

# Overview of communication avoiding algorithms

Laura Grigori

*Alpines*

Inria Paris - LJLL, UPMC



# Plan

- Motivation
- Selected past work on reducing communication
- Brief overview of communication avoiding for dense linear algebra
  - LU, QR, Rank Revealing QR factorizations
  - Progressively implemented in ScaLAPACK, LAPACK
- Communication avoiding for sparse linear algebra
  - Krylov subspace methods
- Conclusions

# Motivation - the communication wall

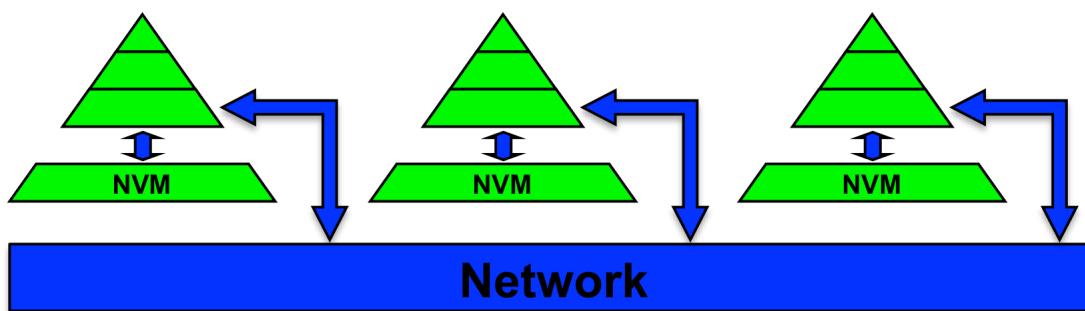
- Time to move data  $\gg$  time per flop
  - Gap steadily and exponentially growing over time

Annual improvements			
Time/flop		Bandwidth	Latency
59%	Network	26%	15%
	DRAM	23%	5.5%

“Getting up to speed, *The future of supercomputing*” 2004, data from 1995-2004

“We are going to hit the **memory wall**, unless something basic changes”

[W. Wulf, S. McKee, 95]



# Compelling numbers

## DRAM bandwidth:

- Mid 90's ~ 0.2 bytes/flop – 1 byte/flop
- Past few years ~ 0.02 to 0.05 bytes/flop

## DRAM latency:

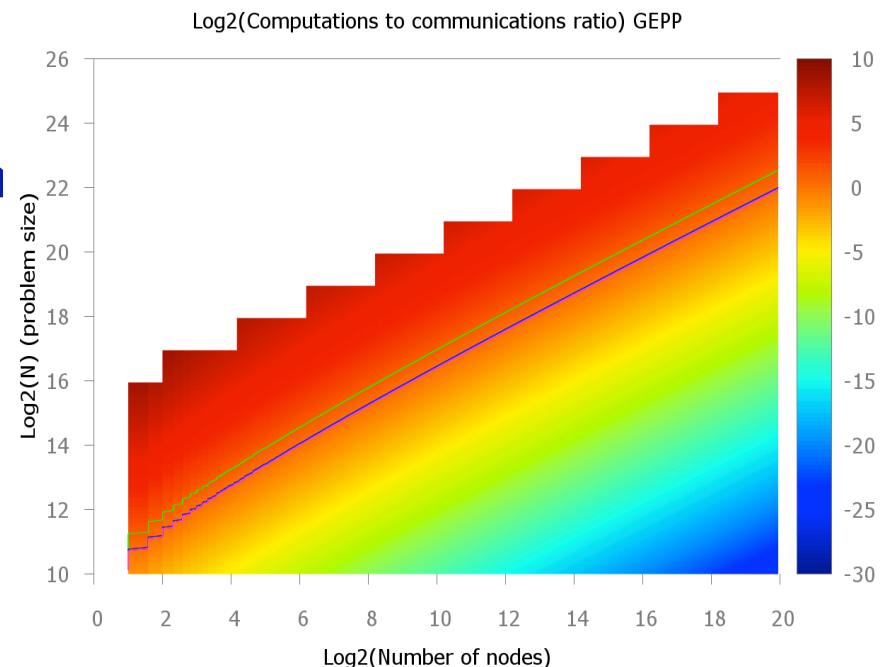
- DDR2 (2007) ~ 120 ns 1x
- DDR4 (2014) ~ 45 ns 2.6x in 7 years
- Stacked memory ~ similar to DDR4

## Time/flop

- 2006 Intel Yonah ~ 2GHz x 2 cores (32 GFlops/chip) 1x
- 2015 Intel Haswell ~2.3GHz x 16 cores (588 GFlops/chip) 18x in 9 years

