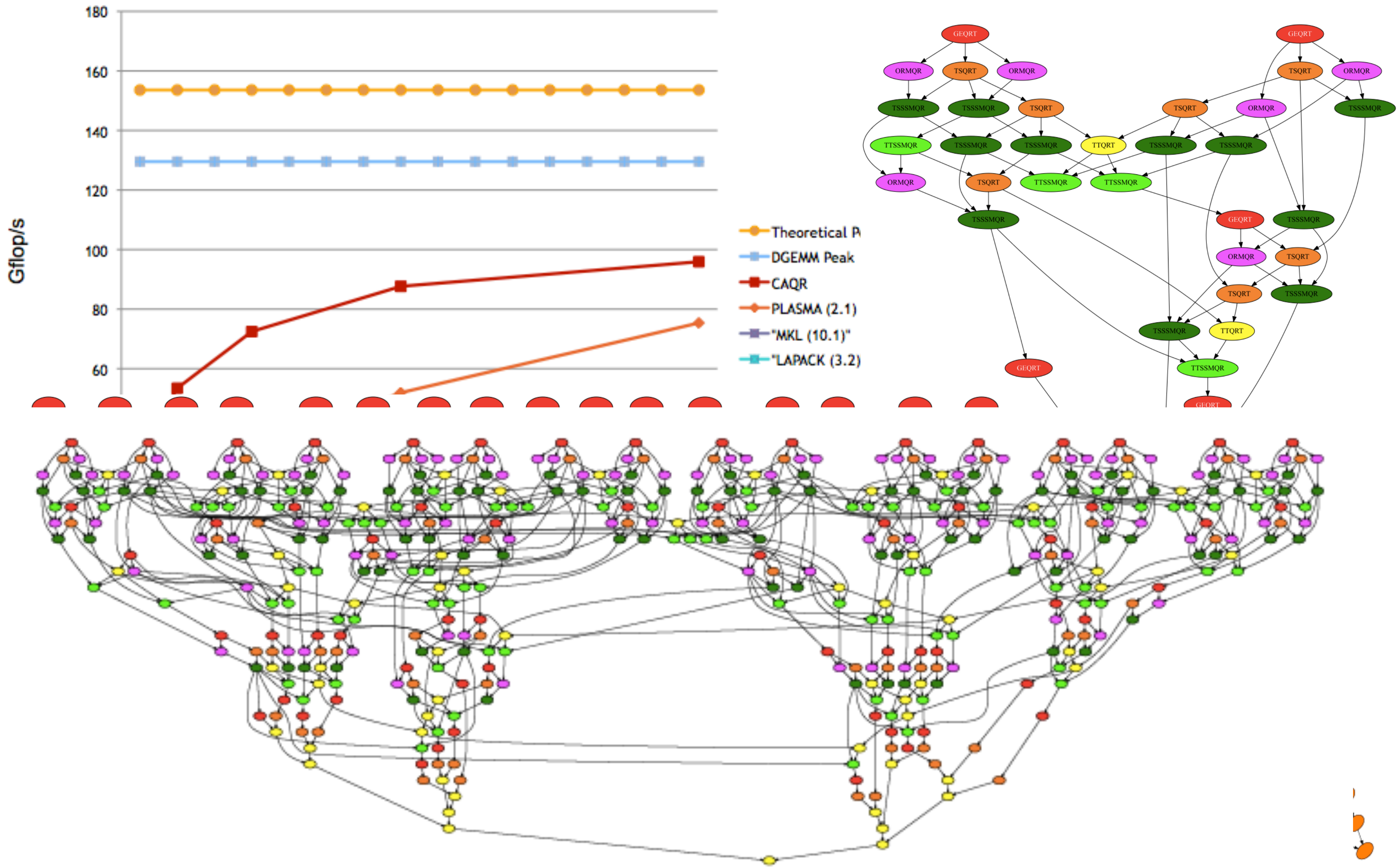
Fig. 11. Parallel execution traces of SP-16 with MT=32 and NT=4 on 8 cores.

TABLE III
IMPROVEMENT OF SP-CAQR AGAINST OTHER LIBRARIES (PERFORMANCE RATIO).

