FP7 Support Action - European Exascale Software Initiative

DG Information Society and the unit e-Infrastructures

Addressing the Challenge of Exascale

European Exascale Software Initiative EESI

Towards Exascale roadmap implementation

EESI2 – Recommendations

Algorithms for Communication and Data-Movement Avoidance

Laura Grigori
Inria/UPMC
I. Duff, S. Filippone, H. Ltaief, U. Rude

Major obstacle: the communication wall

- Runtime of an algorithm is the sum of:
  - #flops x time_per_flop
  - #words_moved / bandwidth
  - #messages x latency

- Time to move data >> time per flop

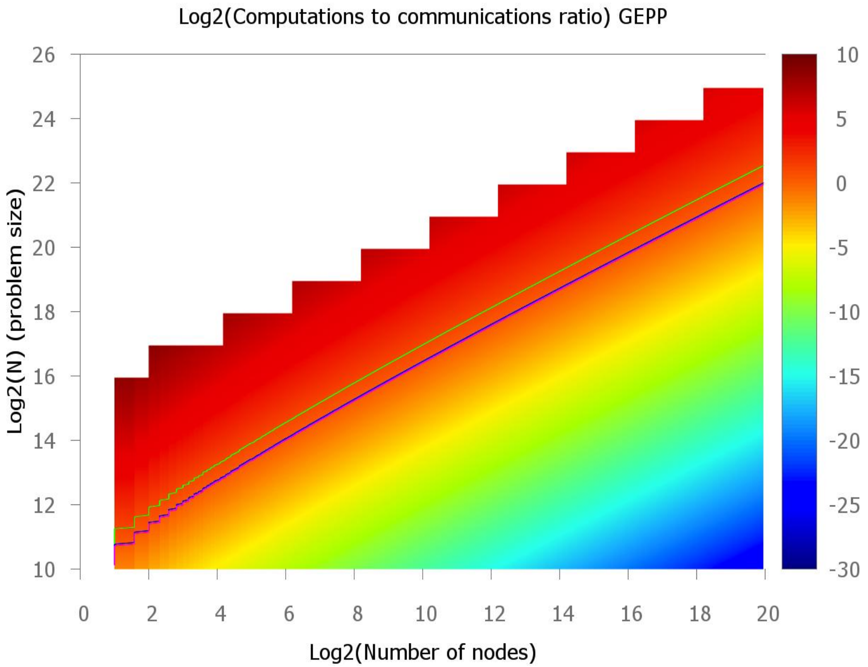
  Gap steadily and exponentially growing over time

Annual improvements			
Time/flop		Bandwidth	Latency
59%	Network	**26%**	**15%**
	DRAM	**23%**	**5%**

"We are going to hit the memory wall, unless something basic changes"
[W. Wulf, S. McKee, 95]

Previous work on reducing communication

- Hiding
  - Overlap communication and computation, at most a factor of 2 speedup
- Ghosting
  - Store redundantly data from neighboring processors for future computations
- Scheduling
  - Block algorithms for linear algebra
    - Barron and Swinnerton-Dyer, 1960
    - ScaLAPACK, Blackford et al 97



Recommendation

Unlock the exa-scalability potential

of numerical algorithms

by

integrating the communication dimension into the numerical algorithmic design

Recommendation

- Design communication avoiding algorithms for linear algebra and beyond
  - For all critical stages of computationally intensive applications, e.g. mesh generation algorithms, parallel in time methods.
- Focus on operations that are at the intersection with the data mining community
- Enable the development of sustainable software that implements this new generation of communication avoiding numerical algorithms

- Enable leadership of European researchers in selected areas
  - e.g. CA algorithms, H-matrices, Fast Multipole Methods
- Enable the coordination and federation of multiple efforts to reach a critical mass.