# Approaches for reducing communication

- Tuning
  - Overlap communication and computation, at most a factor of 2 speedup
- Same numerical algorithm, different schedule of the computation
  - Block algorithms for NLA
    - Barron and Swinnerton-Dyer, 1960
    - ScaLAPACK, Blackford et al 97
  - Cache oblivious algorithms for NLA
    - Gustavson 97, Toledo 97, Frens and Wise 03, Ahmed and Pingali 00
- Same algebraic framework, different numerical algorithm
  - The approach used in CA algorithms
  - More opportunities for reducing communication, may affect stability



# Communication Complexity of Dense Linear Algebra

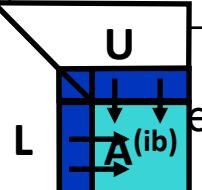
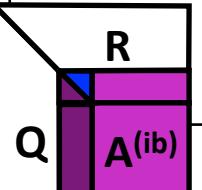
- Matrix multiply, using  $2n^3$  flops (sequential or parallel)
  - Hong-Kung (1981), Irony/Tishkin/Toledo (2004)
  - Lower bound on Bandwidth =  $\Omega (\text{#flops} / M^{1/2})$
  - Lower bound on Latency =  $\Omega (\text{#flops} / M^{3/2})$
- Same lower bounds apply to LU using reduction
  - Demmel, LG, Hoemmen, Langou 2008

$$\begin{pmatrix} I & -B \\ A & I \\ & I \end{pmatrix} = \begin{pmatrix} I & & \\ A & I & \\ & & I \end{pmatrix} \begin{pmatrix} I & -B \\ & I & AB \\ & & I \end{pmatrix}$$

- And to almost all direct linear algebra [Ballard, Demmel, Holtz, Schwartz, 09]

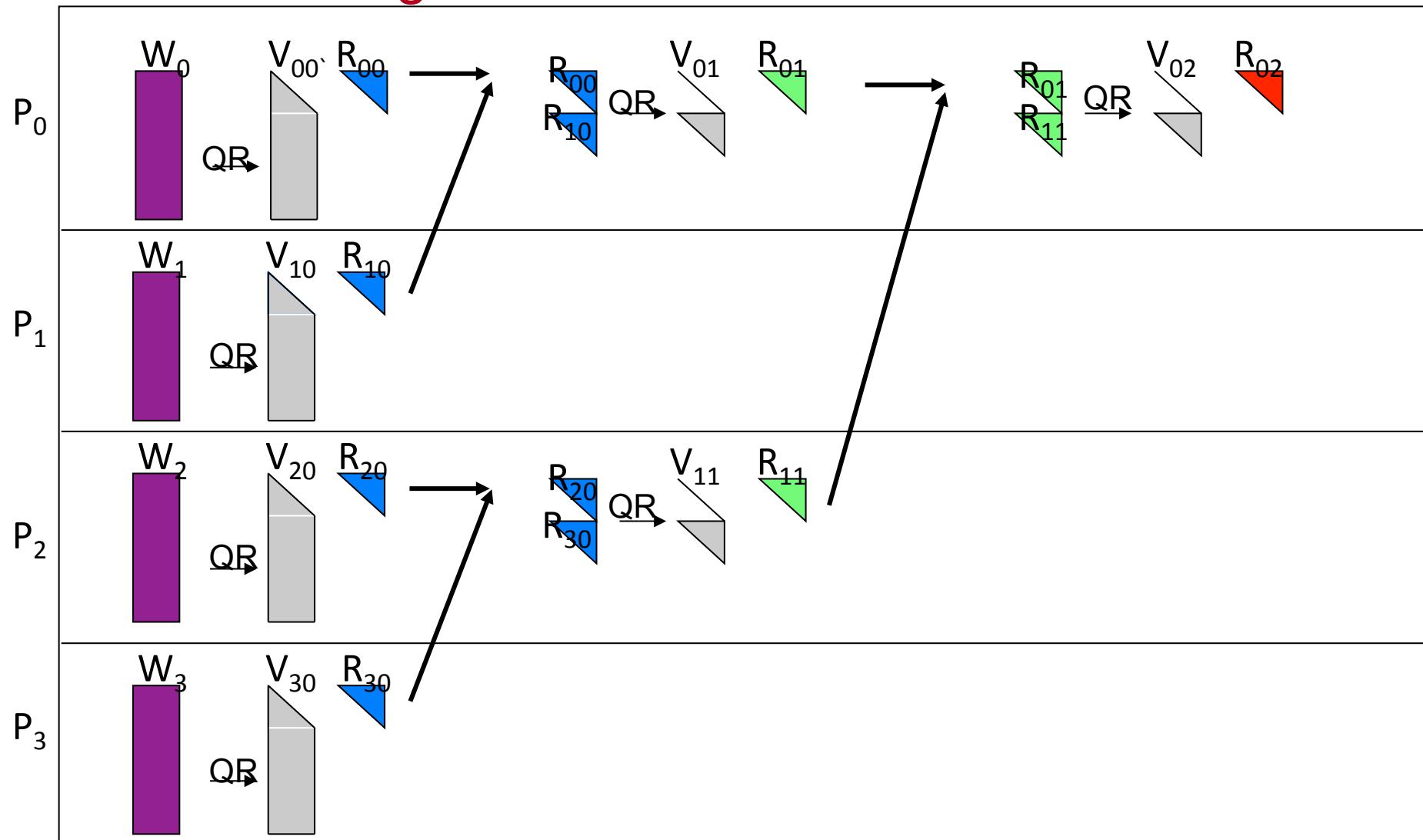
## 2D Parallel algorithms and communication bounds

