Honoring: Yousuff Hussaini

Scalable Parallel Sparse Matrix Computations

Ahmed Sameh

Computer Science, Purdue University

September 28, 2012

Joint work with:

M. Manguoglu, F. Saied, O. Schenk

Support: ARO, Intel, NSF.

Sparse Matrix Computations

- *Importance*
  - *They arise in:*
    - *computational engineering applications*
    - *network analysis*
    - *analysis of large data sets*
  - *They give rise to indirect addressing which often leads to significant performance degradation on various parallel architectures.*
  - *Performance of sparse matrix primitives and algorithms on parallel architectures often governs the overall performance of many applications.*

Sparse Matrix Computations…

- *Fresh ideas for designing parallel sparse matrix algorithms are needed:*
  - *the availability of various parallel programming tools proved to be insufficient to assure high performance in implementing familiar sequential sparse matrix kernels and algorithms.*

The focus here is on the design of sparse matrix computation schemes that:

- *exhibit ample concurrency,*
- *address memory management bottlenecks within a node, and*
- *minimize internode communications.*

Outline

- *Parallel sparse matrix primitives:*
  - *matrix reordering*
  - *sparse matrix-vector (multivector) multiplication*
- *Parallel sparse matrix algorithms for two fundamental linear algebra problems with wide applications:*
  - *linear systems of equations*
  - *symmetric algebraic eigenvalue problems*

Computing Platform

- *Endeavor Intel cluster with infiniband interconnect*
- *Each node contains 12 to 80 cores*
- *Local memory per node ≤ 48 GB*
- *Architectures ranging from Nehalem to Sandy Bridge.*
- *Most recent version of `MKL` and Olaf Schenk's direct sparse system solver -- `PARDISO`.*

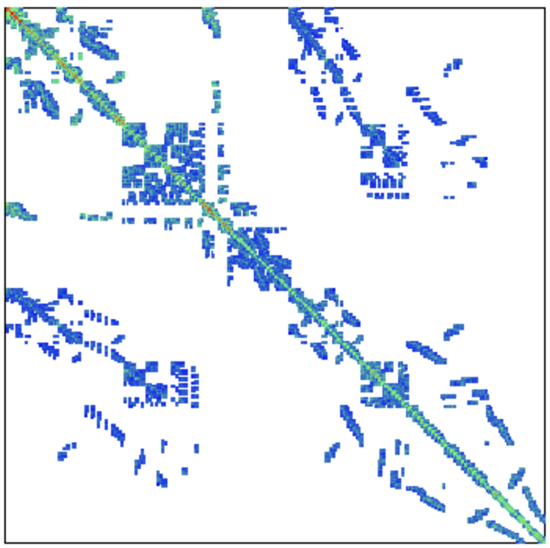
Two important sparse matrix primitives

Primitive 1: Reordering

- *Parallel sparse matrix reordering enables:*
  - *Faster sparse matrix-vector multiplications.*
  - *Extracting more effective parallel preconditioners for iterative sparse linear system solvers.*

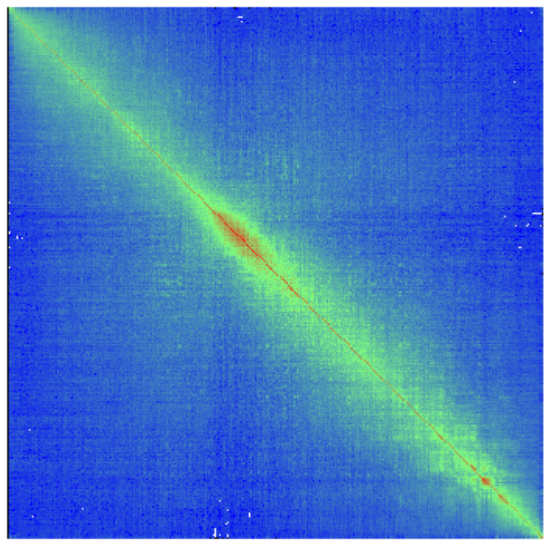
UFL: smt -- structural mechanics

N: 25,710 NNZ: 3,749,582



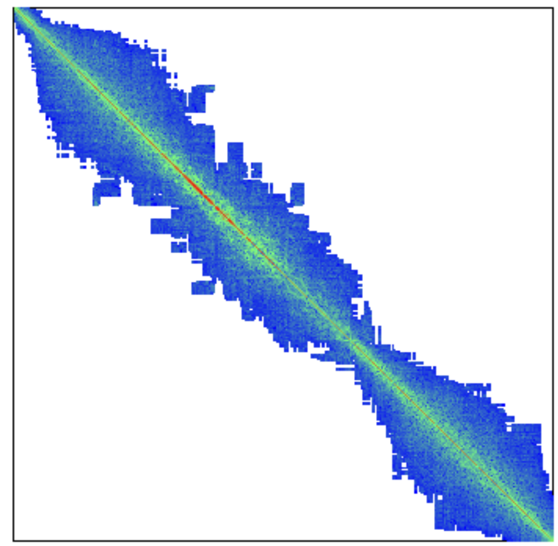
Original matrix

after HSL-MC73



After MC73

after TraceMIN-Fiedler

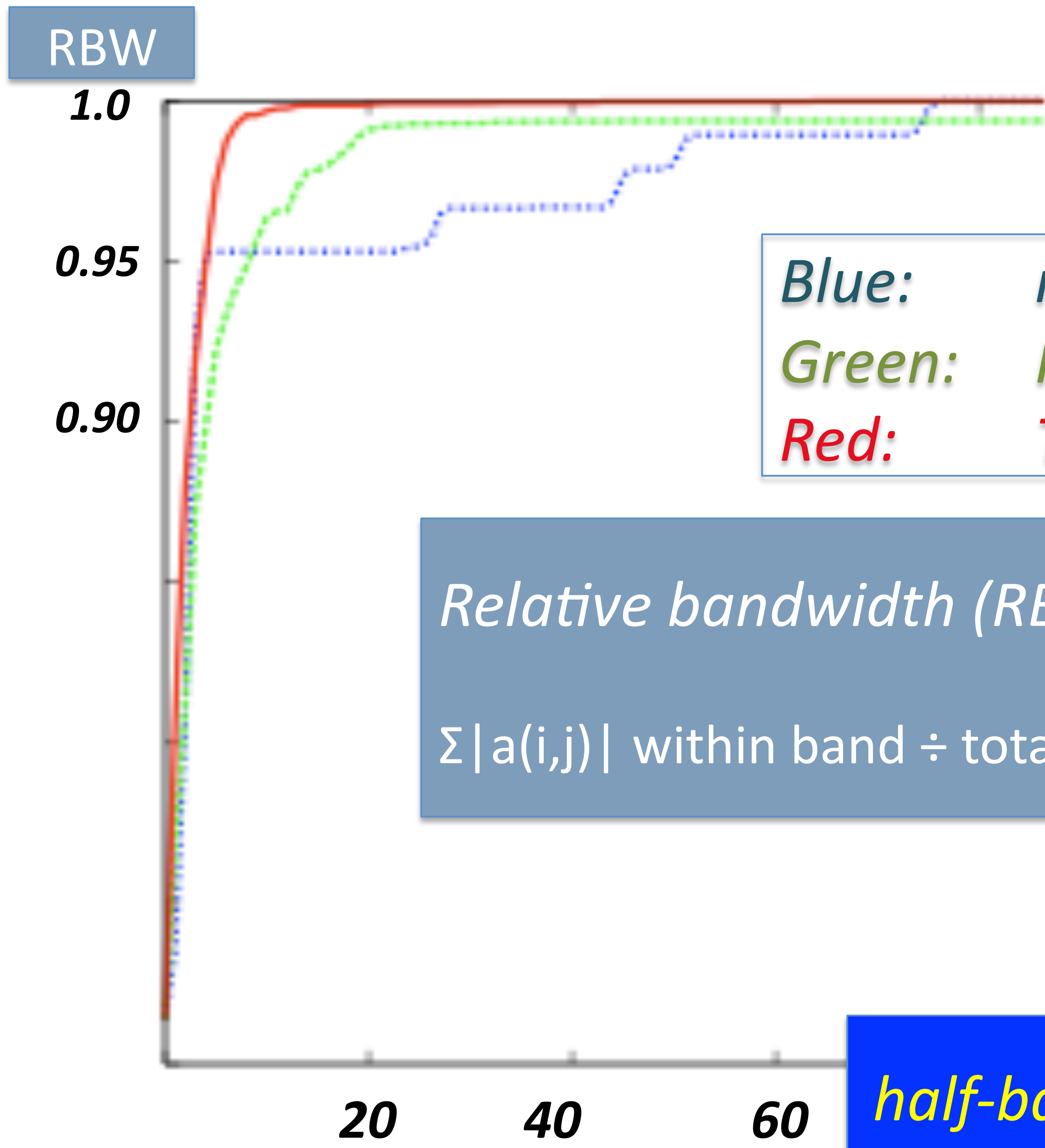


After TraceMin-Fiedler

obtaining the Fiedler vector via the eigensolver: TraceMIN

(Wisniewski and A.S. -- SINUM, ’82)

Reordering
A:= BCSSTK22



half-bandwidth: k

Blue:	*no reordering*
Green:	*HSL-MC73*
Red:	*TraceMIN-Fiedler*

Relative bandwidth (RBW):

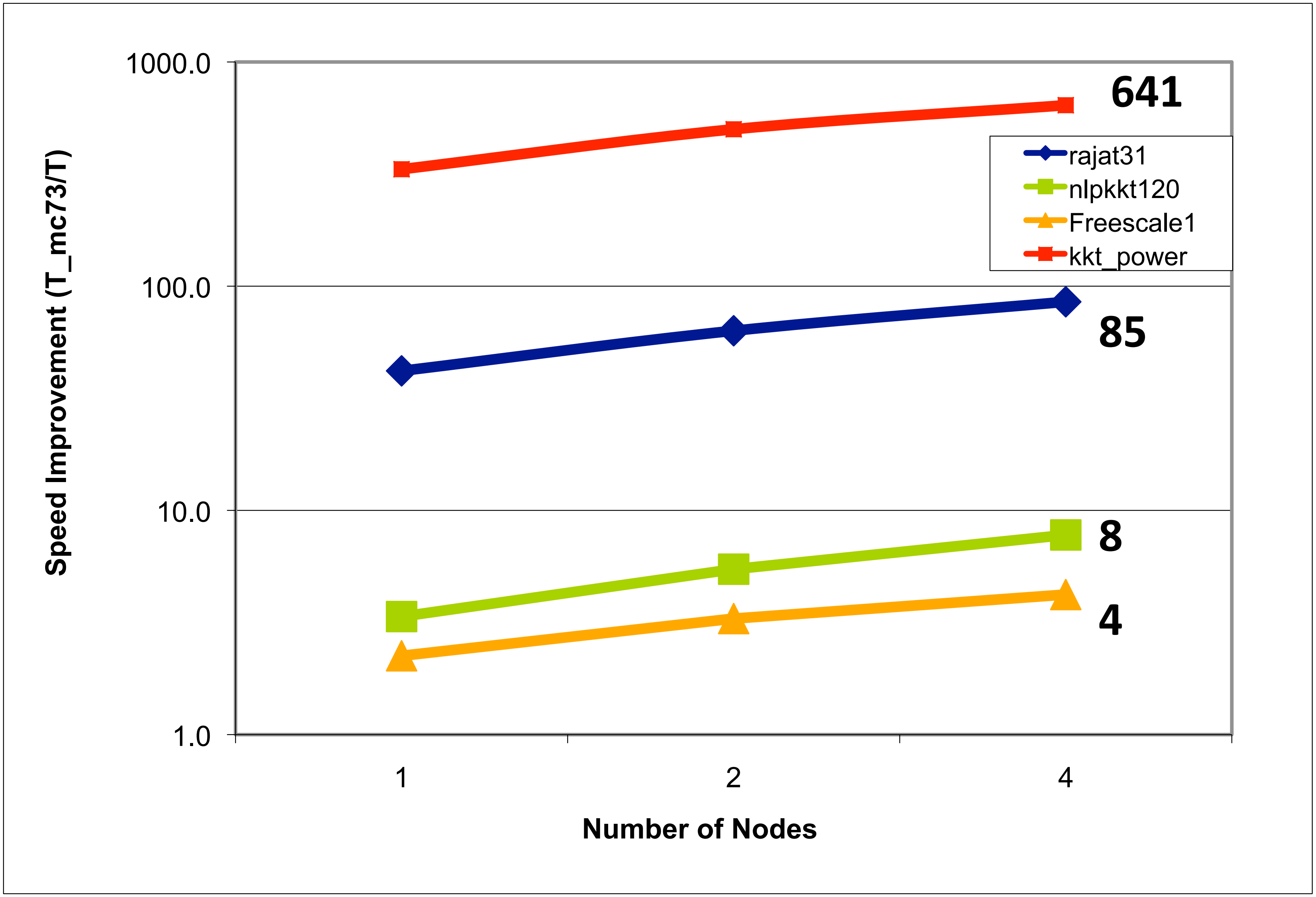
Σ|a(i,j)| within band ÷ total Σ |a(i,j)|

Parallel Scalability of our weighted spectral reordering scheme

TraceMIN-Fiedler vs. HSL-MC73 (Pothen & Simon)

Matrix Group/Name	n	nnz	symmetric
1. Rajat/rajat31	4,690,002	20,316,253	no
2. Schenk/nlpkkt120	3,542,400	95,117,792	yes
3. Freescale/Freescale1	3,428,755	17,052,626	no
4. Zaoui/kkt_power	2,063,494	12,771,361	yes

T(HSL-MC73) ÷ T(TraceMIN-Fiedler)