Matrix sizes	PLASMA	MKL	ScaLAPACK	LAPACK
51200 – 200	9.54	8.77	3.38	28.63
51200 – 3200	1.27	4.10	2.88	11.05

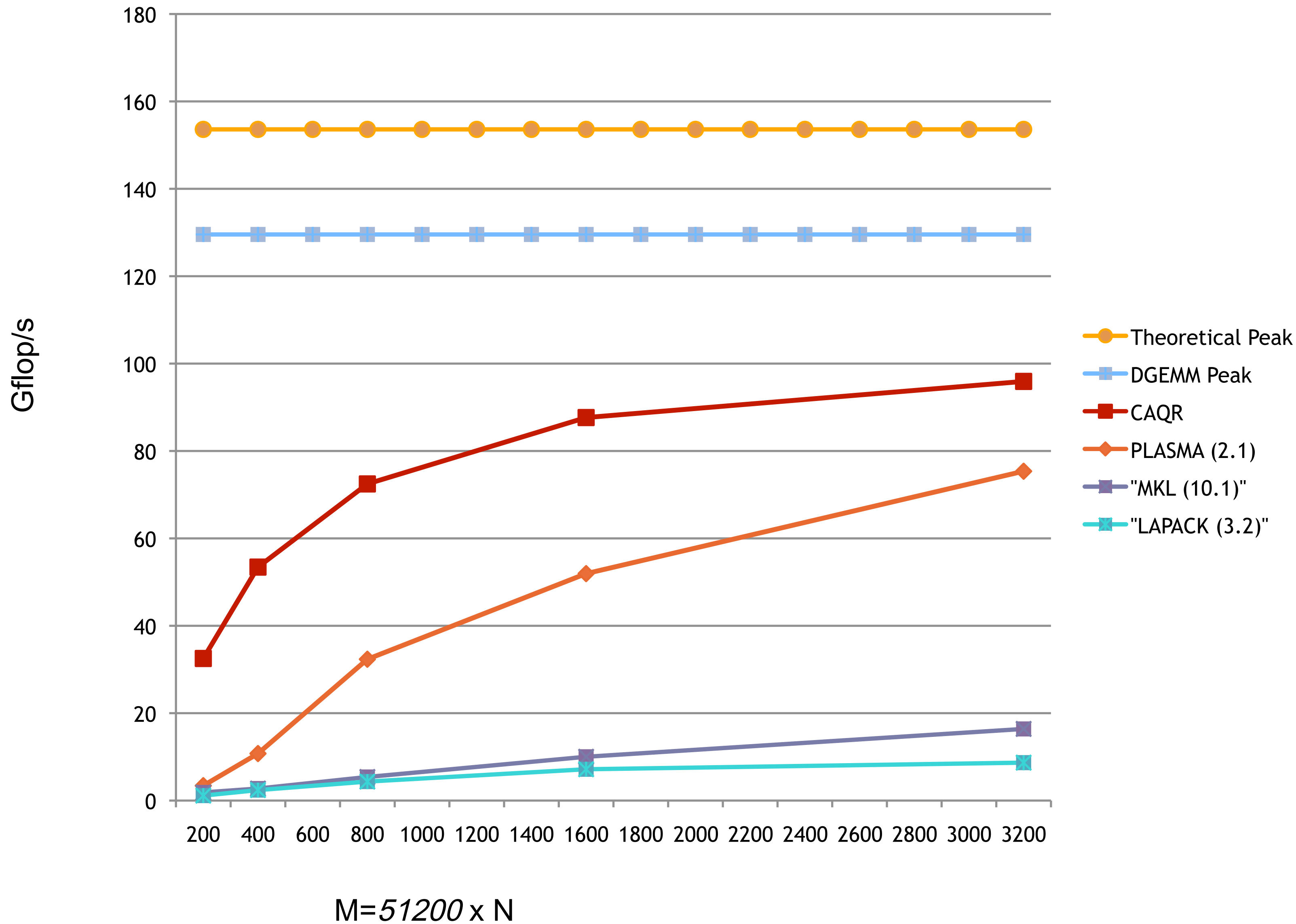
16 core run

Communication Reducing QR Factorization



(a) One domain: SP-1 (or PLASMA-like Tile QR factorization).

