PFEAST: A High Performance Sparse Eigenvalue Solver Using Distributed-Memory Linear Solvers

James Kestyn, Vasileios Kalantzis, Eric Polizzi, and Yousef Saad

Tuesday, November 15th, 2016

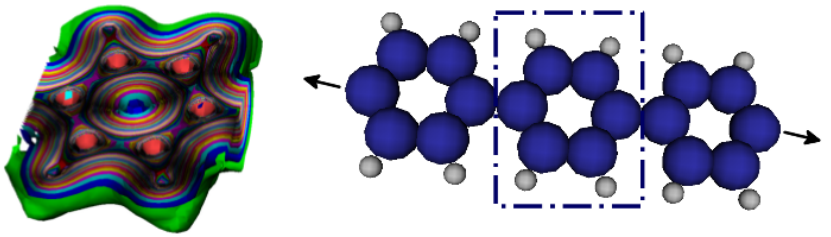
Eigenvalue Problems

$$AX = BX\Lambda$$

Ubiquitous in science and engineering with a diverse range of applications from Physics and Chemistry to Numerical Analysis and Data Analytics

- Principal Component Analysis (Statistics …)
- Stability of Electrical Networks
- Vibrational Analysis (Structural Engineering)
- Dimensionality Reduction

Electronic Structure



- Electron energies/configuration within a material (i.e. bonding)
- Used to predict and verify material properties from first principles

Eigenvalue Algorithms/Solvers

Dense Problems

- Small matrices
- Compute the entire spectrum
- Efficient solvers in LAPACK

Sparse Problems

- Large matrices
- Interior Eigenproblem

  (subset of eigenpairs)
- Many different algorithms exist

Sparse Eigensolvers

Krylov Subspace (Arnoldi, Lanczos)
ARPACK, FILTLAN and TRLAN
Jacobi-Davidson
PRIMME and JADAMILU
LOBPCG
BLOPEX
Other Libraries and Algorithms
SLEPc and ANASAZI
TraceMin
Contour Integration Techniques
zPARES and FEAST

Difficulty: must compute slices of spectrum independently

The FEAST Eigensolver

- Interior eigenvalue problems
- Compute eigenvalue by solving linear systems
- Standard/generalized Hermitian/non-Hermitian problems
- Matrix format independent
  - Banded, sparse and dense predefined interfaces
  - Can import a custom linear solver through RCI

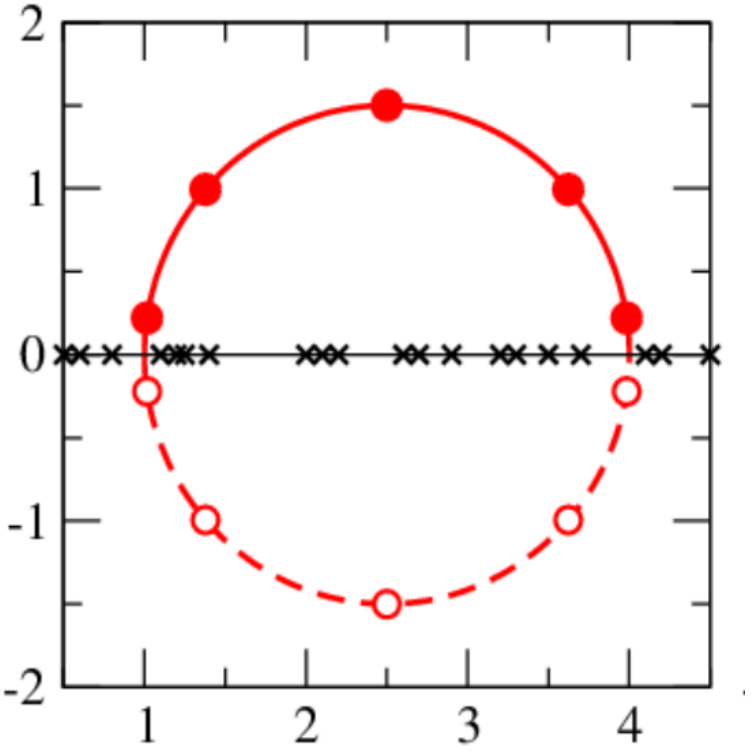
www.feast-solver.org

v2.1 included in Intel MKL

J. Kestyn, E. Polizzi, P.T.P Tang, SIAM Journal on Scientific Computing (SISC), 38, S772-S799 (2016)
P. T. P. Tang, E. Polizzi, SIAM Journal on Matrix Analysis and Applications (SIMAX), 35, 354-390 (2014)
E. Polizzi, Phys. Rev. B. Vol. 79, 115112 (2009)