- If memory per processor =  $n^2 / P$ , the lower bounds on communication are  
 $\#words\_moved \geq \Omega ( n^2 / P^{1/2} ), \quad \#messages \geq \Omega ( P^{1/2} )$

Algorithm	Minimizing $\#words$ (not $\#messages$ )	Minimizing $\#words$ and $\#messages$
Cholesky	ScalAPACK	ScalAPACK
LU	 L (blue) and U (white) blocks. A (ib) block is highlighted in cyan. Arrows indicate partial pivoting. ScalAPACK uses partial pivoting	[LG, Demmel, Xiang, 08] [Khabou, Demmel, LG, Gu, 12] uses tournament pivoting
QR	 Q (purple) and R (white) blocks. A (ib) block is highlighted in magenta. ScalAPACK	[Demmel, LG, Hoemmen, Langou, 08] uses different representation of Q
RRQR	 Q (purple) and A (ib) blocks. ScalAPACK	[Demmel, LG, Gu, Xiang 13] uses tournament pivoting, 3x flops

- Only several references shown, block algorithms (ScalAPACK) and communication avoiding algorithms
- CA algorithms exist also for SVD and eigenvalue computation

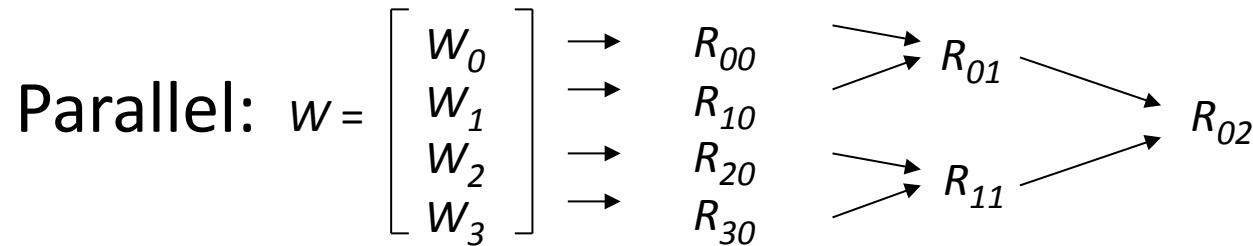
# TSQR: QR factorization of a tall skinny matrix using Householder transformations



J. Demmel, LG, M. Hoemmen, J. Langou, 08

References: Golub, Plemmons, Sameh 88, Pothen, Raghavan, 89, Da Cunha, Becker, Patterson, 02