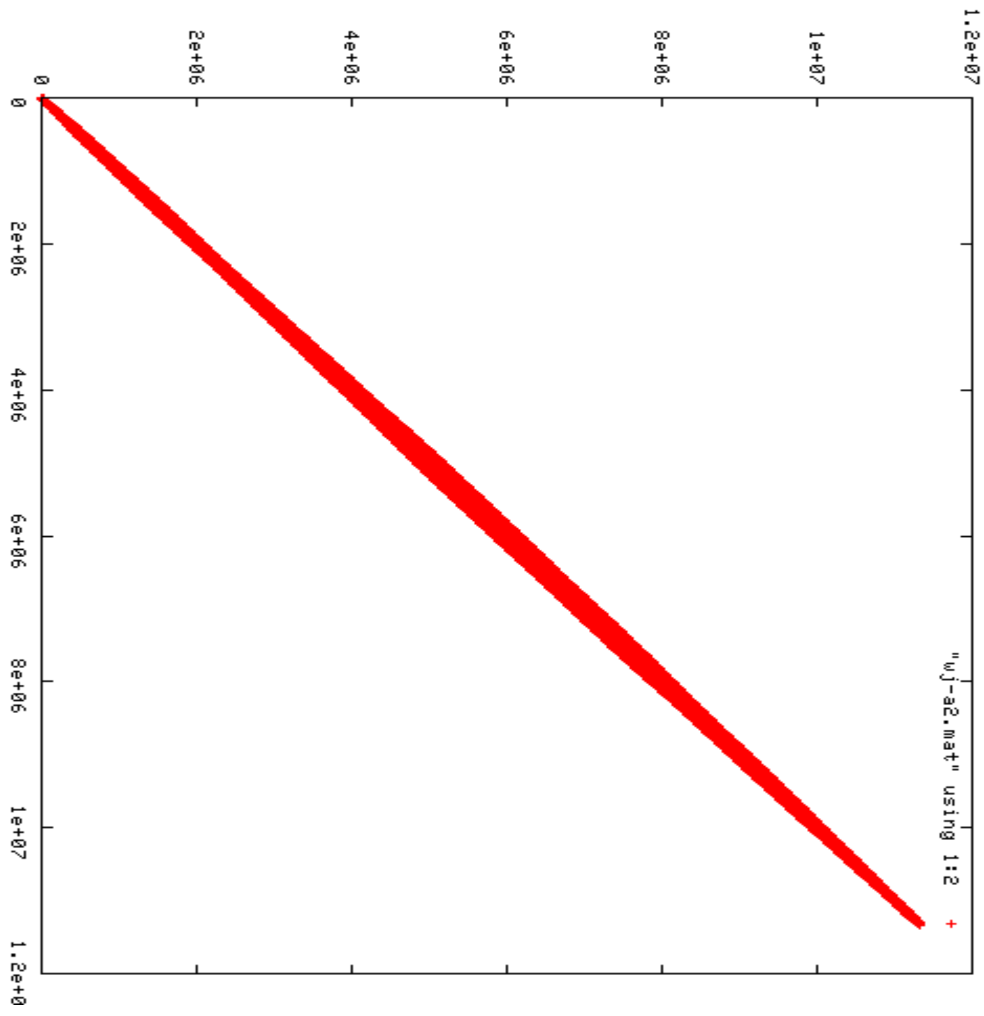
Weighted spectral reordering of MEMS benchmark 1

System size:

N = 11,333,520

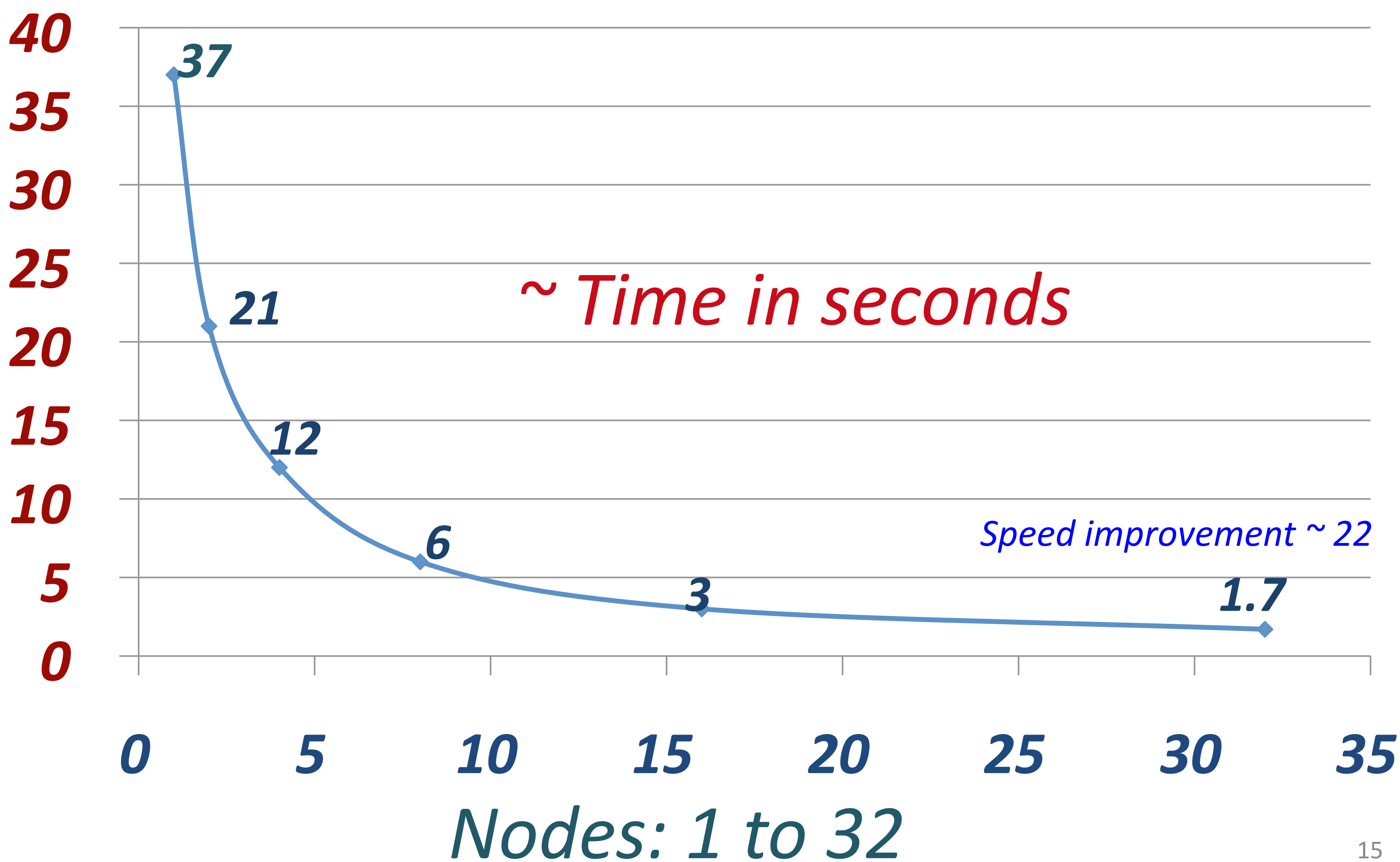
# of nonzeros:
61,026,416

bandwidth:
334,613



14

Scalability of TraceMin-Fiedler



Primitive 2:
Matrix-vector multiplication (MATVEC)

- $P A P' = B + E$ (symmetric reordering)
  - A: sparse
  - B: banded, E: sparse of low rank
- $y = A * x$
  1. $u = P * x$
  2. $v = B * u$; $w = E * u$
  3. $z = (v + w)$
  4. $y = P' * z$

High performance: $B * u$
Low cost: $u = P * x$ & $y = P' * z$

Target Computational Loop

Integration

Newton iteration

Linear system solvers

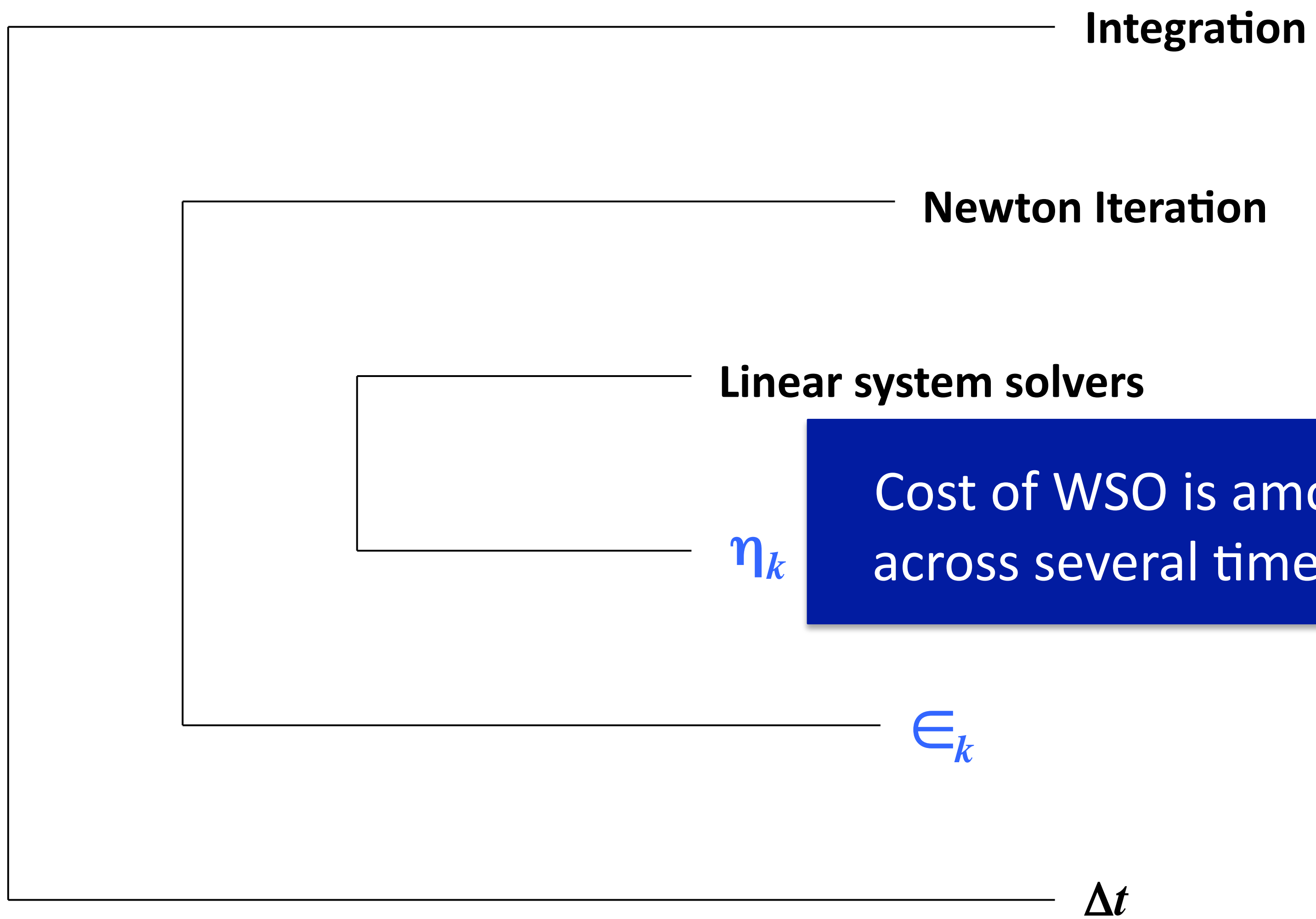
require relative residuals of $O(10^{-5}$ or $10^{-6})$

η_k

$\in_k$

Δt

How expensive is spectral reordering?



Cost of WSO is amortized across several time steps.

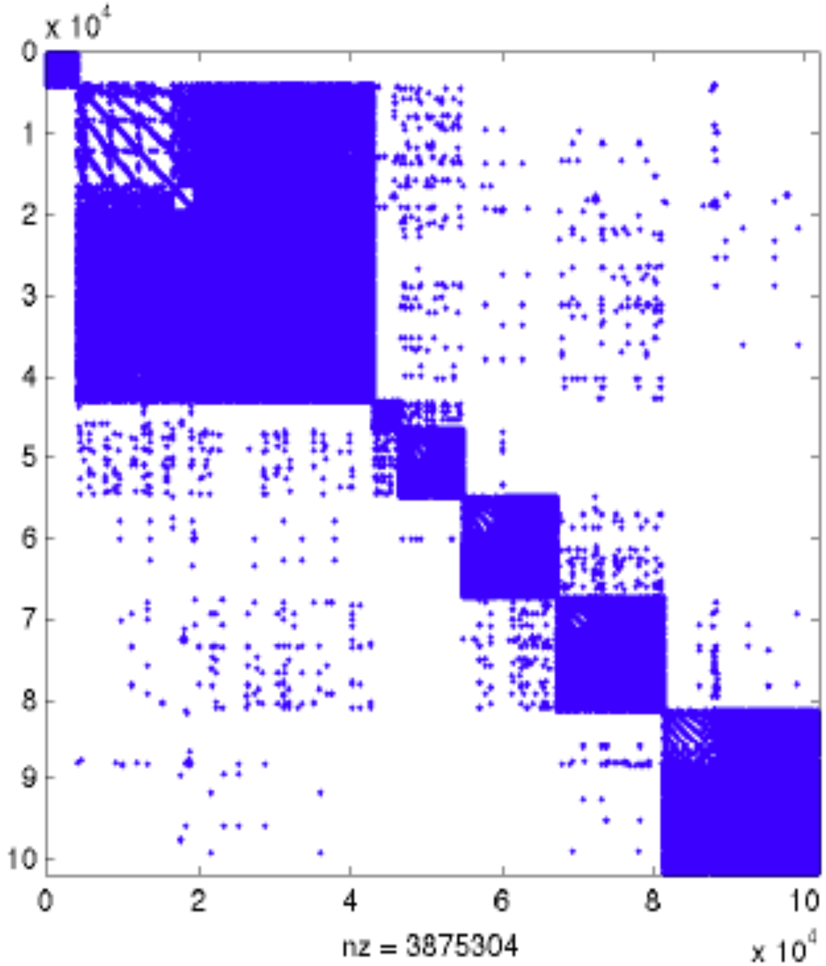
will return to this issue later

Impact of a faster MATVEC on a time-dependent problem:

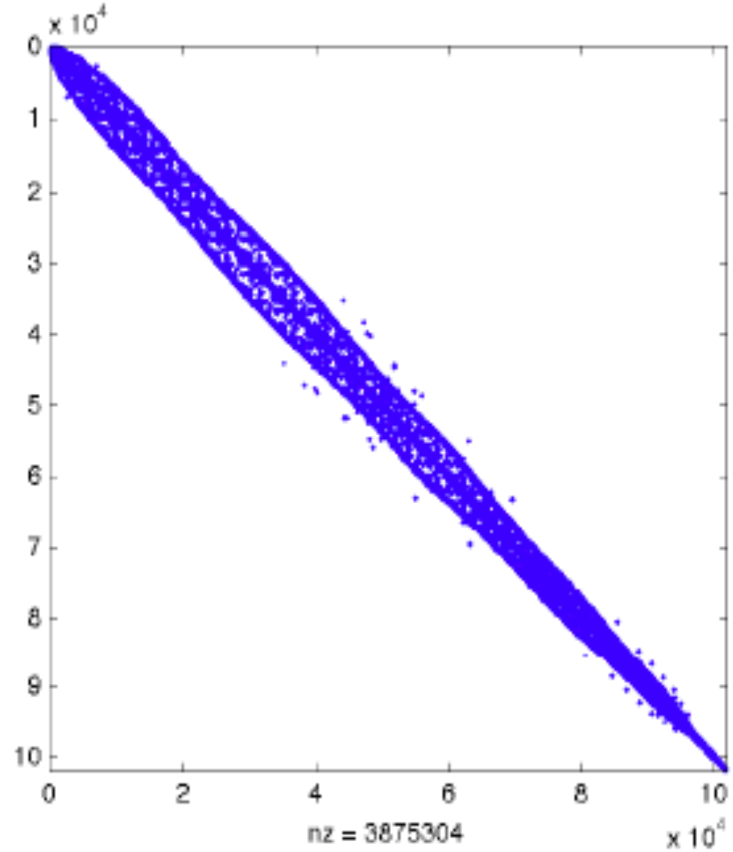
Animation

Solving s.p.d. systems via a preconditioned C.G. scheme at each time step

Permutation of time-step #1 applied to time-step #2

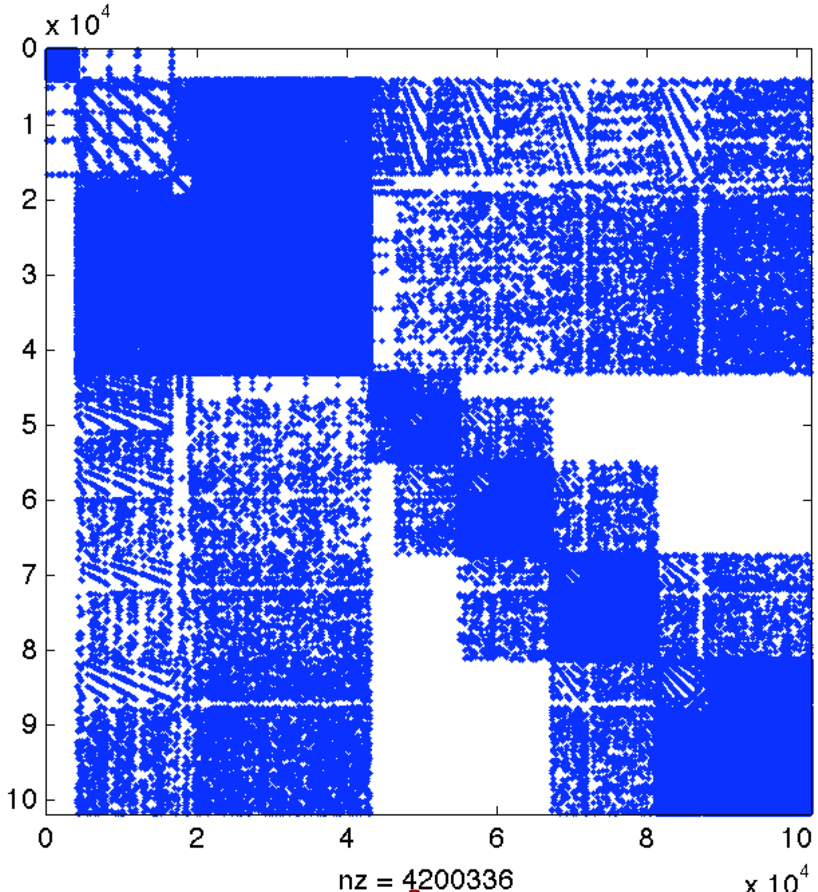


A_2

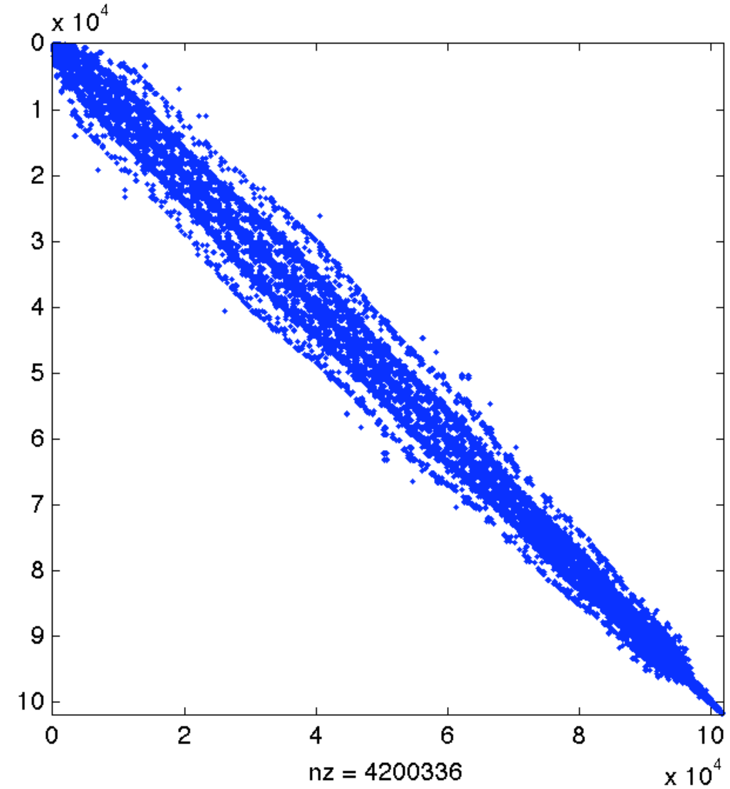


$C_2 = P_1 A_2 P_1^T$

Permutation of time-step#1 applied to time-step#16



A_{16}



$C_{16} = P_1 \, A_{16} \, P_1^T$

Time in seconds to process one frame (16 time steps)

ISV	MKL Matvec	our Matvec after Reorder.	our Matvec after Reorder.	our Matvec after Reorder.
8-core Nehalem	12-core Westmere	12-core Westmere	40-core Westmere	16 12-core nodes Westmere)
3.04 1	1.32 2.3	0.84 3.6	0.30 10	0.14 22

A Hybrid Sparse Linear System Solver: PSPIKE

Target Computational Loop

Integration

Newton iteration

Linear system solvers

require relative residuals of $O(10^{-5}$ or $10^{-6})$

η_k

ϵ_k

Δt

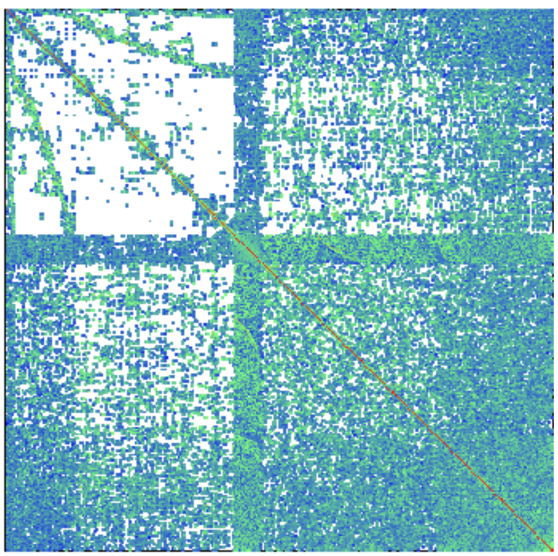
PSPIKE

- *Systematic approach for solving sparse linear systems:*
  - *Apply our parallel spectral reordering scheme via our eigensolver `TraceMIN_Fiedler`.*
  - *Extract preconditioner*
  - *Use the nested iterative scheme:*
    - *Outer Krylov subspace method*
    - *Inner modified Richardson splitting***

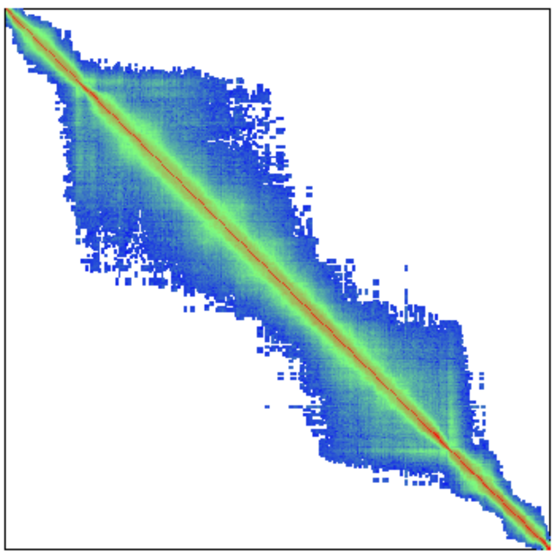
*** the multicore sparse direct solver PARDISO is applied simultaneously to handle several smaller systems one per node*

UFL: f2 -- structural mechanics

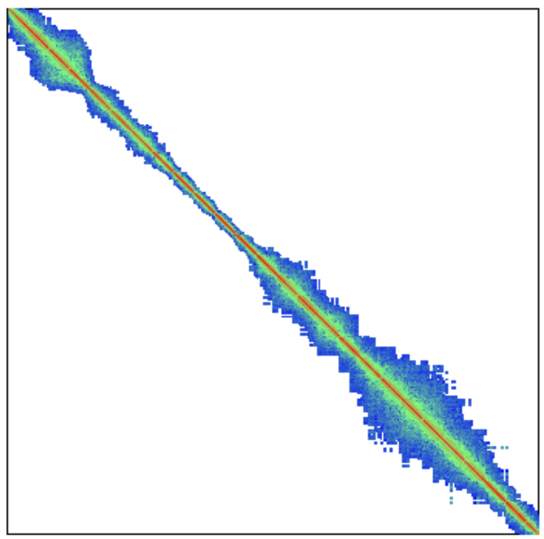
N: 71,505 NNZ: 5,294,285



Original matrix



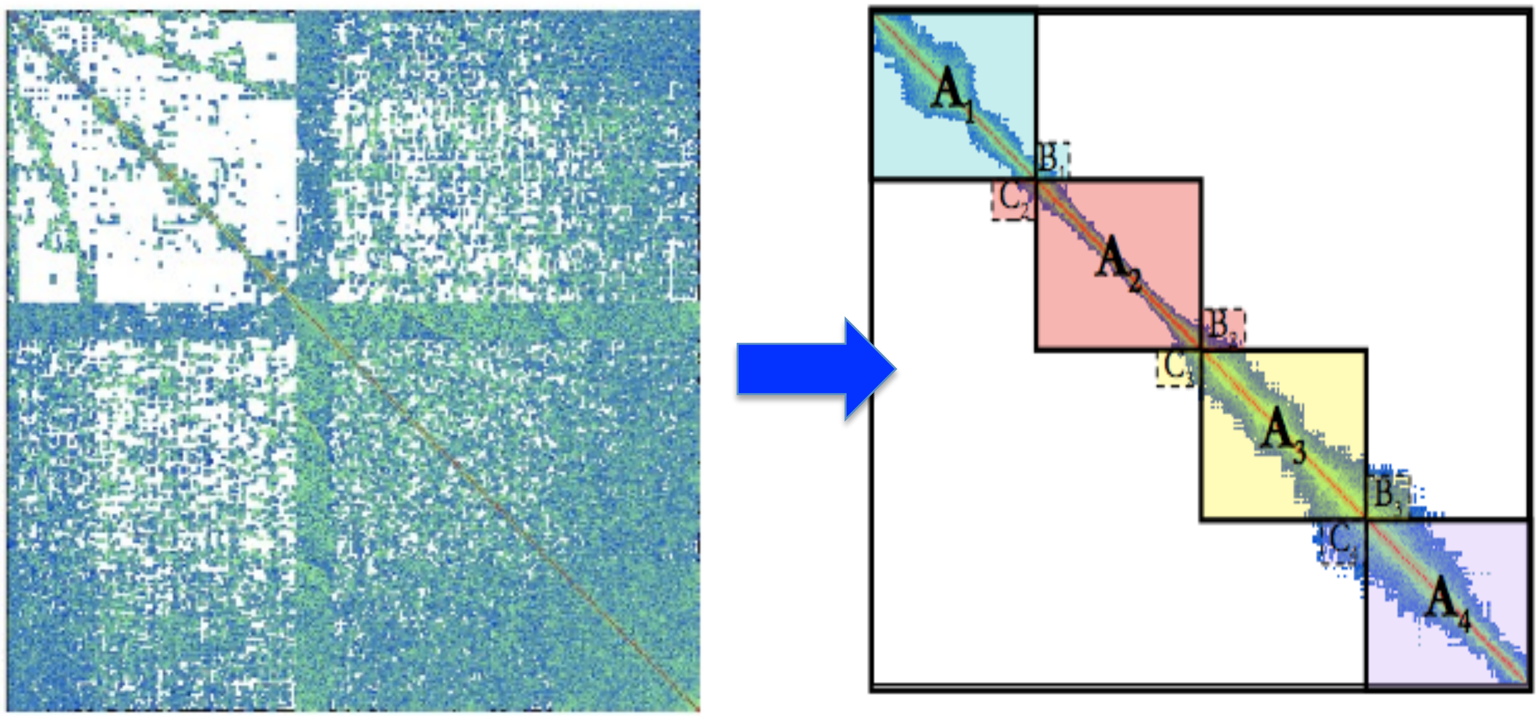
After MC73



After TraceMin-Fiedler

TraceMIN-Fiedler: Murat Manguoglu et. al.