FEAST = Subspace Iteration + Spectral Projection

1. **Define accelerator**



$$Q_{m_0} = \sum_{j=1}^{n_e} \omega_j (z_j B - A)^{-1} B \tilde{X}_{m_0}$$

$$Q_{m_0} = \sum \omega_j Q_{m_0}^{(j)}$$

$$(z_j B - A) Q_{m_0}^{(j)} = B \tilde{X}_{m_0}$$

2. **Project matrices**

$$A_q = Q^H A Q \quad B_q = Q^H B Q$$

3. **Solve reduced Problem**

$$A_q W_q = B_q W_q \Lambda_q$$

4. **Recover approximate eigenpairs**

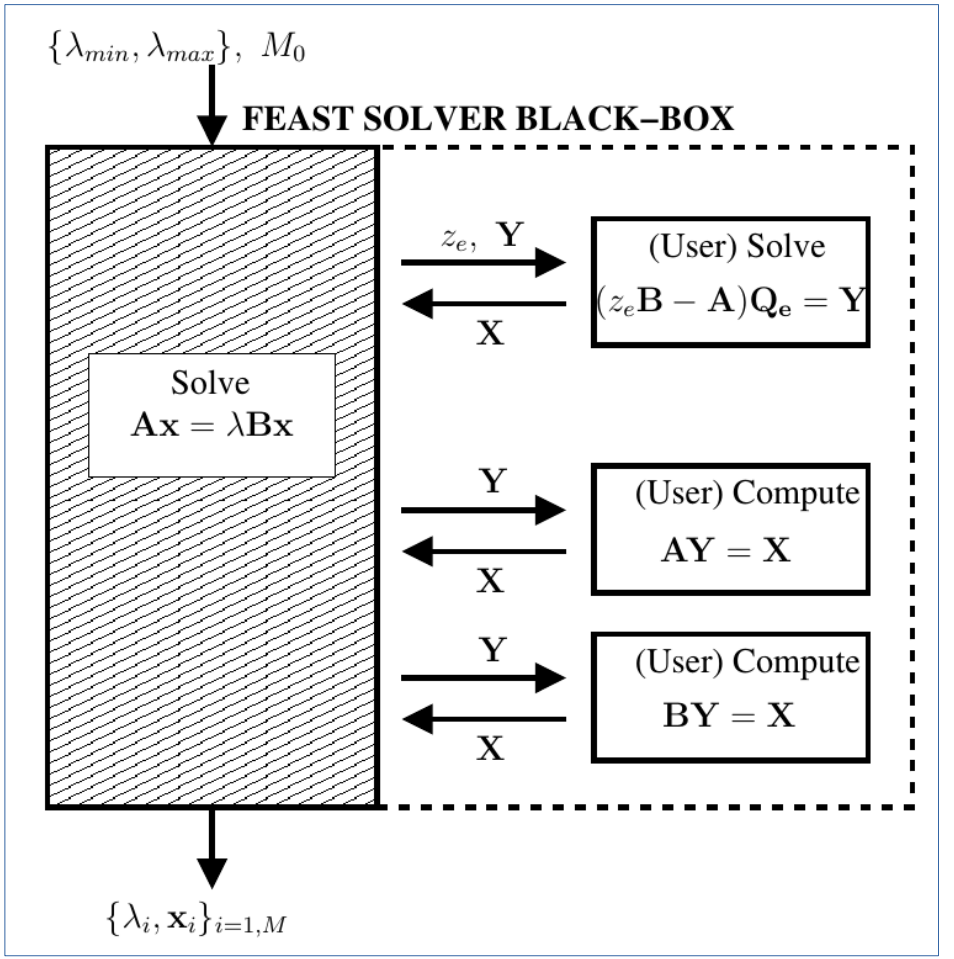
$$\tilde{X} = Q W_q \qquad \tilde{\Lambda} = \Lambda_q$$

5. **Iterate until convergence**

FEAST Pseudo Code and RCI

Algorithm 1 The FEAST Algorithm

input: A, B, $\tilde{X}_{m_0}$, $\{(z_j, \omega_j)\}_{1,\ldots,n_e}$, ϵ
while ($||A\tilde{X}_m - B\tilde{X}_m\Lambda_m|| > \epsilon$) **do**
  $Q_{m_0} = 0$
  for ($j = 0; j < n_e; j = j + 1$) **do**
    $Q_{m_0}^{(j)} \leftarrow (z_j B - A)^{-1} B\tilde{X}_{m_0}$
    $Q_{m_0} \leftarrow Q_{m_0} + \omega_j Q_{m_0}^{(j)}$
  end for
  $A_q = Q^H A Q \qquad B_q = Q^H B Q$
  Solve $A_q W_q = B_q W_q \Lambda_q$
  $\tilde{X}_{m_0} = Q_{m_0} W_q \qquad \tilde{\Lambda} = \Lambda_q$
end while
output: $\tilde{X}_m$, $\tilde{\Lambda}_m$