Example: CA for Linear Algebra

Communication avoiding algorithms - a novel perspective for numerical linear algebra

- Asymptotically reduce communication
- Minimize volume of communication / number of messages
- Allow redundant computations (preferably as a low order term)

Previous results: matrix multiply, using $2n^3$ flops (bandwidth)

- Hong/Kung (1981), Irony/Tishkin/Toledo (2004)

Lower bounds for LU and QR factorizations

- For bandwidth and **latency** [Demmel, LG, Hoemmen, Langou 2008, SISC]

  #words_moved $\geq \Omega(n^2 / P^{1/2})$ #messages $\geq \Omega(P^{1/2})$
- Extended to almost all direct dense linear algebra [Demmel et al, 2009]

Example: CA for Linear Algebra

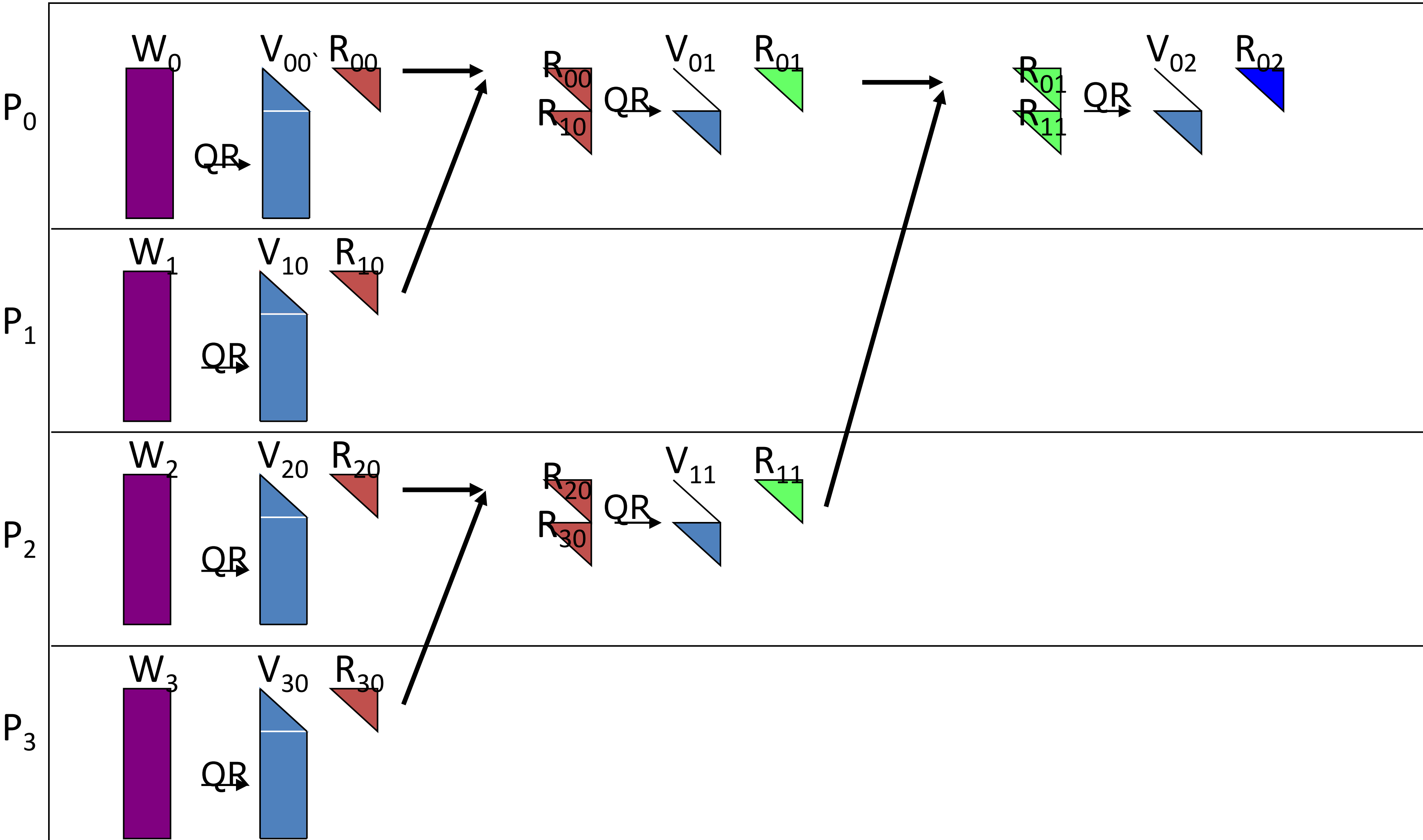
Algorithm	Minimizing #words (not #messages)	Minimizing #words and #messages
Cholesky	ScaLAPACK	ScaLAPACK
LU	ScaLAPACK uses partial pivoting	[LG, Demmel, Xiang, 08] [Khabou, Demmel, LG, Gu, 12] uses tournament pivoting
QR	ScaLAPACK	[Demmel, LG, Hoemmen, Langou, 08] uses different representation of Q
RRQR	ScaLAPACK	[Demmel, LG, Gu, Xiang 13] uses tournament pivoting, 3x flops