## Algebra of TSQR



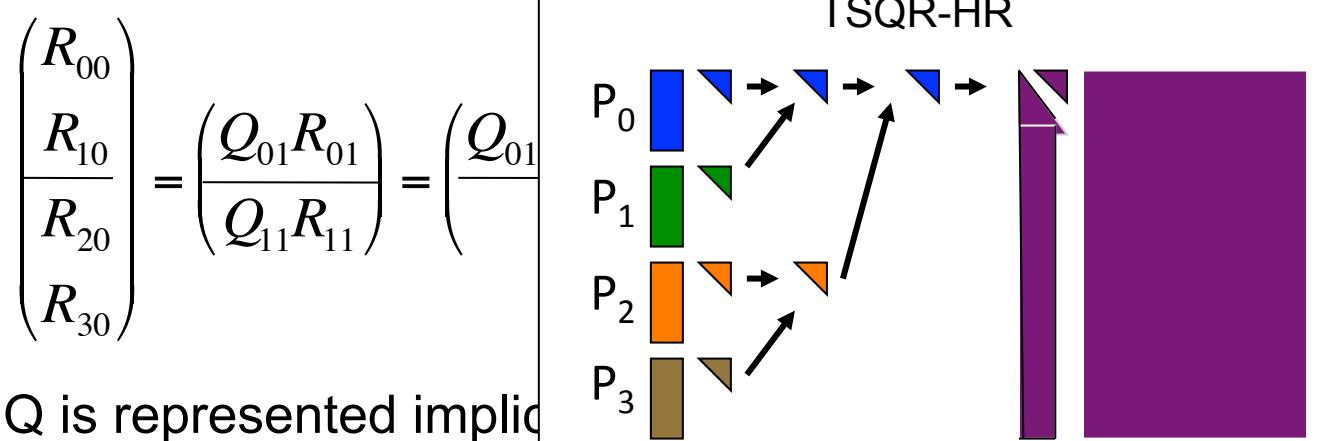
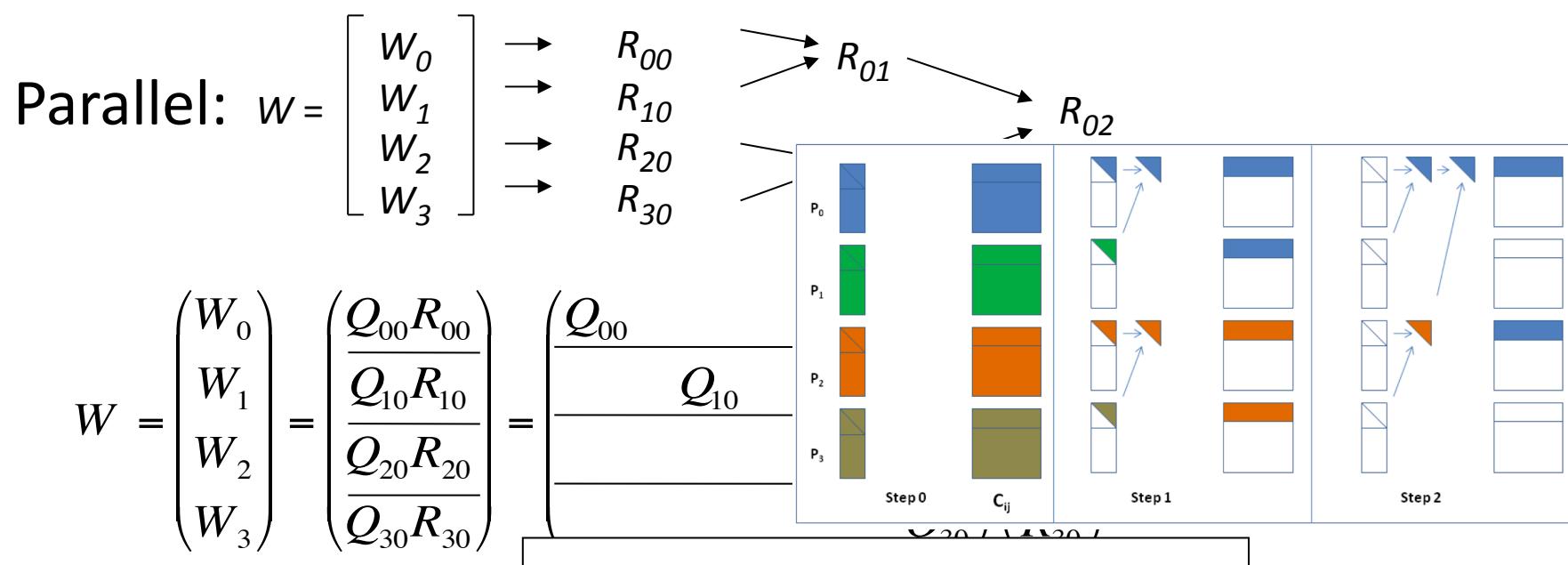
$$W = \begin{pmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \end{pmatrix} = \begin{pmatrix} Q_{00}R_{00} \\ Q_{10}R_{10} \\ Q_{20}R_{20} \\ Q_{30}R_{30} \end{pmatrix} = \begin{pmatrix} Q_{00} \\ Q_{10} \\ Q_{20} \\ Q_{30} \end{pmatrix} \begin{pmatrix} R_{00} \\ R_{10} \\ R_{20} \\ R_{30} \end{pmatrix}$$

$$\begin{pmatrix} R_{00} \\ R_{10} \\ R_{20} \\ R_{30} \end{pmatrix} = \begin{pmatrix} Q_{01}R_{01} \\ Q_{11}R_{11} \end{pmatrix} = \begin{pmatrix} Q_{01} \\ Q_{11} \end{pmatrix} \begin{pmatrix} R_{01} \\ R_{11} \end{pmatrix} \quad \begin{pmatrix} R_{01} \\ R_{11} \end{pmatrix} = Q_{02}R_{02}$$