Three Levels of MPI Parallelism

Previous releases of FEAST (v3.0 and prior) offer only 2 levels of MPI parallelism

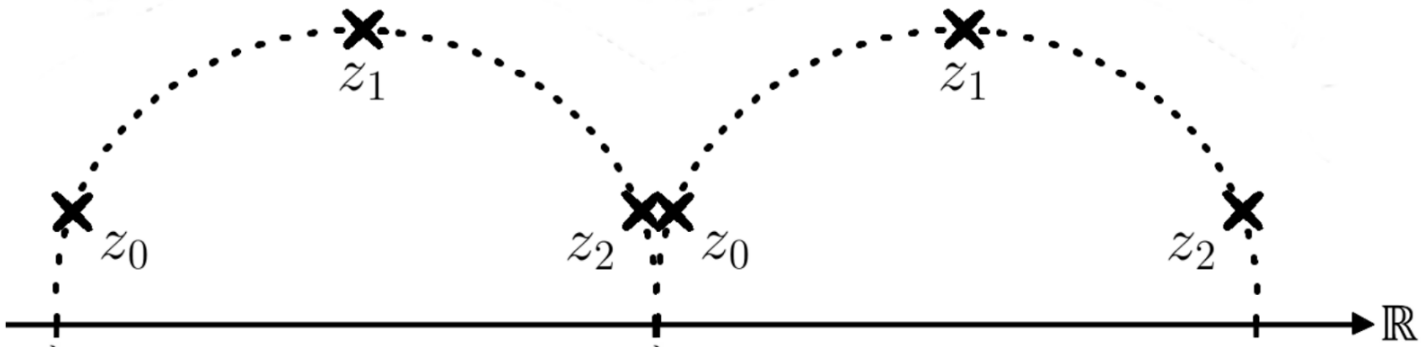
L1: Solving multiple search intervals in parallel (no overlap)

L2: Solving the multiple (independent) linear systems in parallel

L3: Solving each linear system in parallel

First Level of Parallelism

L1: Solving multiple search intervals in parallel (no overlap)



Allows to limit the search subspace size

m_0 should be less than ~1000 because of $O(m_0^3)$ scaling of the reduced eigenvalue problem

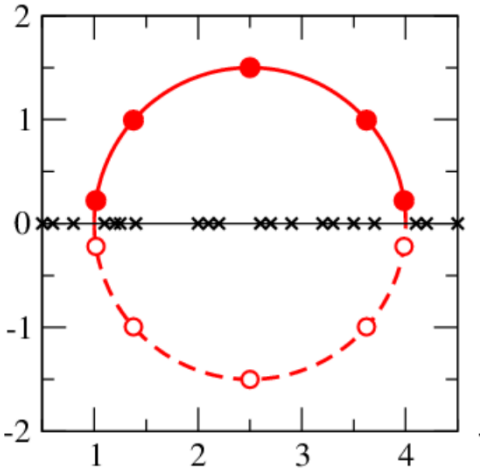
Global orthogonality (between contours) is largely preserved

Second Level of Parallelism

L2: Solving the multiple (independent) linear systems in parallel

$$Q_{m_0} = \sum_{j=1}^{n_e} \omega_j (z_j B - A)^{-1} B \tilde{X}_{m_0}$$

$$(z_j B - A) Q_{m_0}^{(j)} = B \tilde{X}_{m_0}$$



Advantages:

- Perfect parallelism up to the number of quadrature nodes
- Ideal case #**L2** = n_e → Factorization computed once