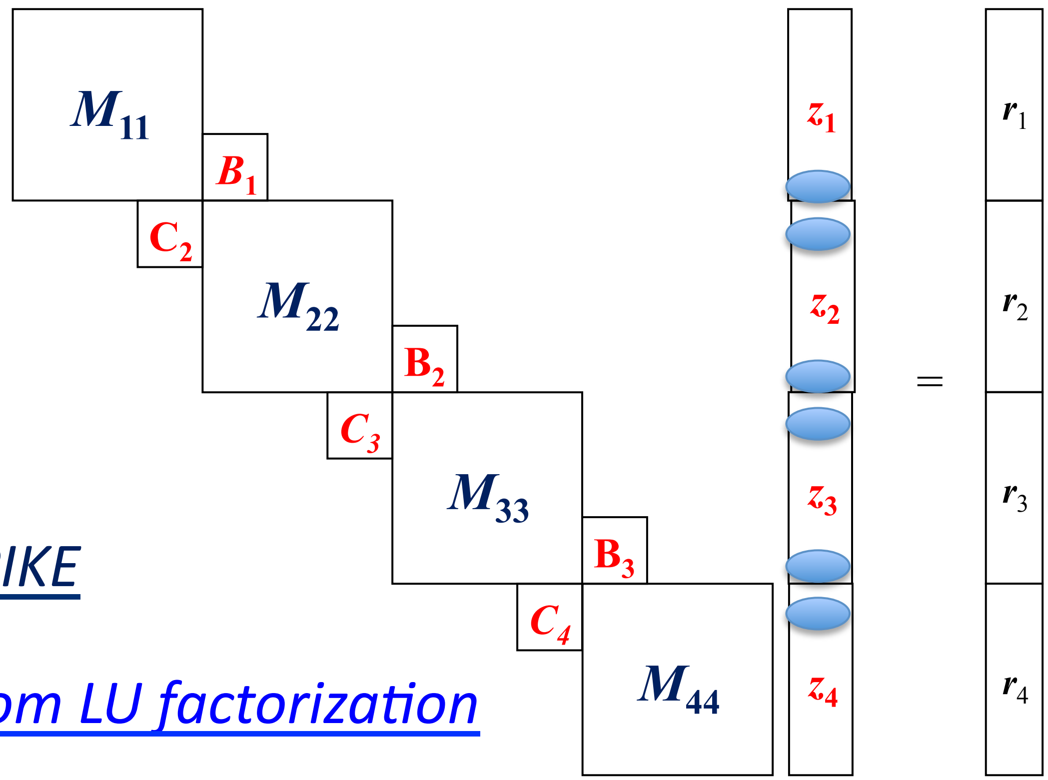
UFL – f2



Before reordering

After reordering via TraceMIN-Fiedler

$M z = r$ (M is "banded")



Each M_{kk} is a general sparse matrix

PSPIKE:

Pardiso-SPIKE

departure from LU factorization

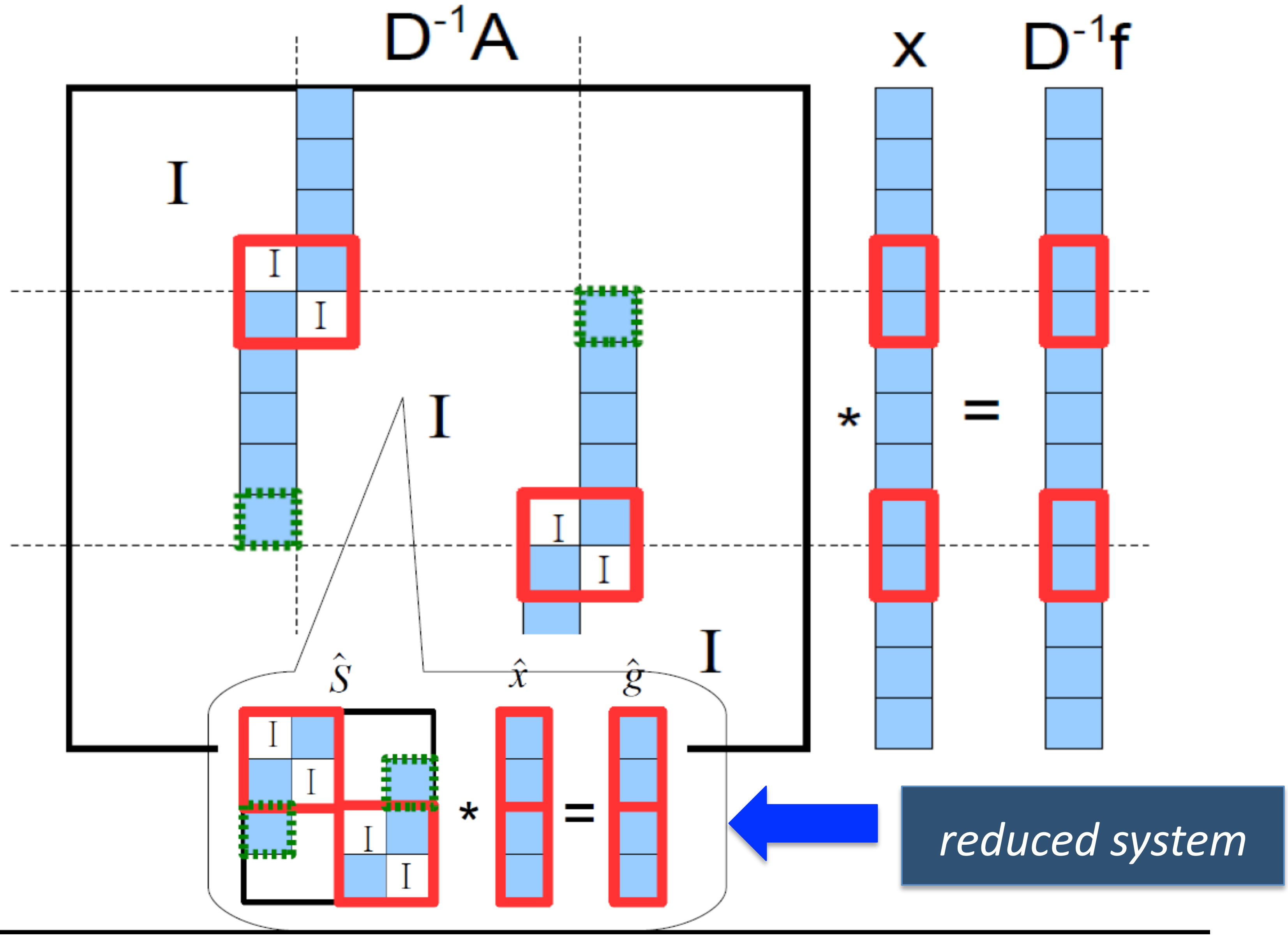
$$P = M + \delta(M) = D' * S'$$

(i) Solve $D' y = r$

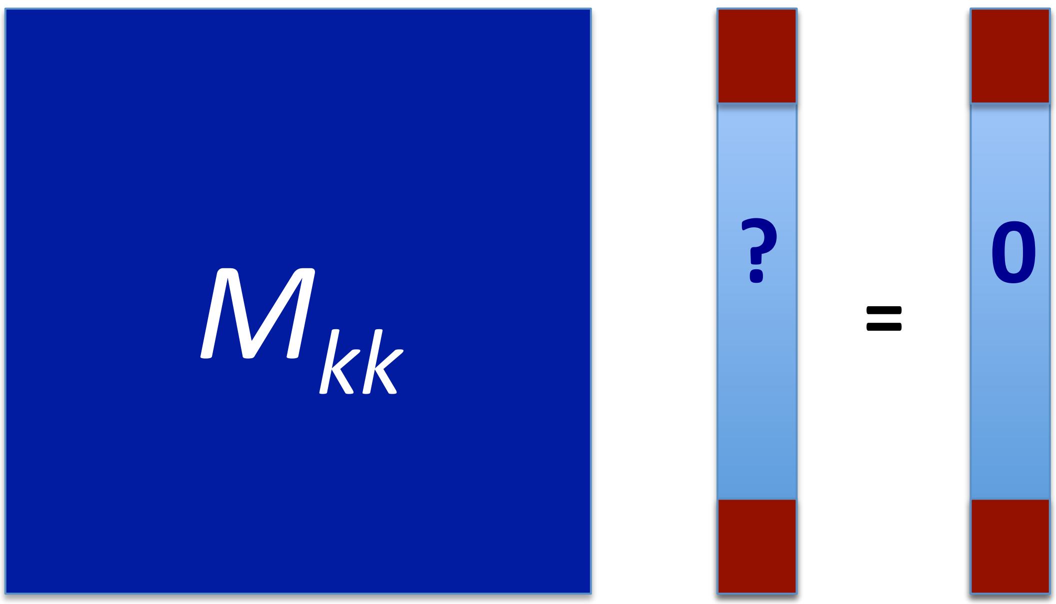
(ii) Solve $S' z = y$

Solving systems involving The preconditioner $P z = r$

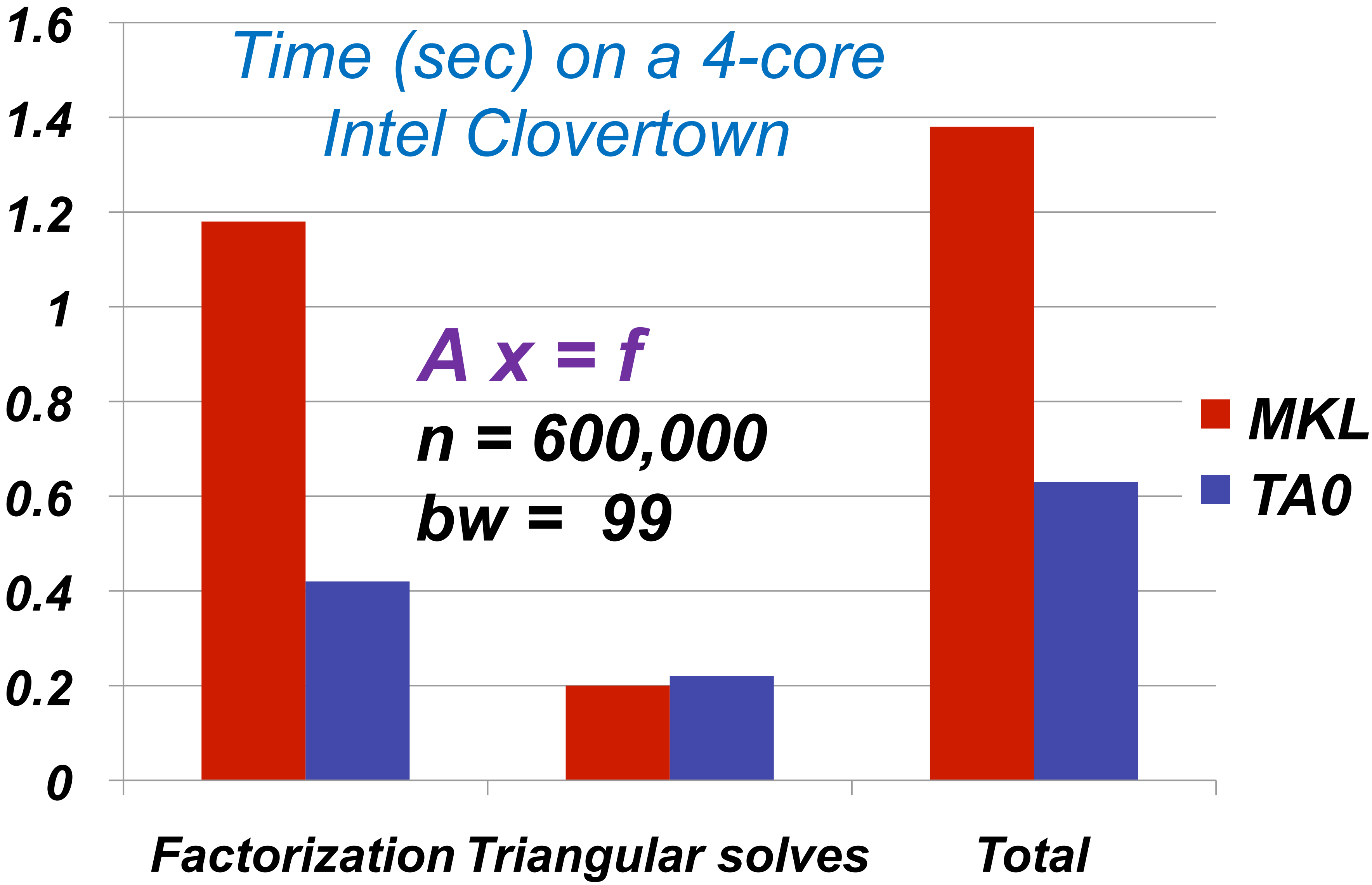
Spike Matrix S for 3 partitions



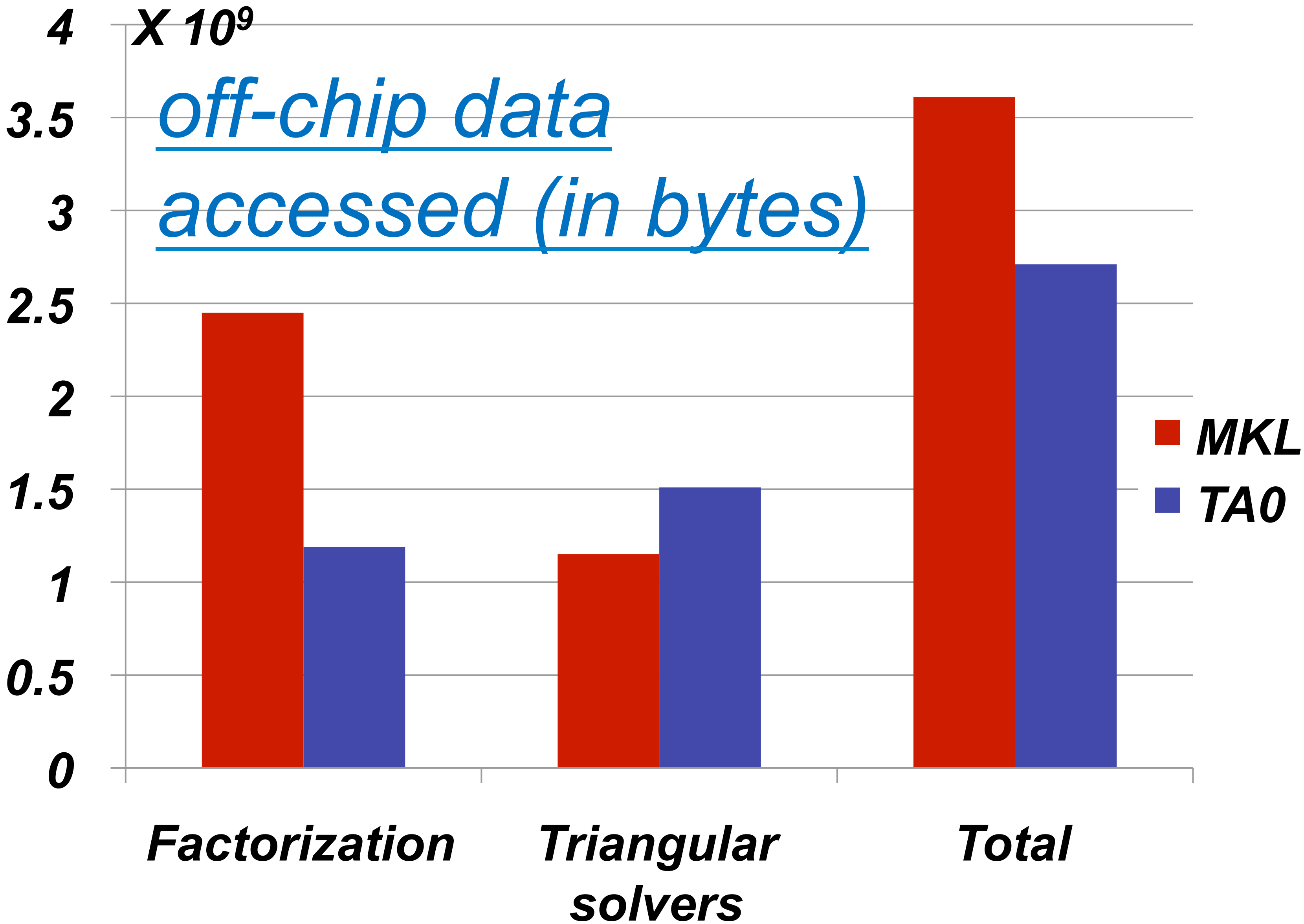
Generating tips of the spikes



Obtain the upper and lower tips of the solution block via the modified direct sparse system solver "Pardiso".



MKL uses ScaLapack (LU factorization)



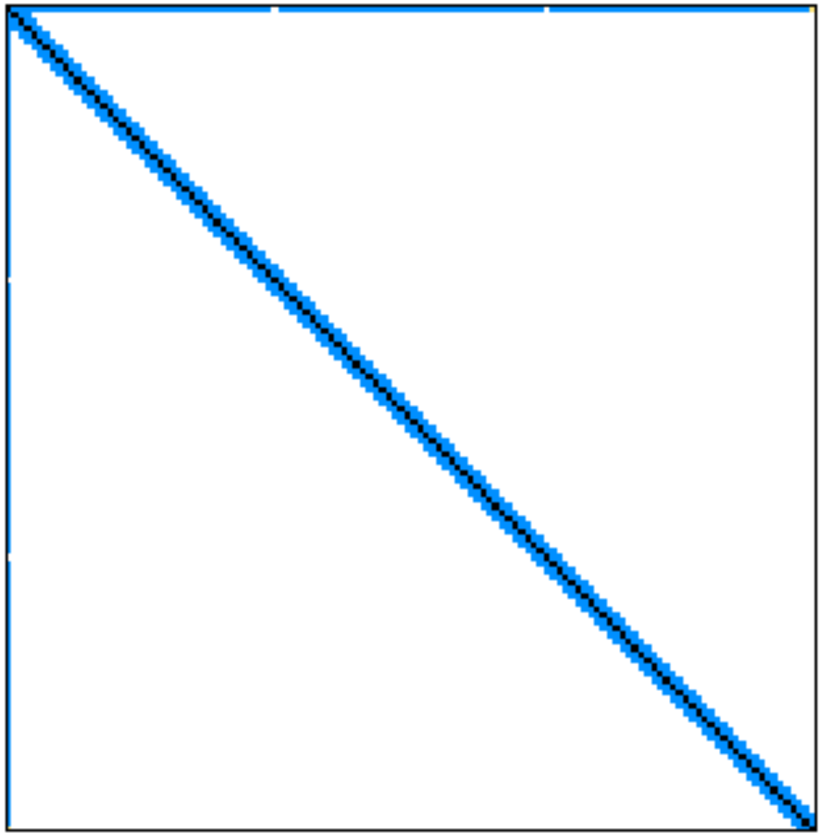
"Analyzing memory access intensity in parallel programs for multicore architectures"

L. Liu, Z. Li, and A. S.

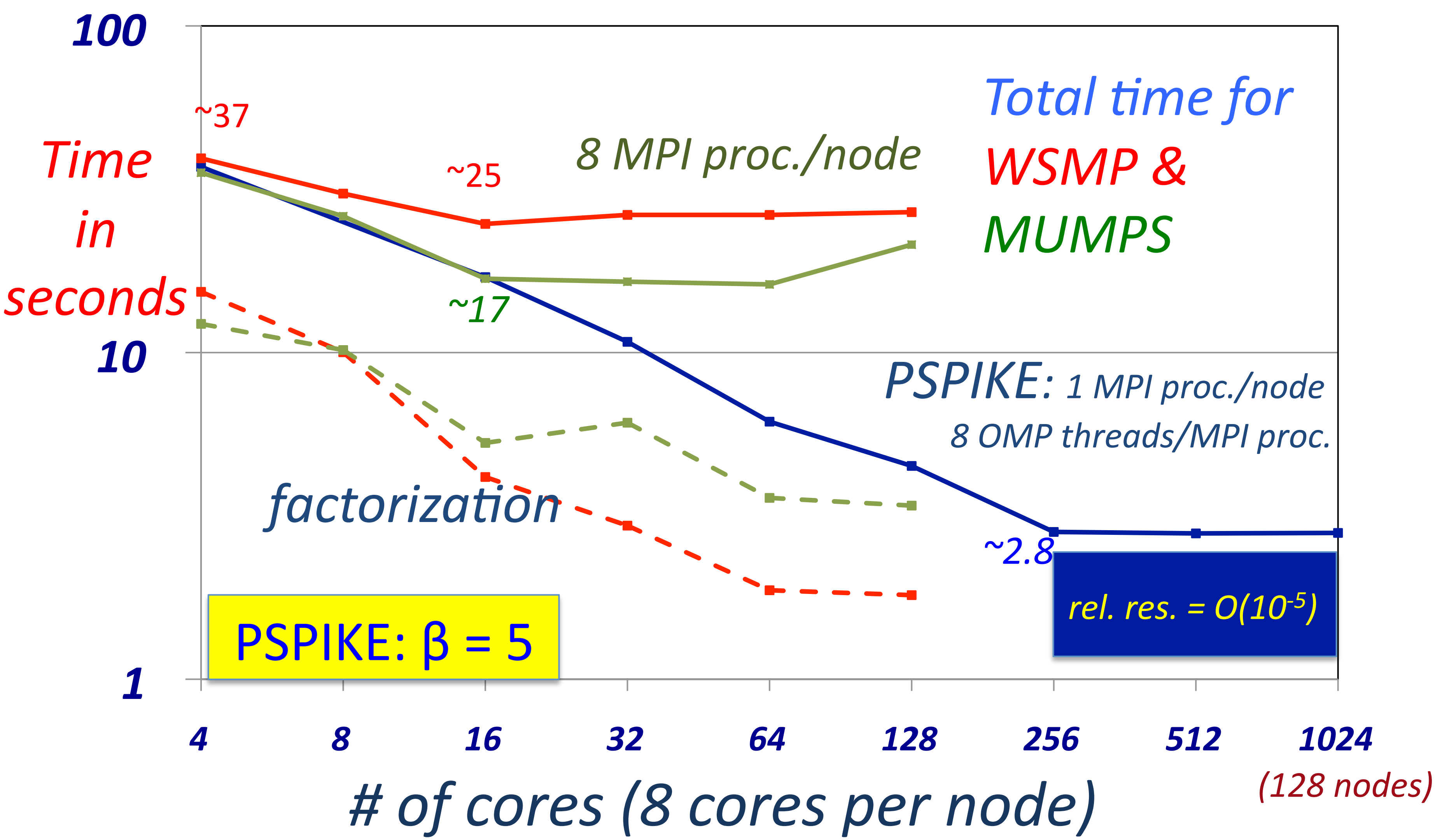
Parallel Scalability of PSPIKE vs. direct solvers

UFL – Rajat31 (circuit simulation)

N ~ 4.7 M
nnz ~ 20 M
nonsymmetric



PSPIKE – WSMP -- MUMPS



Pardiso vs. PSPIKE on a single node

- *PSPIKE is used on an 80 core single node (Intel Xeon E7-8870 server, 2.4 GHz) with a number of different choices of*
  - *Number of MPI processes*
  - *Number of OpenMP threads per MPI process*
  - *Number of cores used*

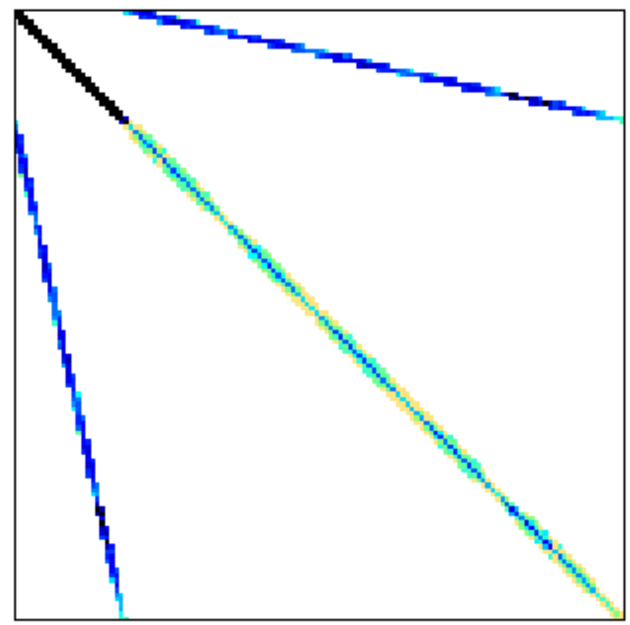
The total number of cores used is the product of the # of MPI processes and the number of threads per process.

System 1: Matrix -- Dziekonski/dielFilterV2real

(High-order finite element method in EM)

http://www.cise.ufl.edu/research/sparse/matrices/Dziekonski/dielFilterV2real.html

Matrix properties	
number of rows	1,157,456
number of columns	1,157,456
nonzeros	48,538,952
structural full rank?	yes
structural rank	1,157,456
# of blocks from dmperm	1
# strongly connected comp.	1
explicit zero entries	0
nonzero pattern symmetry	symmetric
numeric value symmetry	symmetric
type	real
structure	symmetric
Cholesky candidate?	no
positive definite?	no



PSPIKE:

rel. residual ≤ 10^{-8}

PSPIKE vs. Pardiso

(rel. res. ≤ 10^{-8})

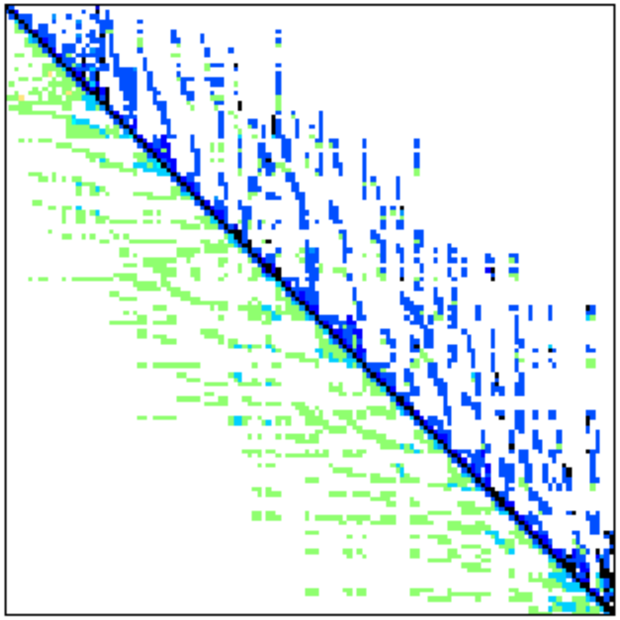
MPI proc.	1	2	4	16	T(Pardiso) ÷ T(PSPIKE)
Cores: 1	400				.95
2	228	163			1.23
4	141	102	43		2.46
64	62	39	16	8	3.88