Q is represented implicitly as a product

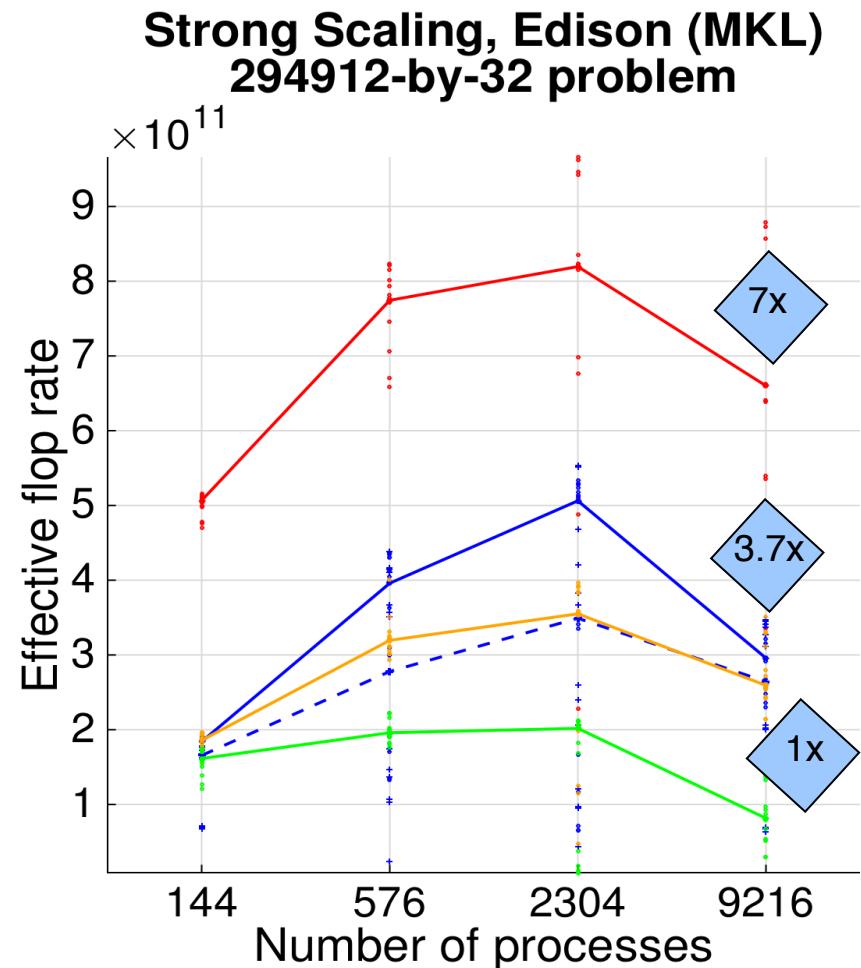
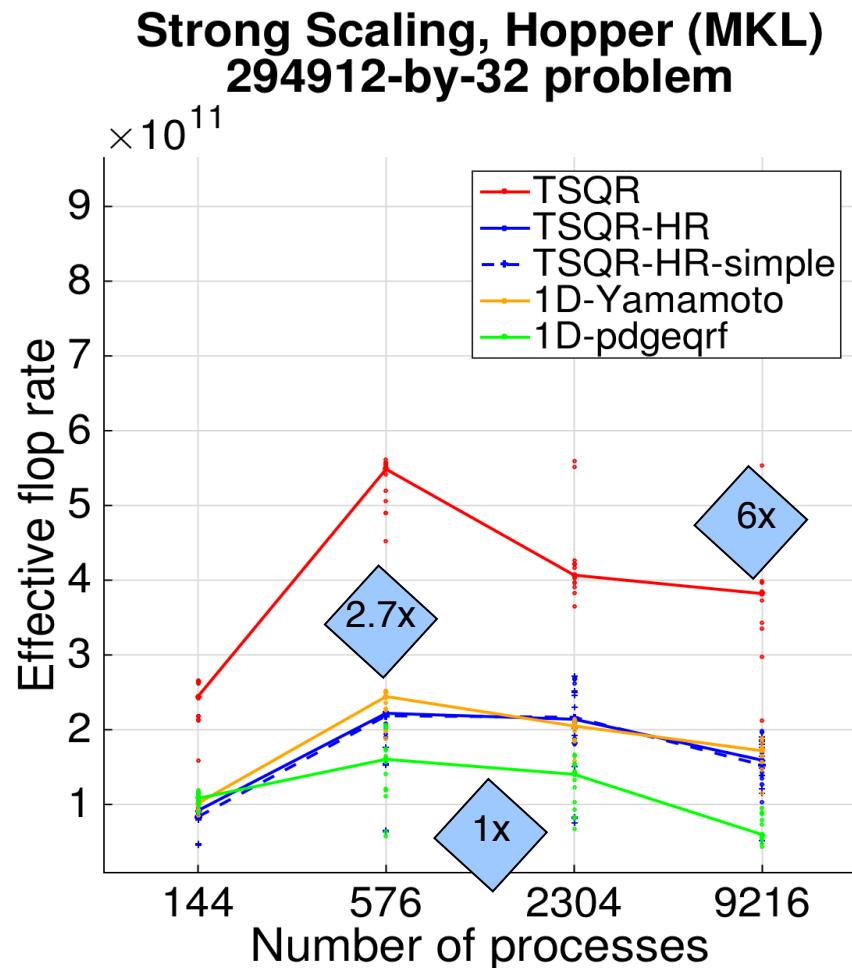
Output:  $\{Q_{00}, Q_{10}, Q_{00}, Q_{20}, Q_{30}, Q_{01}, Q_{11}, Q_{02}, R_{02}\}$