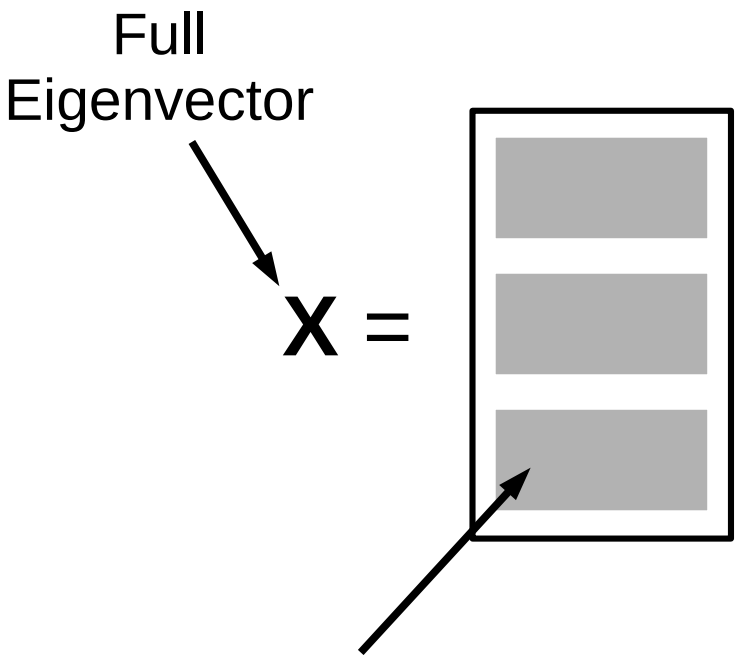
Limitations:

- Memory footprint increased because of extra copies of the matrix and eigenvalues/eigenvectors