System 2: Matrix – vanHeukelum/cage13

(DNA electrophoresis, polymer. A. van Heukelum, Utrecht U)

http://www.cise.ufl.edu/research/sparse/matrices/vanHeukelum/cage13.html

Matrix properties	
number of rows	445,315
number of columns	445,315
nonzeros	7,479,343
# strongly connected comp.	1
explicit zero entries	0
nonzero pattern symmetry	symmetric
numeric value symmetry	20%
type	real
structure	unsymmetric
Cholesky candidate?	no
positive definite?	no



PSPIKE:

rel. residual $\leq 10^{-8}$

PSPIKE vs. Pardiso

(rel. res. ≤ 10^{-8})

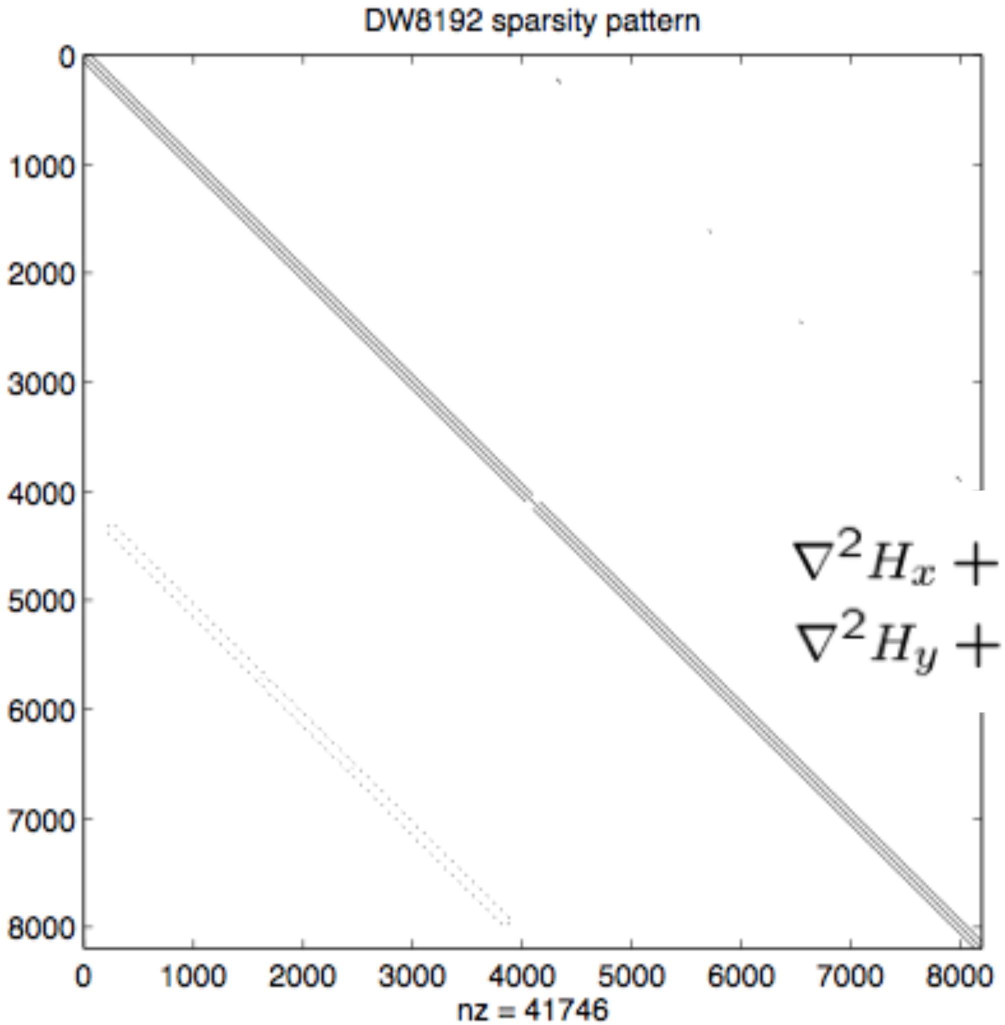
MPI proc.	1	2	4	16	T(Pardiso) ÷ T(PSPIKE)
Cores = 1	21,266				~ 1
2	12,034	5,309			~ 2
4	6,223	3,033	851		~ 6
64	1,055	584	165	12	~ 182

Robustness & Parallel Scalability of PSPIKE

vs.

preconditioned iterative solvers

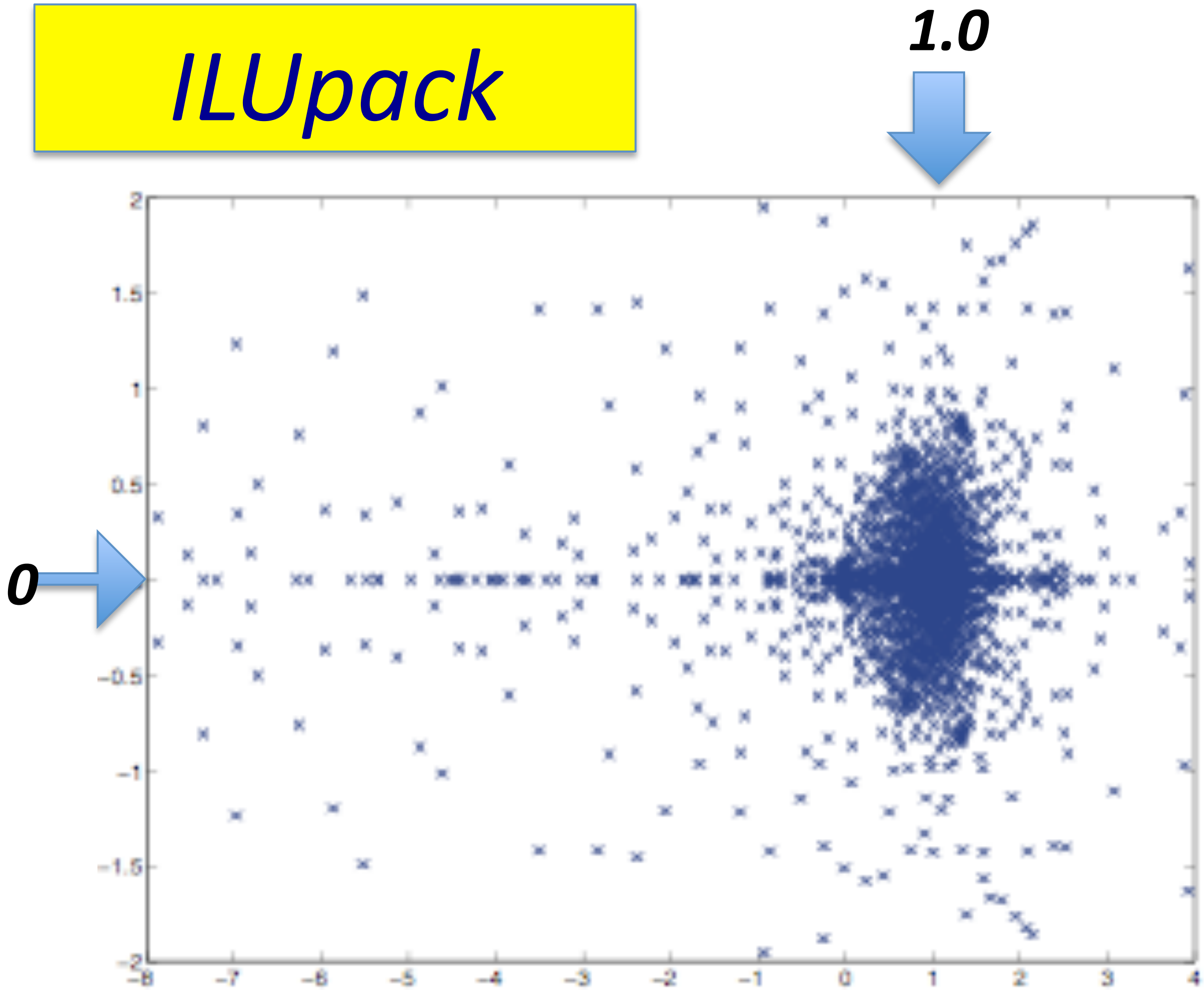
Computational Electromagnetics
UFL: DW8192



Discretization of the Helmholtz equation (2D):

$$\nabla^2 H_x + k^2 n^2(x,y) H_x = \beta^2 H_x,$$
$$\nabla^2 H_y + k^2 n^2(x,y) H_y = \beta^2 H_y.$$

ILUpack



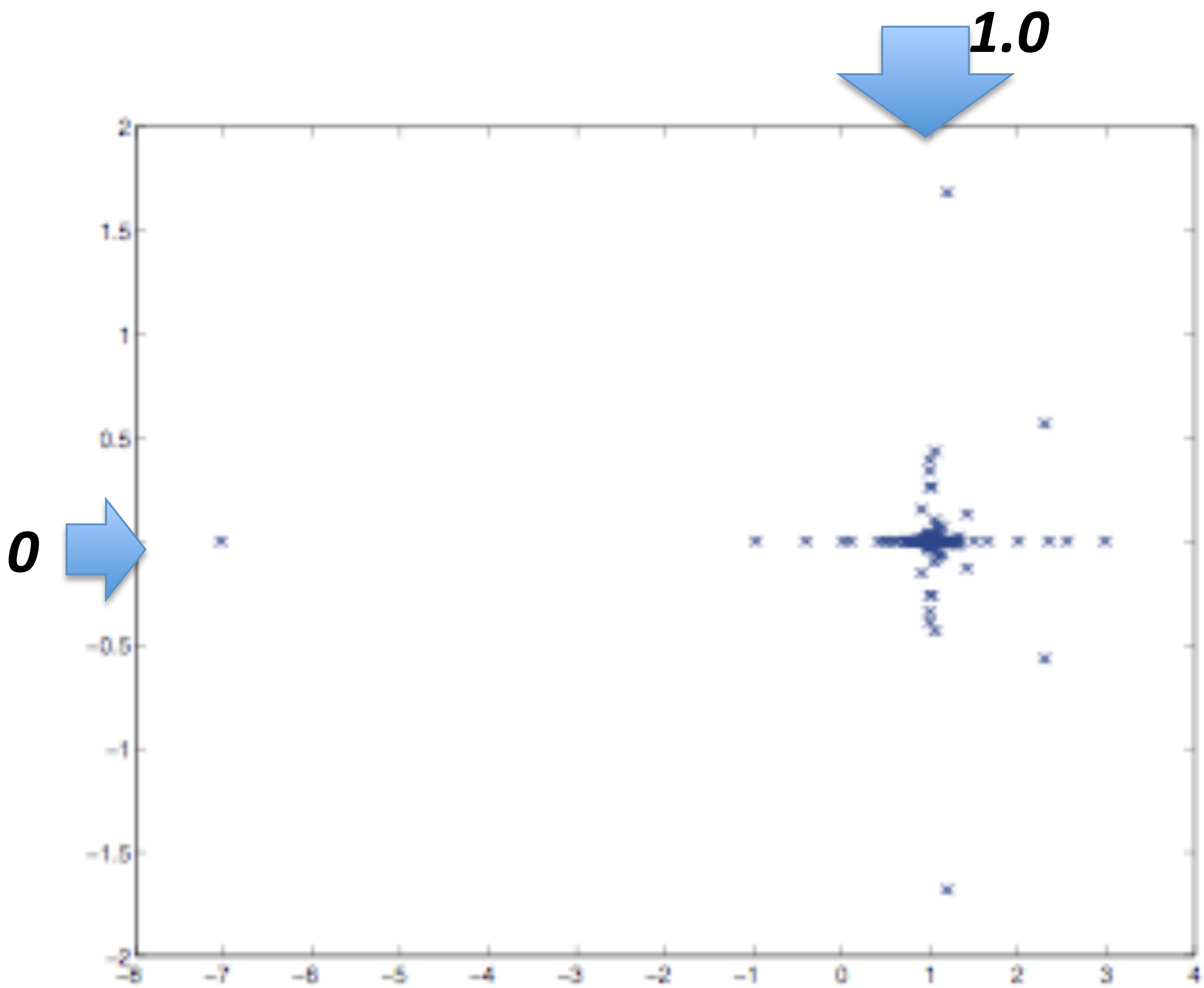
System based on sparse matrix DW8192:

- *n = 8192*
- *nnz = 41,746*
- *κ = O(10^7)*

MC64 + ILUT Preconditioner: P

- *20% fill-in per row*
- *rel. drop tol = 10^{-1}*

Spectrum of $P^{-1}A$



System based on sparse matrix DW8192:

- *n = 8192*
- *nnz = 41,746*
- *κ = O(10^7)*

Spectrum of $M^{-1}A$

WSO + narrow-banded preconditioner: M

- *ε = 10^{-4}*
- half-bandwidth β ≤ *50*

MEMS simulation benchmark 1

System size:

N = 11,333,520

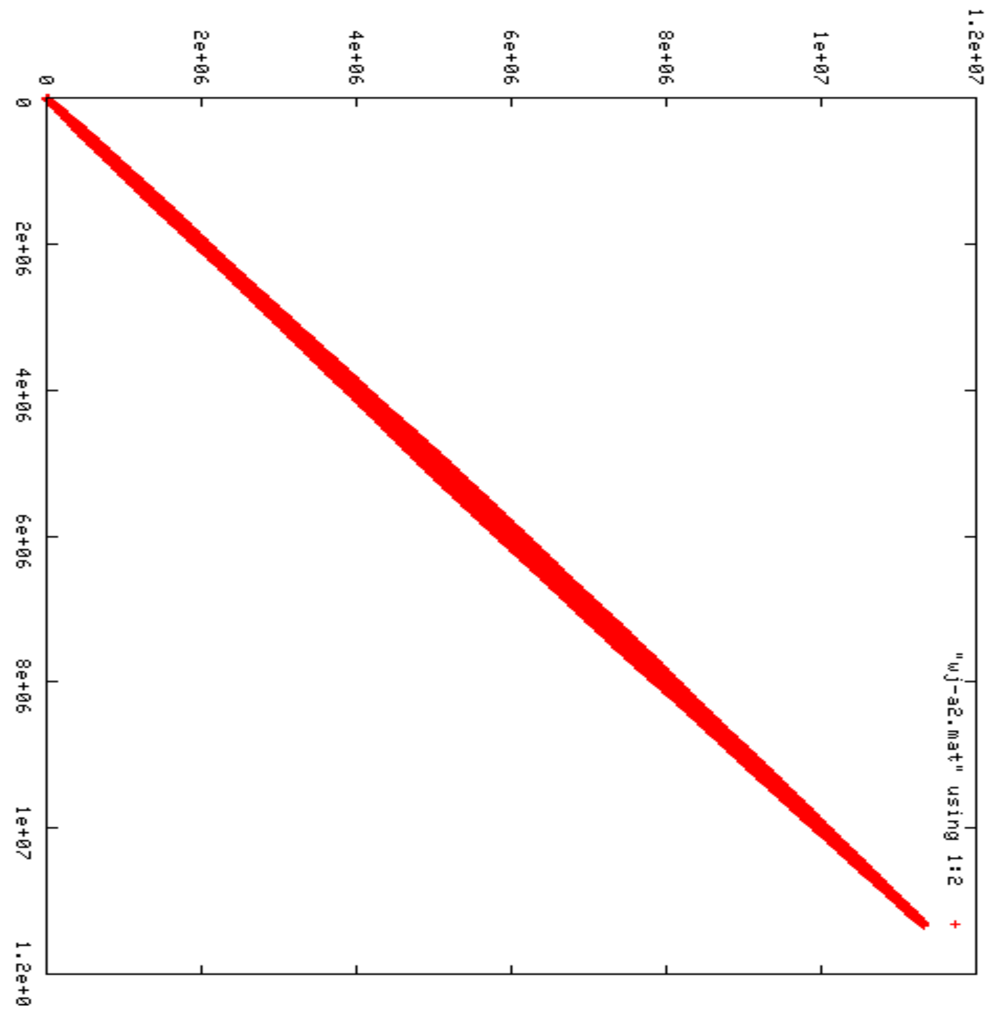
# of nonzeros:

61,026,416

bandwidth:

334,613

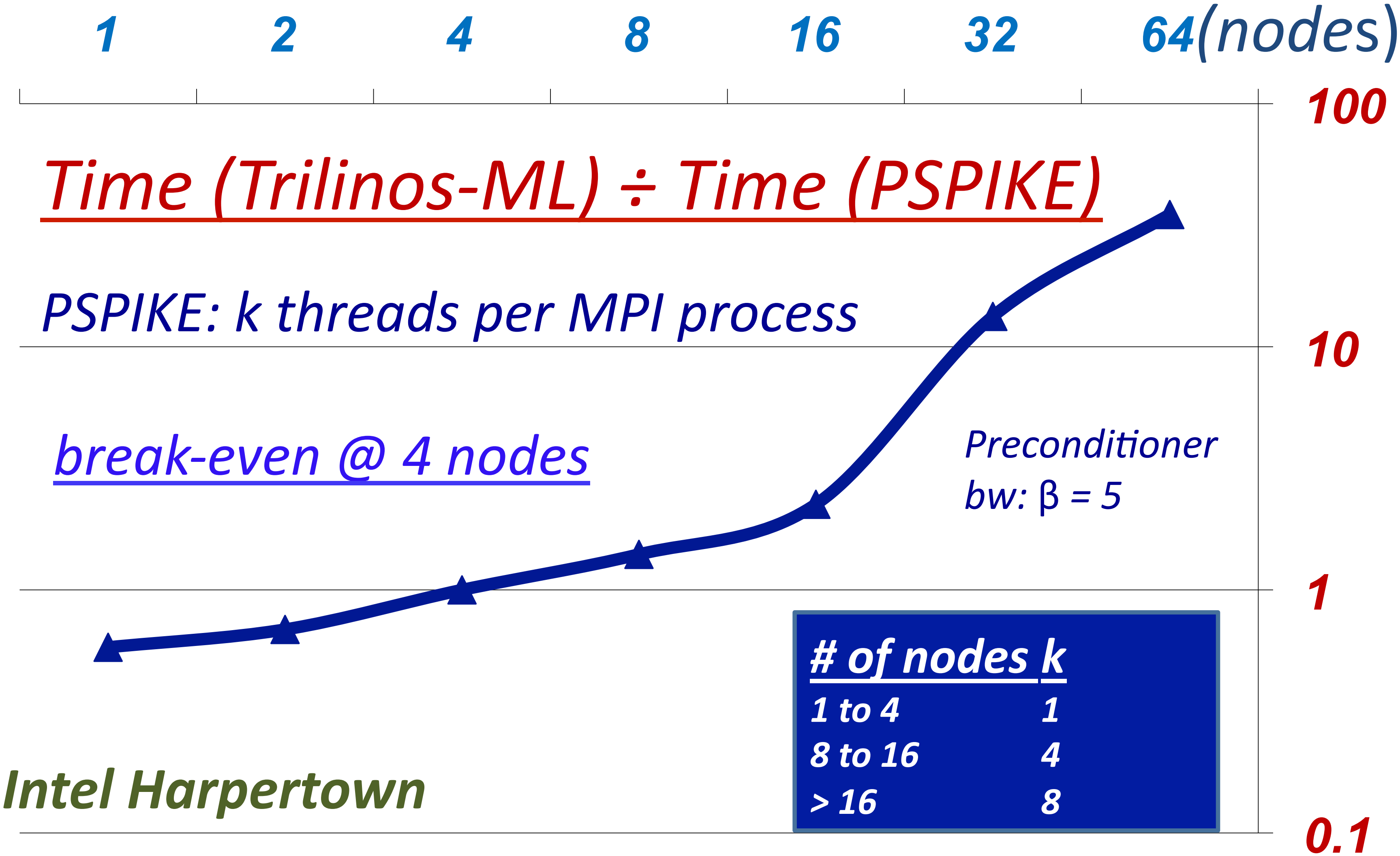
stopping criterion:
rel. res. = $O(10^{-2})$



Scalability of PSPIKE vs. Trilinos
Intel Harpertown

- *Strong scalability of PSPIKE*
  Fixed problem size – 1 to 64 nodes (or 8 to 512 cores)
- *Comparison with AMG-preconditioned Krylov subspace solvers in:*
  - *Hypre (LLNL)*
  - *Trilinos-ML (Sandia)*
    - *Smoother –*
      - *Chebyshev* → *fastest*
      - *Jacobi*
      - *Gauss-Seidel*

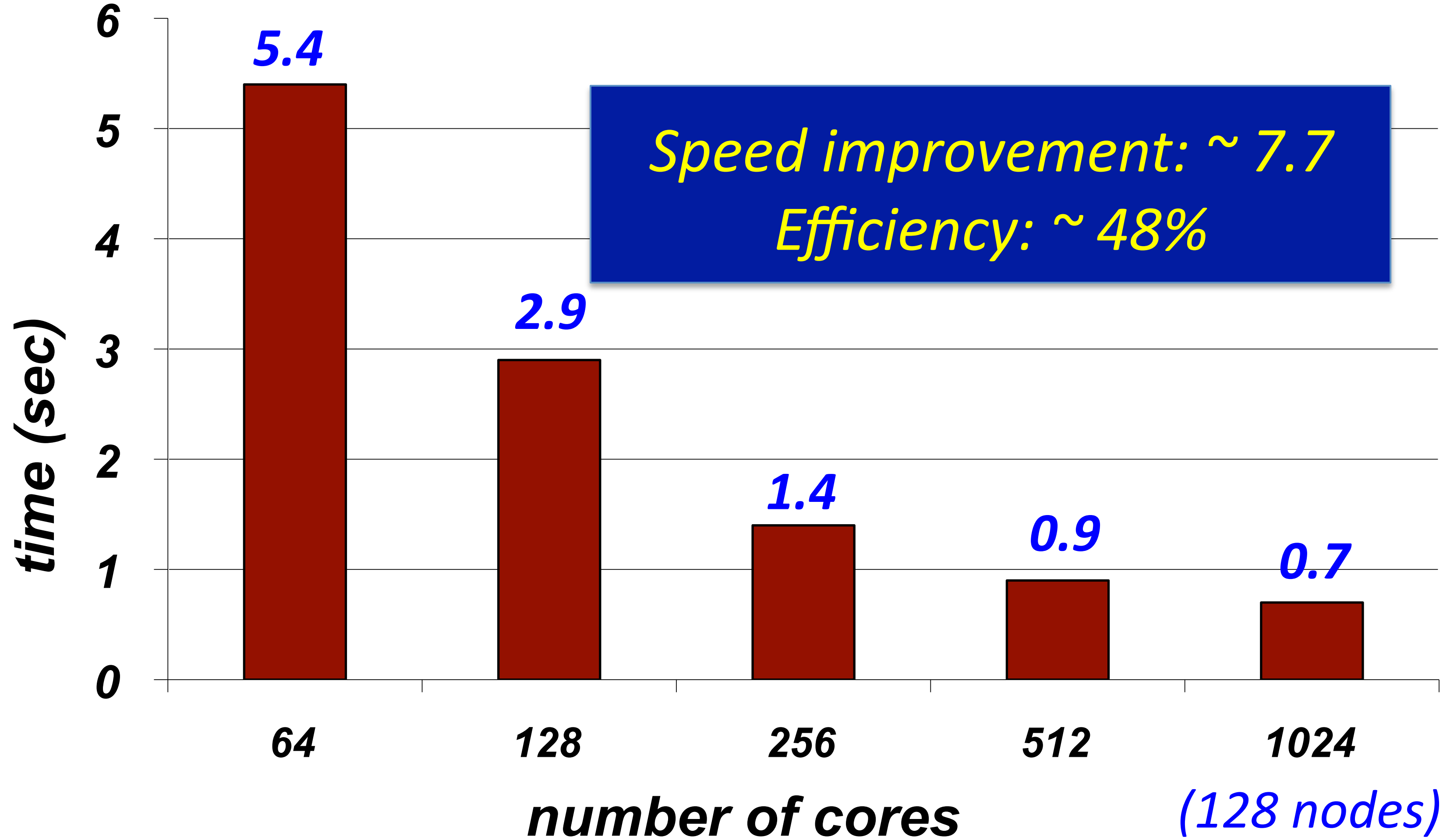
Speed Improvement over Trilinos-ML



MEMS benchmark 1

Strong Scalability on Intel Nehalem

for a MEMS system of order ~ 23M (benchmark 2)



A Parallel Symmetric Eigenvalue Problem Solver: TraceMIN

The Trace minimization scheme:

$Ax = \lambda Bx$; *obtain the p smallest eigenpairs*

$A = A^T$; B: *s.p.d*

$$\min_{Y^T BY = I_p} tr(Y^T AY) = \sum_{i=1}^{p} \lambda_i$$

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p < \lambda_{p+1} \leq \cdots \leq \lambda_n$$

$$Y \in R^{n \times p} \quad ; \quad p << n.$$

A.S. & J. Wisniewski: SINUM, 1982
A.S. & Z. Tong: J. Comp. Appl. Math., 2000.

$$Y_k^T A Y_k = \Sigma_k = diag(\sigma_1^{(k)}, \ldots, \sigma_p^{(k)})$$

$$Y_k^T B Y_k = I_p$$

$$Y_{k+1} = (Y_k - \Delta_k) S_k$$

$$\left\| \begin{array}{ll} \min & tr[(Y_k - \Delta_k)^T A (Y_k - \Delta_k)] \\ s.t. & Y_k^T B \Delta_k = 0 \end{array} \right.$$

Note: if A were s.p.d. we have p indep. problems of the form:

$$\min \quad (y_j^{(k)} - d_j^{(k)})^T A (y_j^{(k)} - d_j^{(k)})$$

$$s.t. \quad Y_k^T B d_j^{(k)} = 0 \qquad j = 1, 2, \ldots, p$$

TraceMin (Outer iterations)

- relative residual $\leq \varepsilon_{out}$
  - form a section

    $$Y^T A Y = \Sigma; Y^T B Y = I_p$$

  - solve

    $$\begin{pmatrix} A & BY \\ Y^T B & O \end{pmatrix} \begin{pmatrix} Y - \Delta \\ -L \end{pmatrix} = \begin{pmatrix} O \\ I_p \end{pmatrix}$$

solve

$$\begin{pmatrix} A & BY_k \\ Y_k^T B & O \end{pmatrix} \begin{pmatrix} \Delta_k \\ L_k \end{pmatrix} = \begin{pmatrix} AY_k \\ O \end{pmatrix}$$

or

$$\begin{pmatrix} A & BY_k \\ Y_k^T B & O \end{pmatrix} \begin{pmatrix} Y_k - \Delta_k \\ -L_k \end{pmatrix} = \begin{pmatrix} O \\ I_p \end{pmatrix}$$

- *different schemes & preconditioners.*
- *TraceMin does not require obtaining solutions with low relative residuals.*

with shifts chosen from

$$\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_p)$$

$$(A - v_j B)x_j = (\lambda - v_j)Bx_j$$

- *convergence rate is ultimately cubic.*
- v_j*'s can be chosen to maintain global convergence.*