# Algebra of TSQR



Output:  $\{Q_{00}, Q_{10}, Q_{00}, Q_{20}, Q_{30}, Q_{01}, Q_{11}, Q_{02}, R_{02}\}$

## Strong scaling



- Hopper: Cray XE6 (NERSC) – 2 x 12-core AMD Magny-Cours (2.1 GHz)
- Edison: Cray CX30 (NERSC) – 2 x 12-core Intel Ivy Bridge (2.4 GHz)
- Effective flop rate, computed by dividing  $2mn^2 - 2n^3/3$  by measured runtime

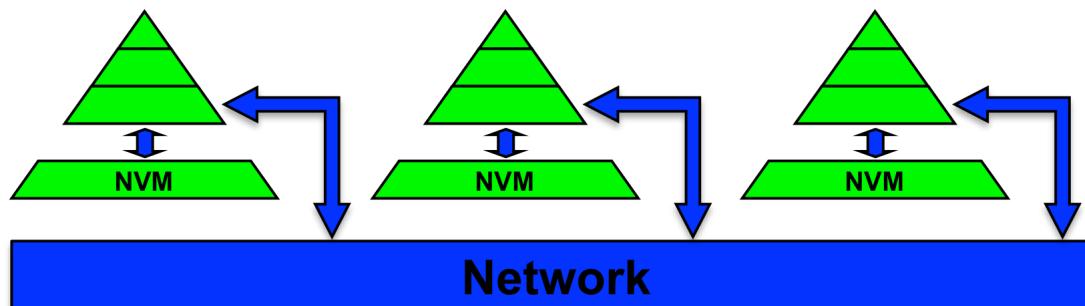
# Parallel write avoiding algorithms

Need to avoid writing suggested by emerging memory technologies, as NVMs:

- Writes more expensive (in time and energy) than reads
- Writes are less reliable than reads

Some examples:

- Phase Change Memory: Reads 25 us latency  
Writes: 15x slower than reads (latency and bandwidth)  
consume 10x more energy
- Conductive Bridging RAM - CBRAM  
Writes: use more energy (1pJ) than reads (50 fJ)
- Gap improving by new technologies such as XPoint and other FLASH alternatives, but not eliminated



# Parallel write-avoiding algorithms

- Matrix A does not fit in DRAM (of size M), need to use NVM (of size  $n^2 / P$ )
- Two lower bounds on volume of communication
  - Interprocessor communication:  $\Omega(n^2 / P^{1/2})$
  - Writes to NVM:  $n^2 / P$
- Result: any three-nested loop algorithm (matrix multiplication, LU,..), must asymptotically exceed at least one of these lower bounds
  - If  $\Omega(n^2 / P^{1/2})$  words are transferred over the network, then  $\Omega(n^2 / P^{2/3})$  words must be written to NVM !
- Parallel LU: choice of best algorithm depends on hardware parameters

	#words interprocessor comm.	#writes NVM
Left-looking	$O((n^3 \log^2 P) / (P M^{1/2}))$	$O(n^2 / P)$
Right-looking	$O((n^2 \log P) / P^{1/2})$	$O((n^2 \log^2 P) / P^{1/2})$

# Plan

- Motivation
- Selected past work on reducing communication
- Brief overview of communication avoiding for dense linear algebra
  - LU, QR, Rank Revealing QR factorizations
  - Progressively implemented in ScaLAPACK, LAPACK
- Communication avoiding for sparse linear algebra
  - Krylov subspace methods
- Conclusions