- LAPACK and ScaLAPACK sub-optimal

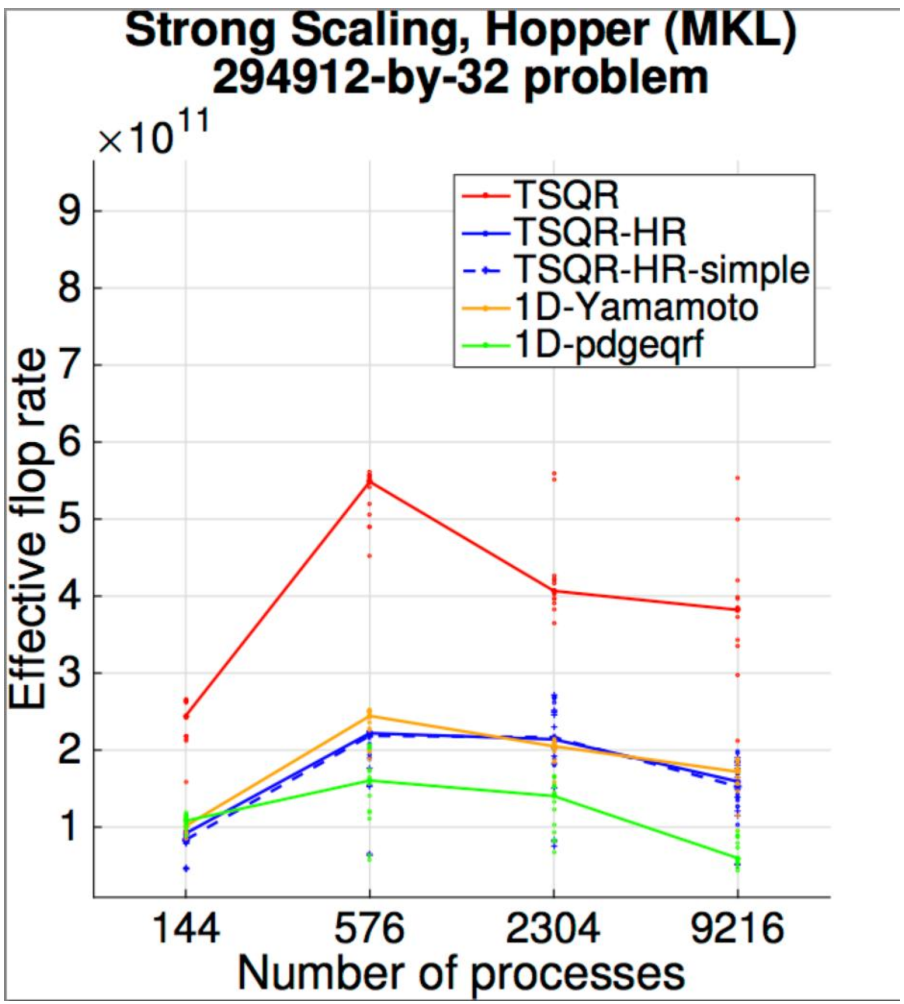
⇒ Communication avoiding algorithms are optimal for dense LA