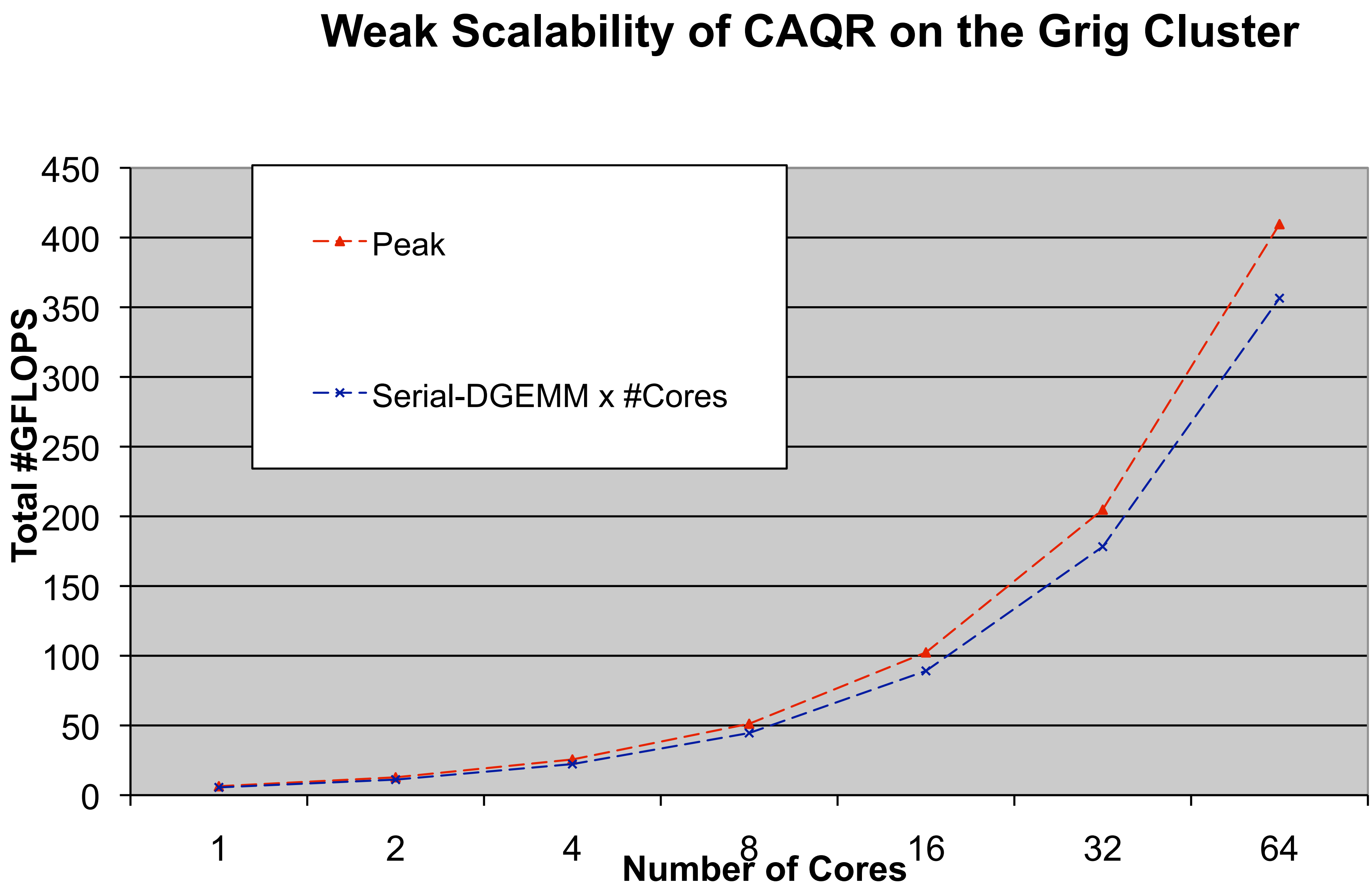
QR Factorization Intel 16 cores
Tall Skinny Matrices