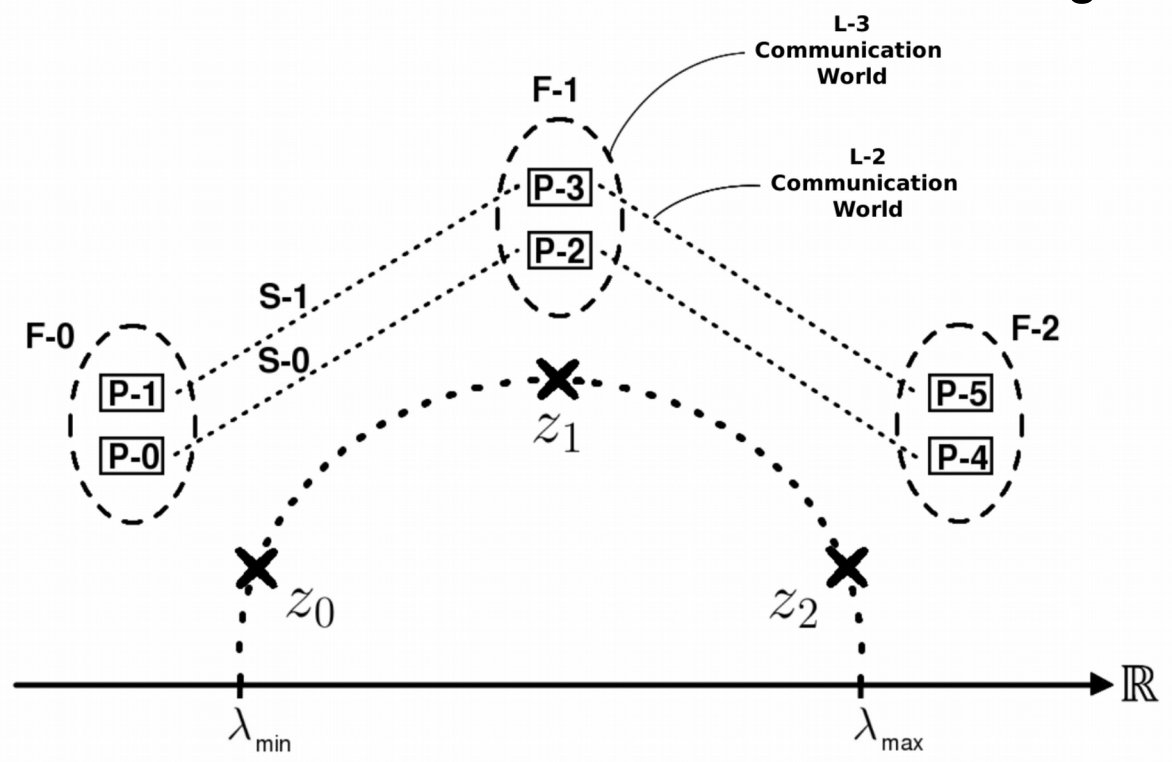
Third Level of Parallelism

L3: Solving each linear system in parallel

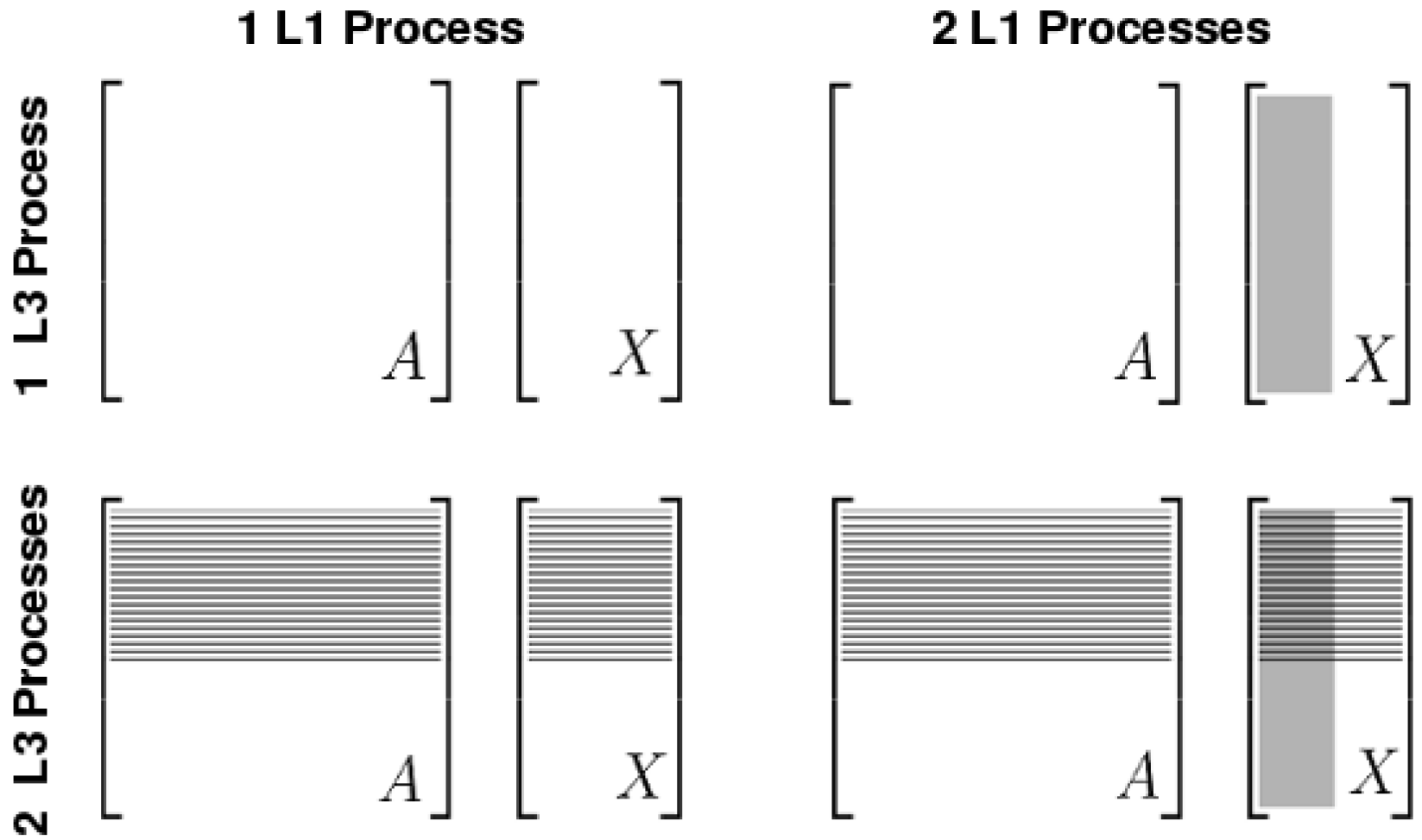
$$(z_j B - A)Q_{m_0}^{(j)} = B\tilde{X}_{m_0}$$



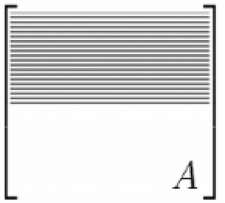
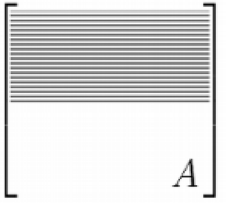
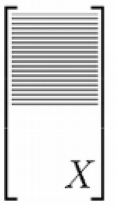
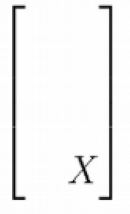
L2 and **L3** Communicators must be managed



Reducing the Memory Footprint with L1 and L3



PFEAST with Sparse Direct Solvers

	MKL-Cluster-PARDISO	MUMPS
Matrix	• Distributed 1D by Row 	• Distributed 1D by Row 
RHS/ Solution	• Distributed 1D by Row 	• Fully Constructed on Head Node 

* P.R. Amestoy, I.S. Duff and J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers (1998)
*A. Kalinkin and K. Arturov “Asynchronous approach to memory management in sparse multifrontal methods on multiprocessors”