TraceMIN vs. Trilinos

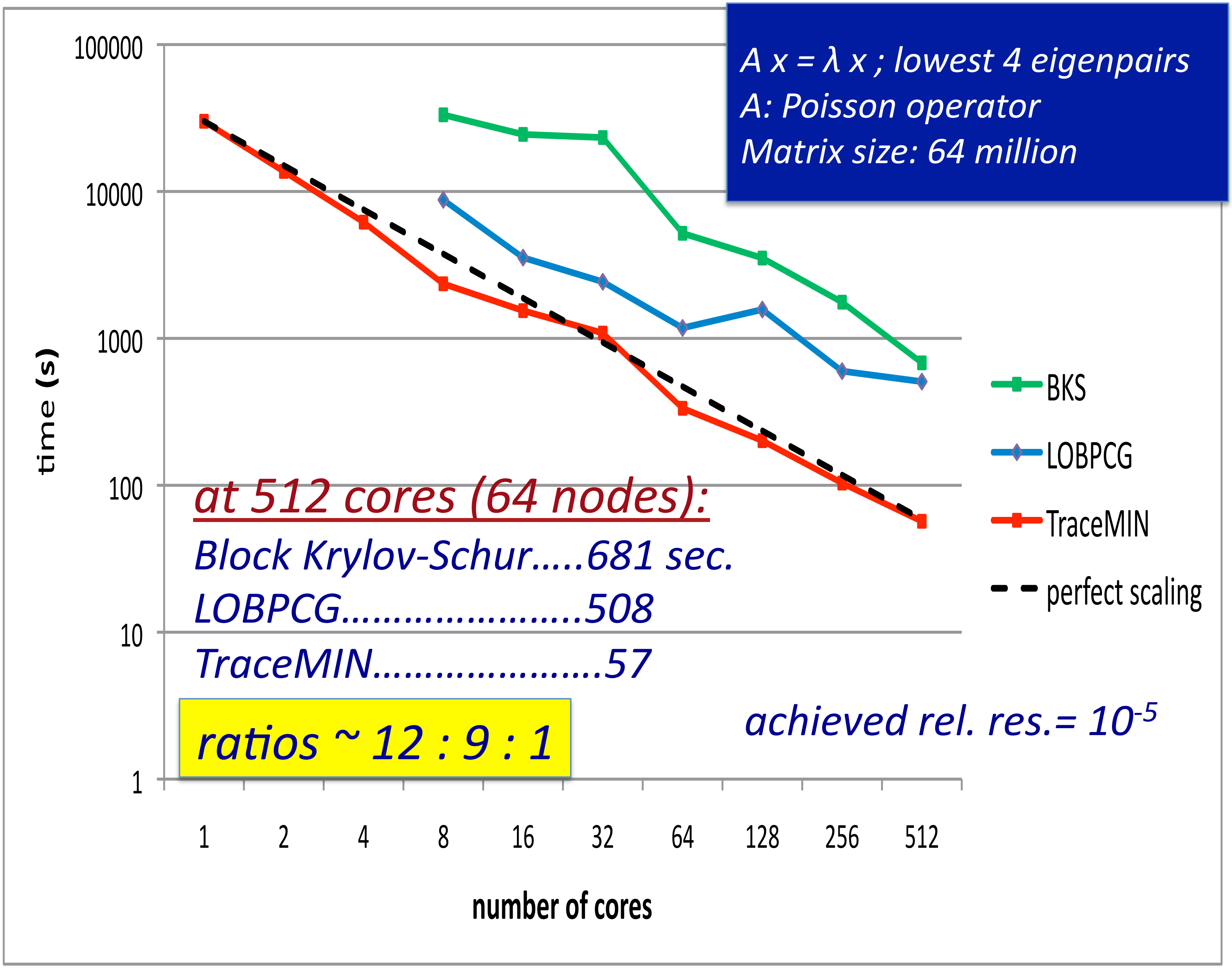
- *We compare our TraceMIN parallel eigensolver against two counterparts in Sandia's parallel Trilinos library:*

`LOBPCG & Block Krylov-Schur`

For two problems:

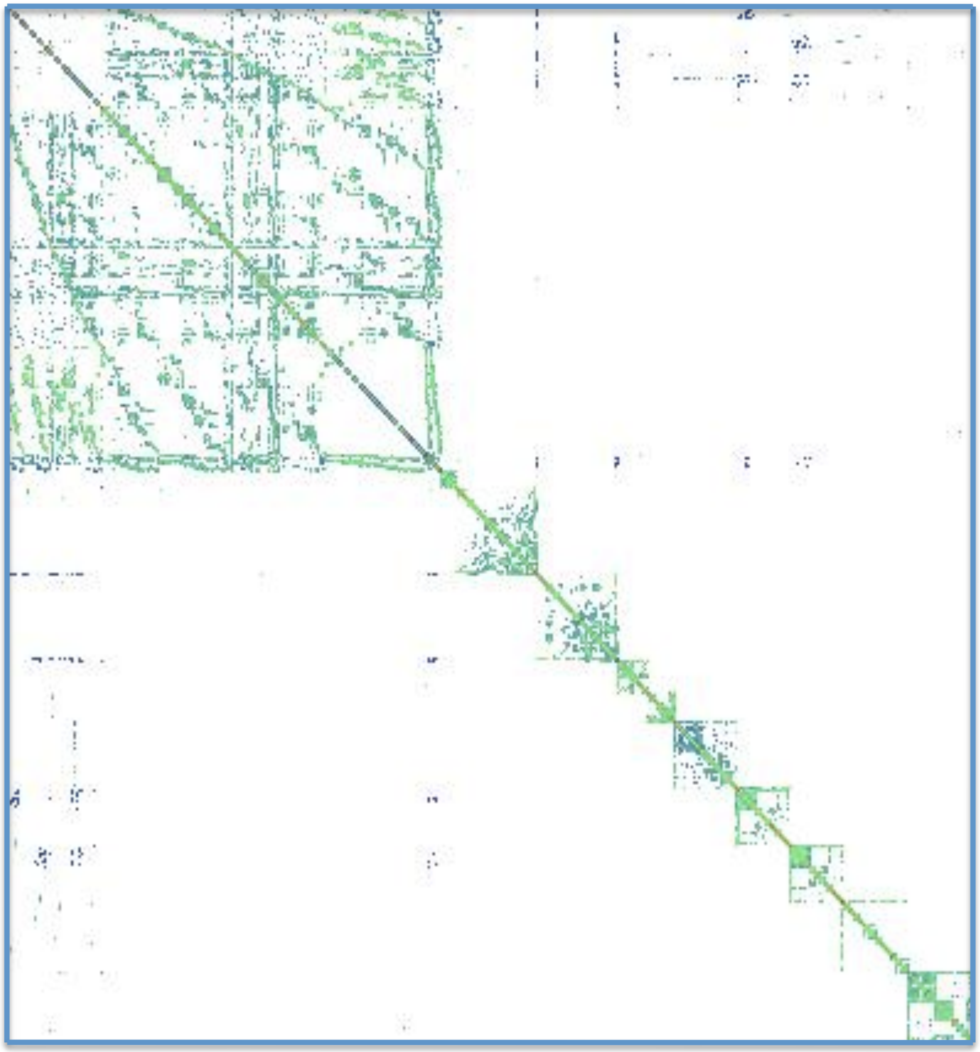
- *Generic 3-D discretization of the Poisson operator on a cube (need lowest 4 eigenpairs),*
- *Predicting car body dynamics at high frequencies (an MSC/NASTRAN benchmark)*

(need lowest 1000 eigenpairs)


A x = λ x ; lowest 4 eigenpairs
A: Poisson operator
Matrix size: 64 million
100000
10000
1000
100
10
1
time (s)
1
2
4
8
16
32
64
128
256
512
number of cores
BKS
LOBPCG
TraceMIN
perfect scaling
at 512 cores (64 nodes):
Block Krylov-Schur…..681 sec.
LOBPCG…………………..508
TraceMIN…………………57
ratios ~ 12 : 9 : 1
achieved rel. res.= 10-5

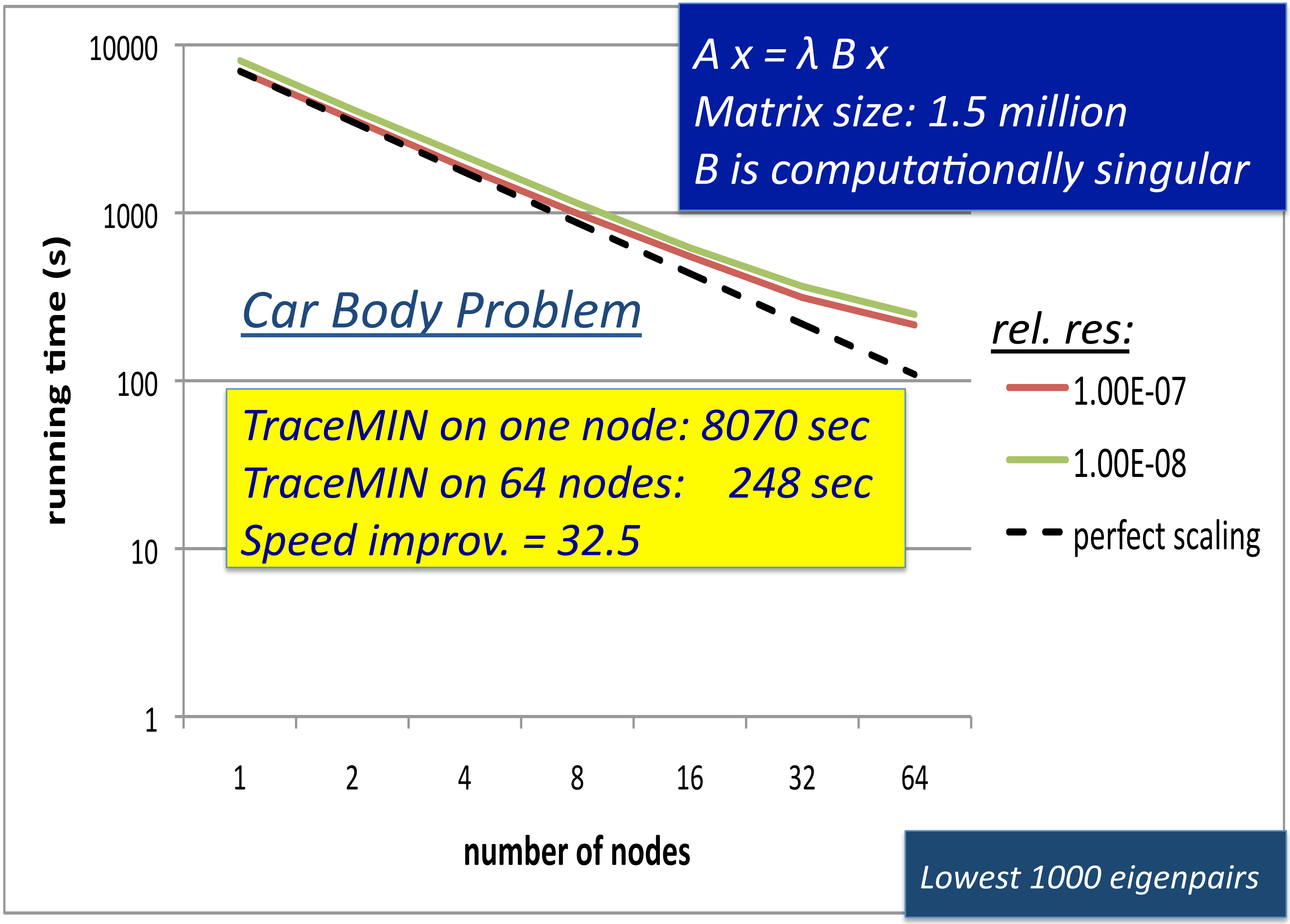
Obtaining selected eigenpairs

- *A generalized symmetric eigenvalue problem resulting from studying car body dynamics at higher frequencies:*
  - $A\,x = \lambda\,B\,x$
  - *A, B are ill-conditioned* ($\kappa \sim O(10^{12})$)
  - *sizes: 1.5 M and 7.2 M*



Sparsity structure of A and B

n ~ 1.5 Million



Both LOBPCG & BKS failed for this problem !

Sampling the spectrum via TraceMIN

4 eigenpairs closest to α_j, j = 1, 2, ...,100

(1.5 M problem)

- *100 nodes – 1 MPI process/node (12 cores)*
- *12 threads/MPI process*
- *One Pardiso factorization per MPI task*
- *Total # of eigenpairs computed: 317*

Time in seconds	*Relative Residual*
20	10^{-5}
21	10^{-6}
22	10^{-9}

7.2 million Car Body Problem

- *$A x = \lambda B x$*
- *Both LOBPCG and BKS in Trilinos failed to solve this generalized eigenvalue problem*
- *TraceMIN time on 2 nodes: 632 seconds*
- *TraceMIN time on 64 nodes: 38 seconds*
- *Speed improvement: ~ 17*
- *Efficiency: ~ 53%*

Thank you!

Generating the weighted graph Laplacian

- *Case 1:*
  - *A is a symmetric matrix of order n*
  - *B = A*
  - *The weighted Laplacian matrix L is given by:*
    - $L(i,i) = \sum |B(i,k)|$; for $k = 1,2,\ldots,n$; $k \neq i$
    - $L(i,j) = -|B(i,j)|$; for $i \neq j$
- *Case 2:*
  - *A is nonsymmetric*
  - $B = (|A|+|A^T|)/2$
  - *L is obtained as in Case 1.*

The Fiedler vector

- *Obtain the eigenvector of the second smallest eigenvalue of* $L\,x = \lambda\,x$ *:*

$$\lambda := \{0 = \lambda_1 < \lambda_2 \leq \ldots\ldots \leq \lambda_n\}$$

- *The sorting process of the Fiedler vector, based on the values of its entries, provides the permutation needed for weighted spectral reordering*

A Parallel Weighted Spectral Reordering Scheme:

*TraceMIN-Fiedler**

** Murat Manguoglu et. al.*

TraceMIN-Fiedler

- $L\,x = \lambda\,x$; L is s.p.s.d.
- Minimize $tr(Y^T L\, Y)$ s.t. $(Y^T Y) = I_p$

$$\Downarrow$$

solution: $\min tr(Y^T L Y) = \Sigma\, \lambda_j$ (j=1,2,..,p)

$$0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_p < \lambda_{p+1} \leq \ldots\ldots \leq \lambda_n$$

Most time consuming kernel in each TraceMIN-Fiedler iteration is solving: $L\,W = Y$ via PCG