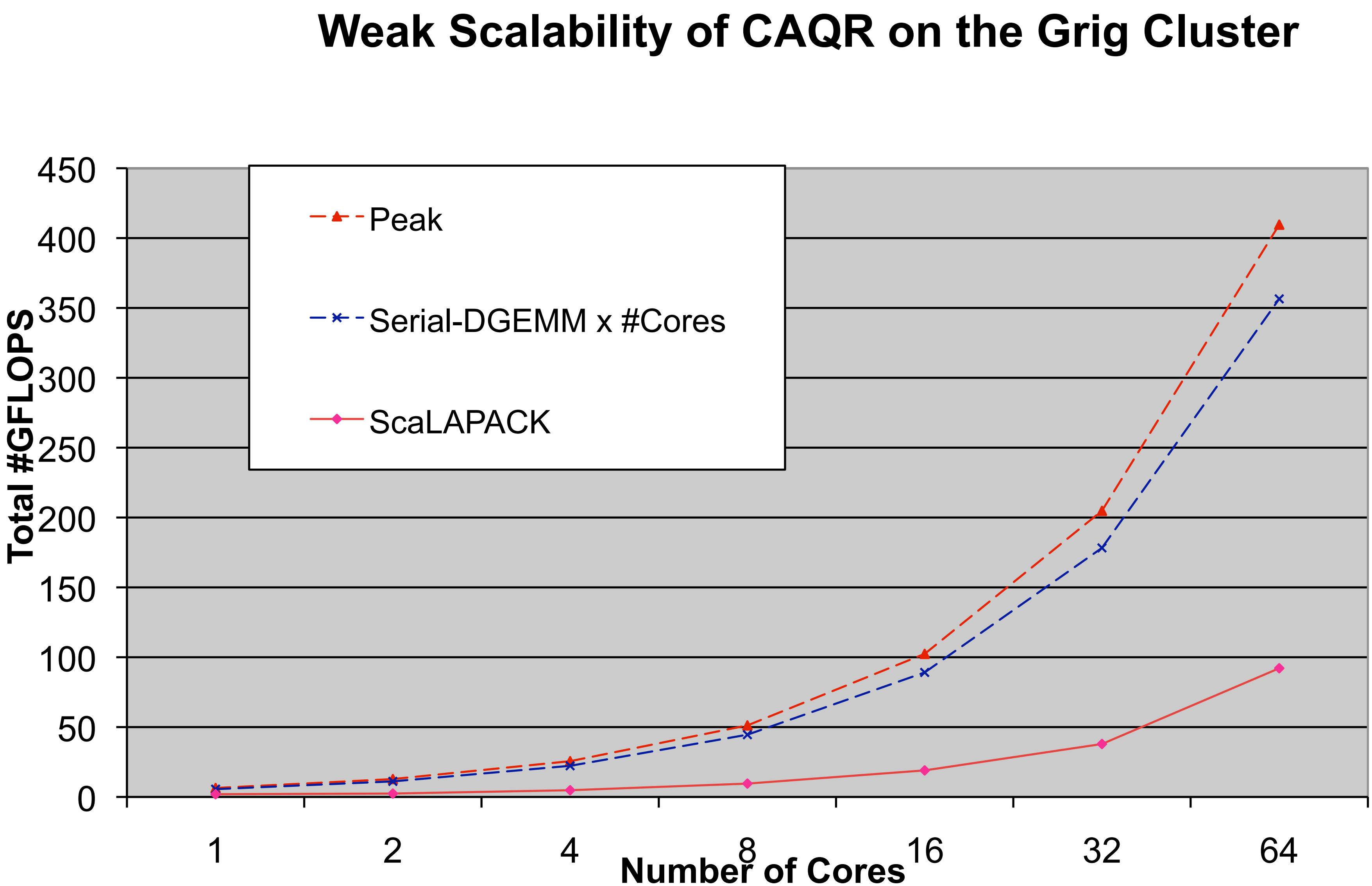
Cluster Experiment

- grig.sinrg.cs.utk.edu
- 61 nodes
  - Two CPUs per node
  - Intel Xeon 3.20GHz
  - Peak performance 6.4 GFLOPS
  - Myrinet interconnection (MX 1.0.0)
- Goto BLAS 1.26
  - DGEMM performance 5.57 GFLOPS (87%)
- MPICH-MX
- gcc 64 bits