Domain Decomposition Solvers

Reordered Matrix

Independent blocks

$$\begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_p \end{bmatrix}$$

$$A = \begin{bmatrix} C & E \\ E^T & D \end{bmatrix}$$

Connections

Linear System to Solve:

$$\begin{bmatrix} C & E \\ 0 & S \end{bmatrix} \begin{bmatrix} X_l \\ X_e \end{bmatrix} = \begin{bmatrix} Y_l \\ Y_e - E^T C^{-1} Y_l \end{bmatrix}$$

Schur Complement

$$(D - E^T C^{-1} E)$$

Three step process for Schur

(i) Solve: $CT = Y_l$

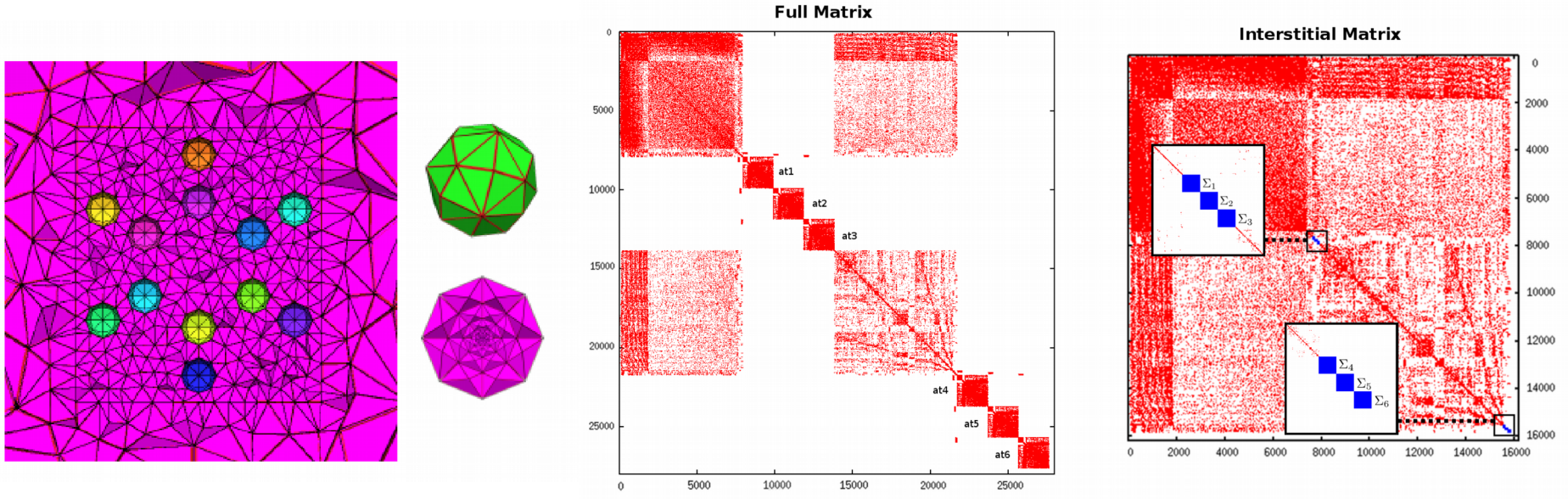
(ii) Solve: $SX_e = Y_e - E^T T$

(iii) Solve: $CX_l = Y_l - E^T X_e$

***Vasileios Kalantzis "Domain Decomposition Techniques for Contour Integration Eigenvalue Solvers"**

Electronic Structure Application

Muffin-tin Domain Decomposition



Use a Schur Complement Approach

- MKL-Cluster-Pardiso (distributed-memory solver) is used to solve the “Interstitial” region
- MKL-Pardiso (shared-memory solver) is used to solve the “Atom” regions

*A. Levin, D. Zang, and E. Polizzi. "FEAST fundamental framework for electronic structure calculations: Reformulation and solution of the muffin-tin problem."