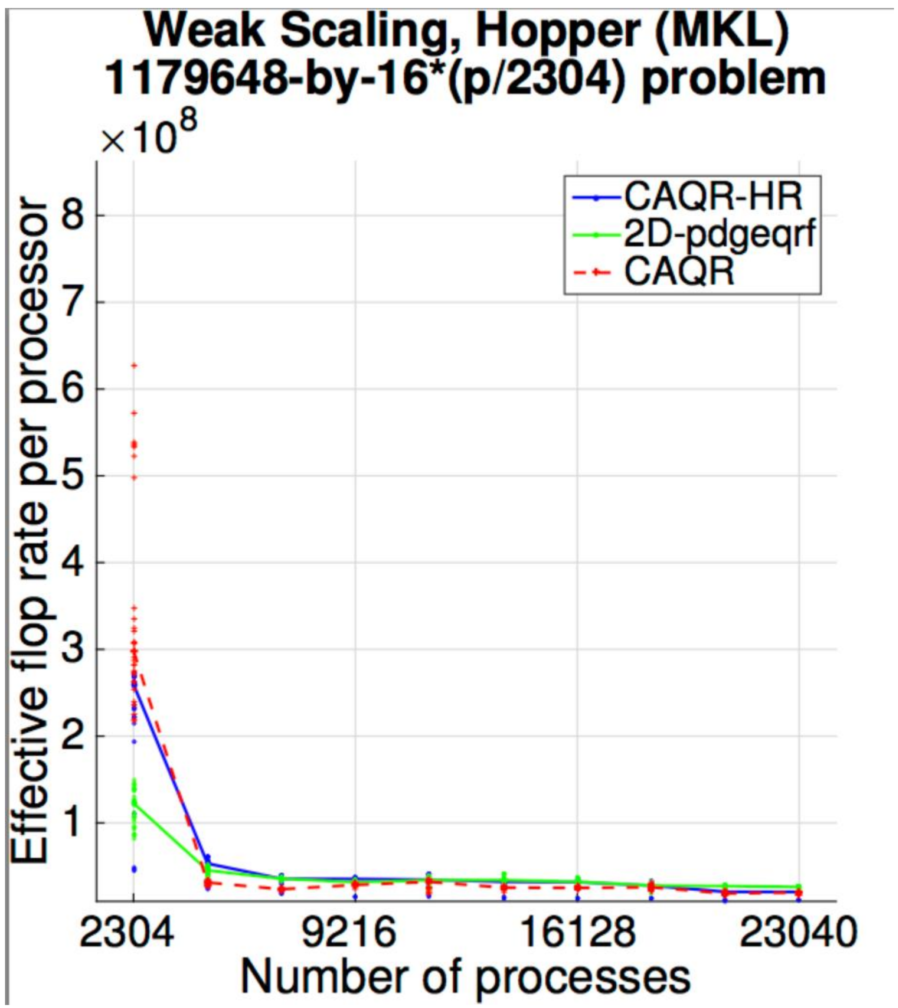
- As stable as classic algorithms
- CAQR, CALU, RRQR considered/implemented by MKL- Intel, Cray, IBM

Parallel TSQR



Performance of TSQR and CAQR





Cray XE6, 2 12-core AMD Magny-Cours (2.1 GHz)