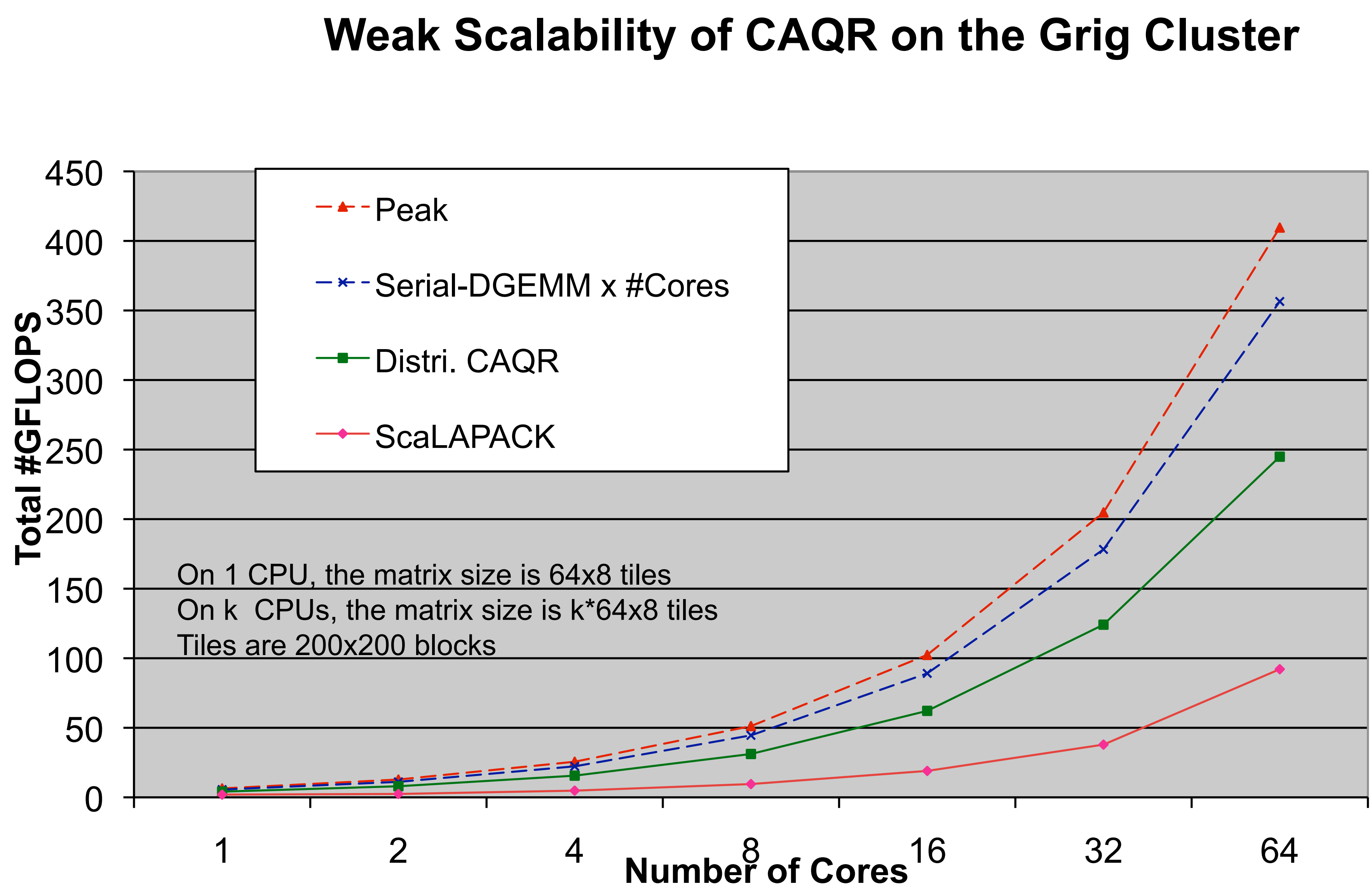
Weak Scalability (8 columns of tiles)