Minnesota Supercomputing Institute

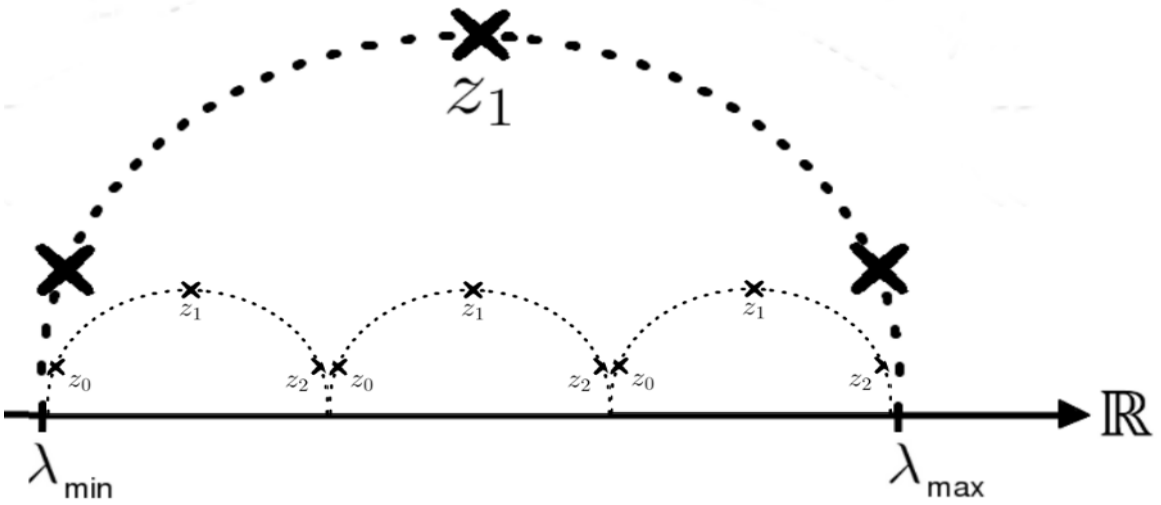
Mesabi Cluster:

- 2 Sockets per Physical Node
  - 2.5GHz Haswell E5-2680v3 12 Core Processor
- 64 GB DRAM
- InfiniBand Interconnect
- Each MPI process defined as a single socket
  - 2 MPI processes per Physical Node

    (12 Cores, 12 Threads, 32GB)

www.msi.umn.edu

Results: L1 Strong Scalability



#**L1** = 1 and 3
#**L2 =** 1
#**L3 = 13**
#**MPI:** from **13** to **39**
#**Cores:** from **156** to **468**

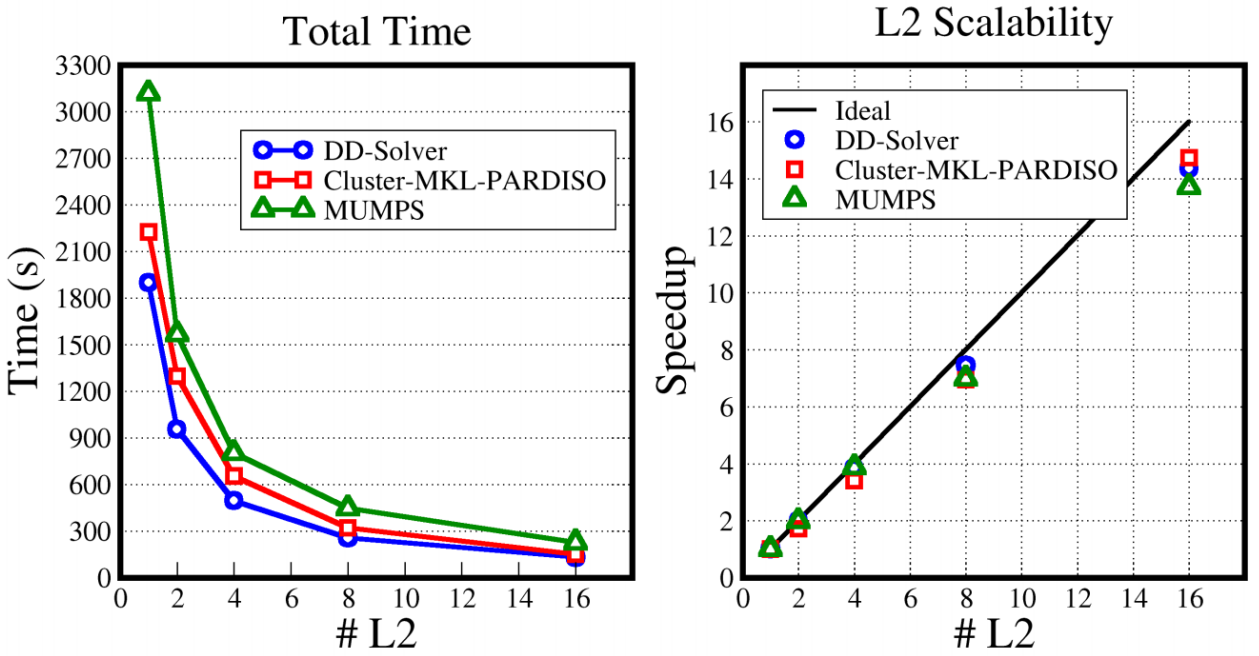
	λ_{min}	λ_{max}	m_0	# Eig	Fact (s)	Solve (s)
C0	-500	-2	600	273	96	**170**
C1	-500	-100	200	66	94	**59**
C2	-100	-16	200	92	99	**57**
C3	-16	-2	200	115	96	**56**