Challenge in getting scalable solvers

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

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

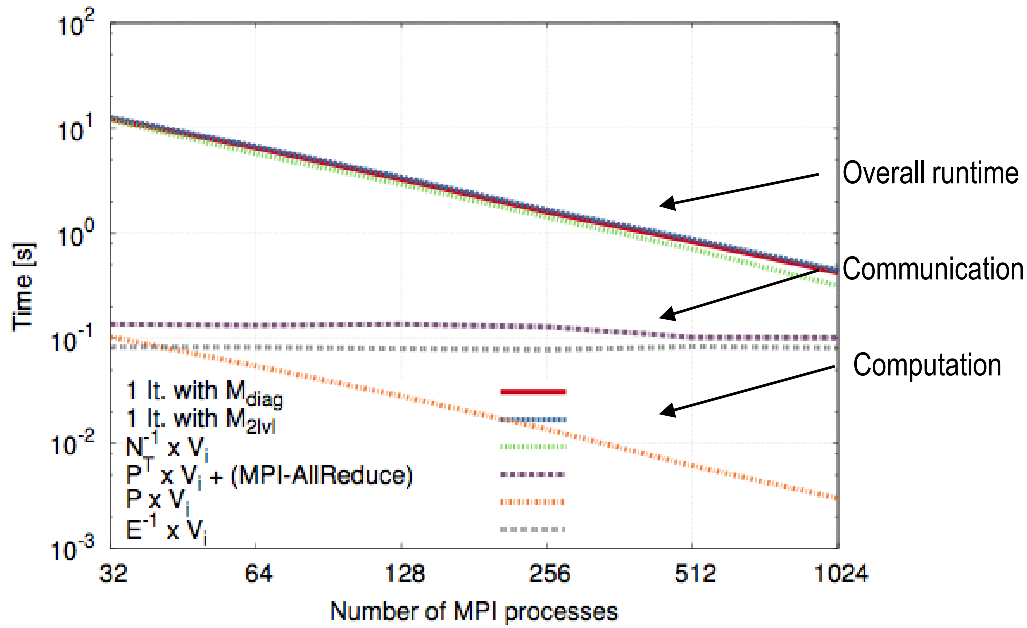
Each iteration requires

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

Our goal:

- Decrease the number of iterations to decrease the number of global communications
- Increase arithmetic intensity

Enlarged Krylov subspace solvers

- Partition the matrix into t domains
- At k-th iteration,
  - split the residual r_{k-1} into t vectors corresponding to the t domains,

$$r_{k-1} \rightarrow T(r_{k-1}) = \begin{bmatrix} * & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ * & 0 & & 0 \\ 0 & * & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & * & & 0 \\ & & \ddots & \\ 0 & 0 & & * \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & * \end{bmatrix}, T_s(r_{k-1}) = \{T(r_{k-1})(:,1), \ldots T(r_{k-1})(:,t)$$

  - generate t new basis vectors, obtain an enlarged Krylov subspace

$$\mathscr{K}_{t,k}(A, r_0) \quad = \quad span\{T_s(r_0), AT_s(r_0), A^2T_s(r0), ..., A^{k-1}T_s(r_0)\}$$

  - search for the solution of the system $Ax = b$ in $\mathscr{K}_{t,k}(A, r_0)$

Enlarged Krylov subspace methods

NLAFET H2020-FETHPC-2014
Umea University, Inria,
University of Manchester, STFC/RAL

New generation of linear algebra
algorithms and software libraries:

- Algorithms: minimize communication, reduce energy consumption, resilience
- Management of parallelism
- Auto-tuning

	CG		SRE-CG	
Pa	Iter	Err	Iter	Err
SKY3D				
8	902	1E-5	211	1E-5
16	902	1E-5	119	9E-6
32	902	1E-5	43	4E-6
ANI3D				
2	4187	4e-5	875	7e-5
4	4146	4e-5	673	8e-5
8	4146	4e-5	449	1e-4
16	4146	4e-5	253	2e-4
32	4146	4e-5	148	2e-4
64	4146	4e-5	92	1e-4
ELAST3D				
2	1098	1e-7	652	1e-7
4	1098	1e-7	445	1e-7
8	1098	1e-7	321	8e-8
16	1098	1e-7	238	4e-8
32	1098	1e-7	168	5e-8
64	1098	1e-7	116	1e-8