# Challenge in getting efficient and scalable solvers

- A Krylov solver finds a solution  $x_k$  from  $x_0 + K_k(A, r_0)$ , where

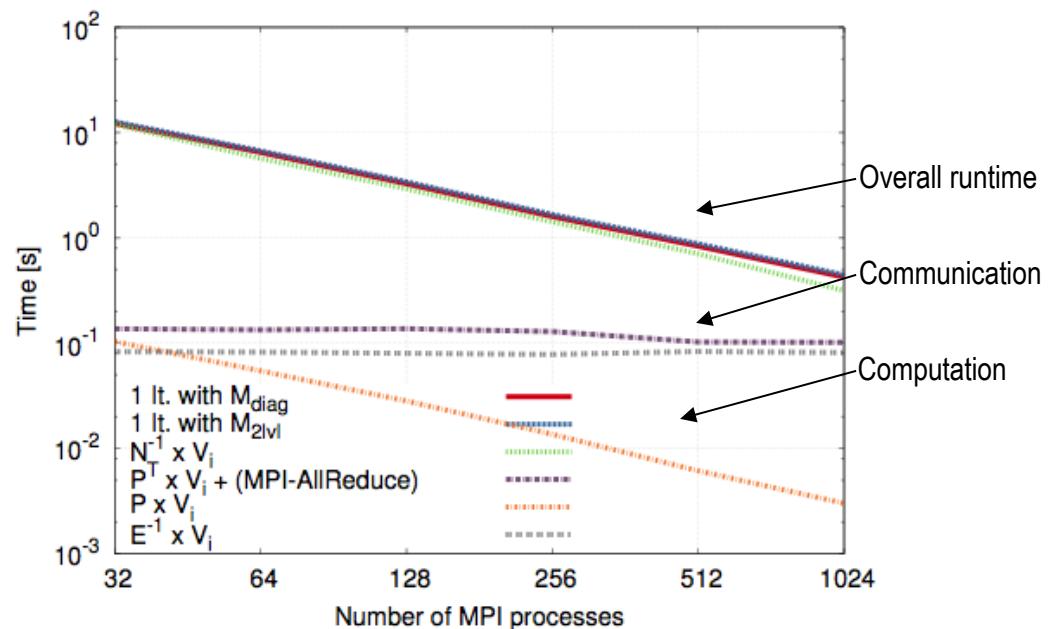
$$K_k(A, r_0) = \text{span} \{r_0, A r_0, \dots, A^{k-1} r_0\}$$

such that the Petrov-Galerkin condition  $b - Ax_k \perp L_k$  is satisfied.

- Does a sequence of  $k$  SpMVs to get vectors  $[x_1, \dots, x_{k-1}]$
- Finds best solution  $x_k$  as linear combination of  $[x_1, \dots, x_{k-1}]$

- Each iteration requires  
Sparse matrix vector product  
-> point to point communication

Dot products for the  
orthogonalization process  
-> global synchronization



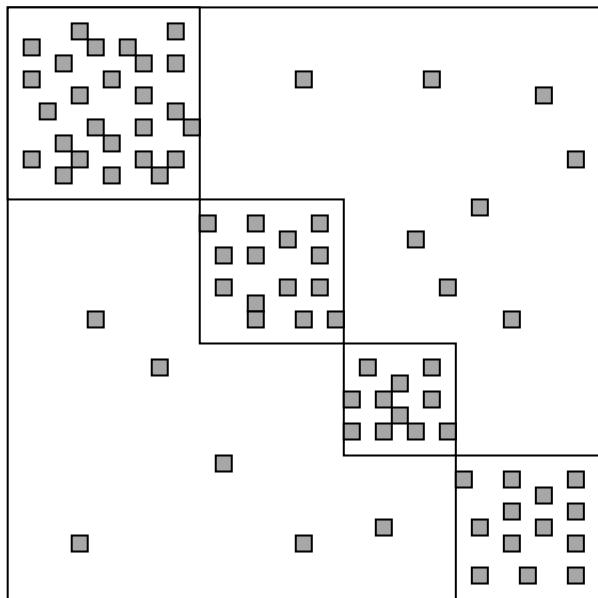
**Map making**, with R. Stompor, M. Szydlarski  
Results obtained on Hopper, Cray XE6, NERSC

## Ways to improve performance

- Improve the performance of sparse matrix-vector product
- Improve the performance of collective communication
- Use preconditioners to decrease the number of iterations till convergence.
- Change numerics – enlarged Krylov methods
  - Decrease the number of iterations to decrease the number of global communications
  - Increase arithmetic intensity – compute sparse matrix-set of vectors product.