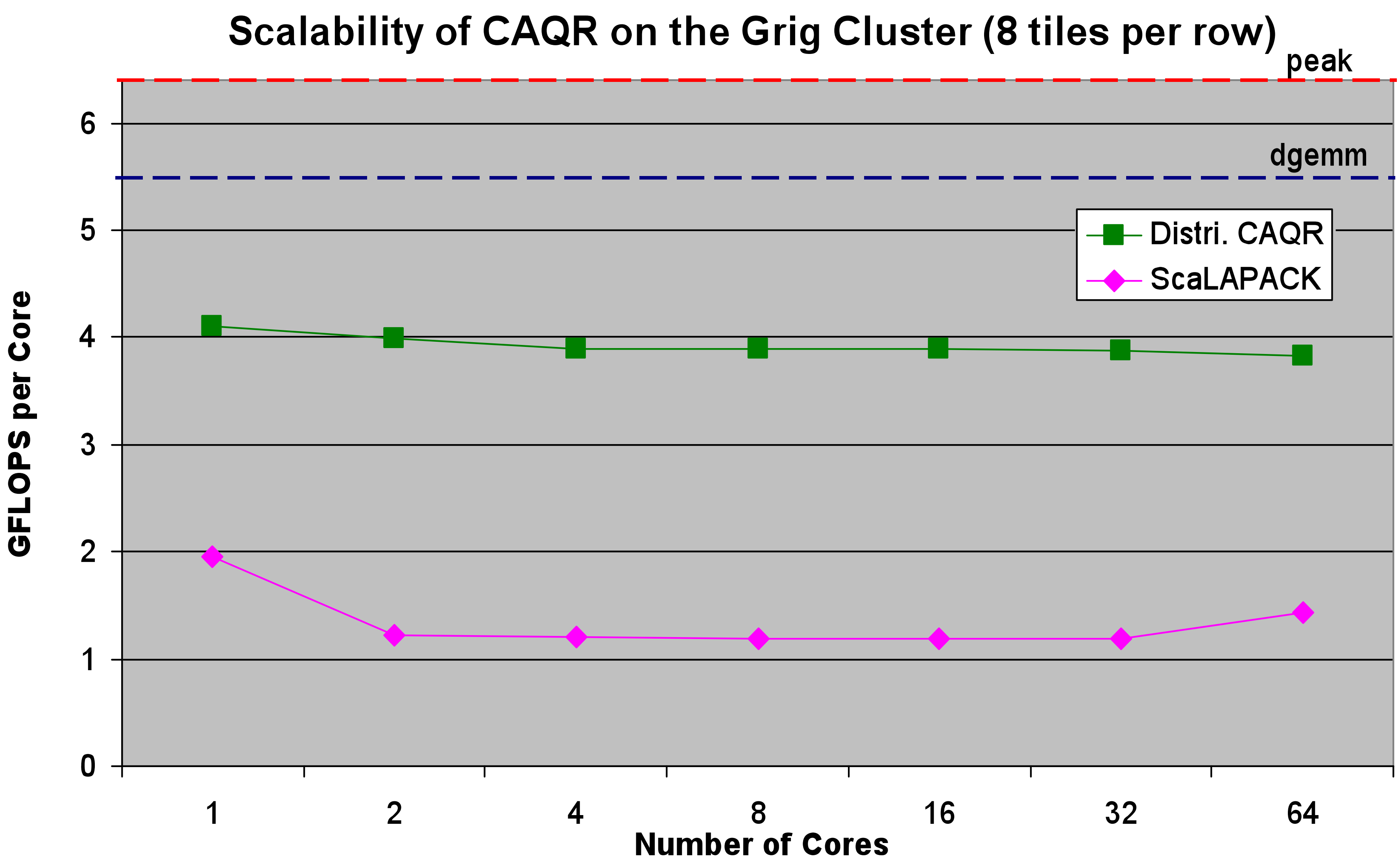
Weak Scalability (8 columns of tiles)



Weak Scalability (8 columns of tiles)



Weak Scalability (8 columns of tiles)



Futures

- Architectural trends are forcing major changes to our algorithms
- Communication avoiding algorithms will be critical for performance.
- PLASMA and MAGMA will make use of CA algorithms.