# Enlarged Krylov subspaces

- Partition the matrix into  $t$  domains
- Split the initial residual into  $t$  vectors corresponding to the  $t$  domains



$$r_0 \rightarrow T(r_0) = \begin{bmatrix} * & 0 & 0 \\ \vdots & \vdots & \vdots \\ * & 0 & 0 \\ 0 & * & 0 \\ \vdots & \vdots & \vdots \\ 0 & * & 0 \\ & \ddots & \\ 0 & 0 & * \\ \vdots & \vdots & \vdots \\ 0 & 0 & * \end{bmatrix}$$

- Generate  $t$  new basis vectors, obtain an enlarged Krylov subspace

$$K_{t,k}(A, r_0) = \text{span}\{T(r_0), AT(r_0), \dots, A^{k-1}T(r_0)\}$$

- Search for the solution of the system  $Ax = b$  in  $K_{t,k}(A, r_0)$

## Numerical results

- Block Jacobi preconditioner (1024 blocks)
- Stopping criterion  $10^{-6}$
- Initial block size 32

	red. size	PCG		EK-CG			$\dim(\mathcal{K}_k^\Delta)$
		iter	error	iter	error		
SKY2D	✗	655	9.0e-08	57	7.3e-12	1824	1824
	✓	655	9.0e-08	59	7.8e-12	1546	
Ela3D100	✗	870	3.5e-09	109	3.2e-11	3488	3488
	✓	870	3.5e-09	116	5.3e-11	2384	
Ela2D200	✗	4551	1.2e-09	253	1.8e-10	8096	8096
	✓	4551	1.2e-09	266	1.8e-10	6553	

# Conclusions

- Need to redesign/reformulate algorithms to reduce communication
  - Derive when possible lower bounds on communication
  - Minimize communication at the cost of redundant computation
  - Communication avoiding algorithms often faster than conventional algorithms in practice
- Remains a lot to do for sparse linear algebra
  - Communication bounds, communication optimal algorithms
  - Preconditioners - limited by memory and communication, not flops
- And BEYOND

# Collaborators, funding

## Collaborators:

- Inria Alpines: A. Ayala, S. Cayrols, S. Donfack, A. Khabou, M. Jacquelin, S. Moufawad, F. Nataf, CNRS, O. Tissot, H. Xiang, S. Yousef
- J. Demmel (UC Berkeley), G. Ballard (Sandia), B. Gropp (UIUC), M. Gu (UC Berkeley), M. Hoemmen (Sandia), N. Knight (NYU), J. Langou (CU Denver), V. Kale (UIUC), P. Henon (Total), P. Ricoux (Total)

Thanks to: EC H2020 NLAFET, Total, ANR Petal and Petalh projects, ANR Midas, Digiteo Xscale NL, COALA INRIA funding

## Further information:

<http://www-rocq.inria.fr/who/Laura.Grigori/>



# References

Results presented from:

- J. Demmel, L. Grigori, M. F. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, UCB-EECS-2008-89, 2008, SIAM journal on Scientific Computing, Vol. 34, No 1, 2012.
- L. Grigori, J. Demmel, and H. Xiang, *Communication avoiding Gaussian elimination*, Proceedings of the IEEE/ACM SuperComputing SC08 Conference, November 2008.
- L. Grigori, J. Demmel, and H. Xiang, *CALU: a communication optimal LU factorization algorithm*, SIAM. J. Matrix Anal. & Appl., 32, pp. 1317-1350, 2011.
- L. Grigori, P.-Y. David, J. Demmel, and S. Peyronnet, *Brief announcement: Lower bounds on communication for sparse Cholesky factorization of a model problem*, ACM SPAA 2010.
- S. Donfack, L. Grigori, and A. Kumar Gupta, *Adapting communication-avoiding LU and QR factorizations to multicore architectures*, Proceedings of IEEE International Parallel & Distributed Processing Symposium IPDPS, April 2010.
- S. Donfack, L. Grigori, W. Gropp, and V. Kale, *Hybrid static/dynamic scheduling for already optimized dense matrix factorization* , Proceedings of IEEE International Parallel & Distributed Processing Symposium IPDPS, 2012.
- J. Demmel, L. Grigori, M. Gu, H. Xiang, Communication avoiding rank revealing QR factorization with column pivoting, SIAM J. Matrix Anal. & Appl, Vol. 36, No. 1, pp. 55-89, 2015.
- G. Ballard, J. Demmel, L. Grigori, M. Jacquelin, N. Knight, J. D. Nguyen, and E. Solomonik *Reconstructing Householder vectors for QR factorization*, Journal of Parallel and Distributed Computing, in press, 2015.
- L. Grigori, S. Moufawad, F. Nataf, *Enlarged Krylov Subspace Conjugate Gradient Methods for Reducing Communication* , INRIA TR 8597.
- H. Al Daas, L. Grigori, P. Henon, P. Ricoux, Enlarged GMRES, In preparation.