N=302,295; 16 Quadrature Nodes; 1 FEAST Iteration

Results: L2 Strong Scalability

Each independent linear system $(z_j B - A)Q_{m_0}^{(j)} = B\tilde{X}_{m_0}$ can be solved in parallel

#**L2**	from 1 to 16
#**L3**	= 3
#**MPI**	from 3 to 48
#**Cores**	from 36 to 576



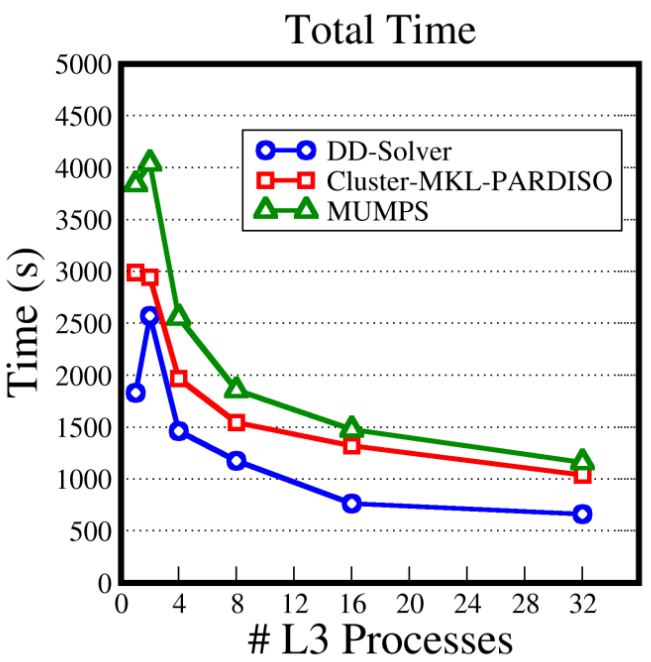
N=302,295; M0=200; 16 Quadrature Point

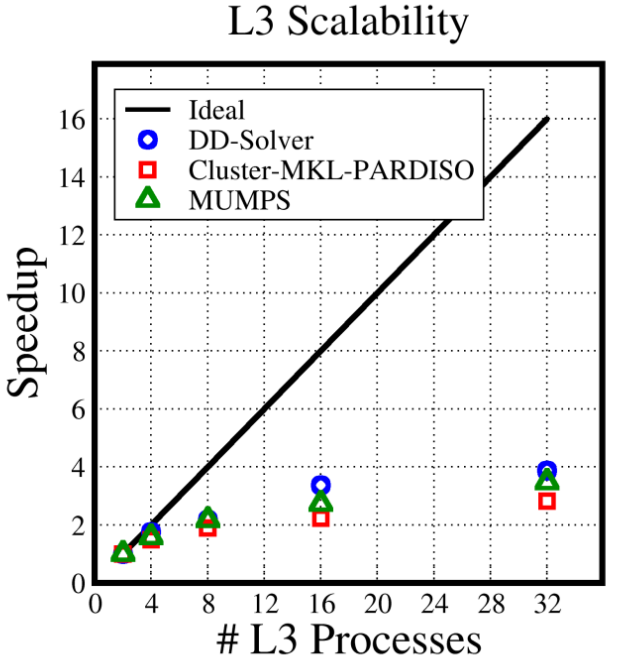
Results: L3 Strong Scalability

Distributed Memory solver for $(z_j B - A)Q_{m_0}^{(j)} = B\tilde{X}_{m_0}$

#**L2** **=** 1
#**L3:** from **1** to **32**

#**MPI:** from **1** to **32**
#**Cores:** from **12** to **384**

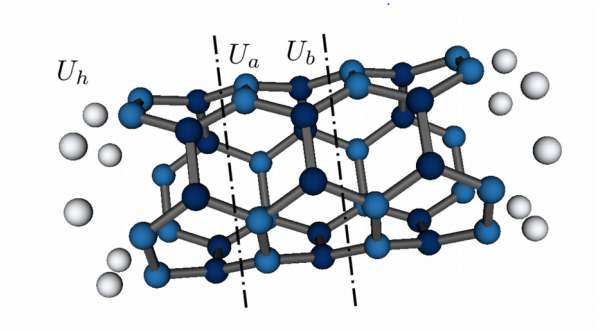




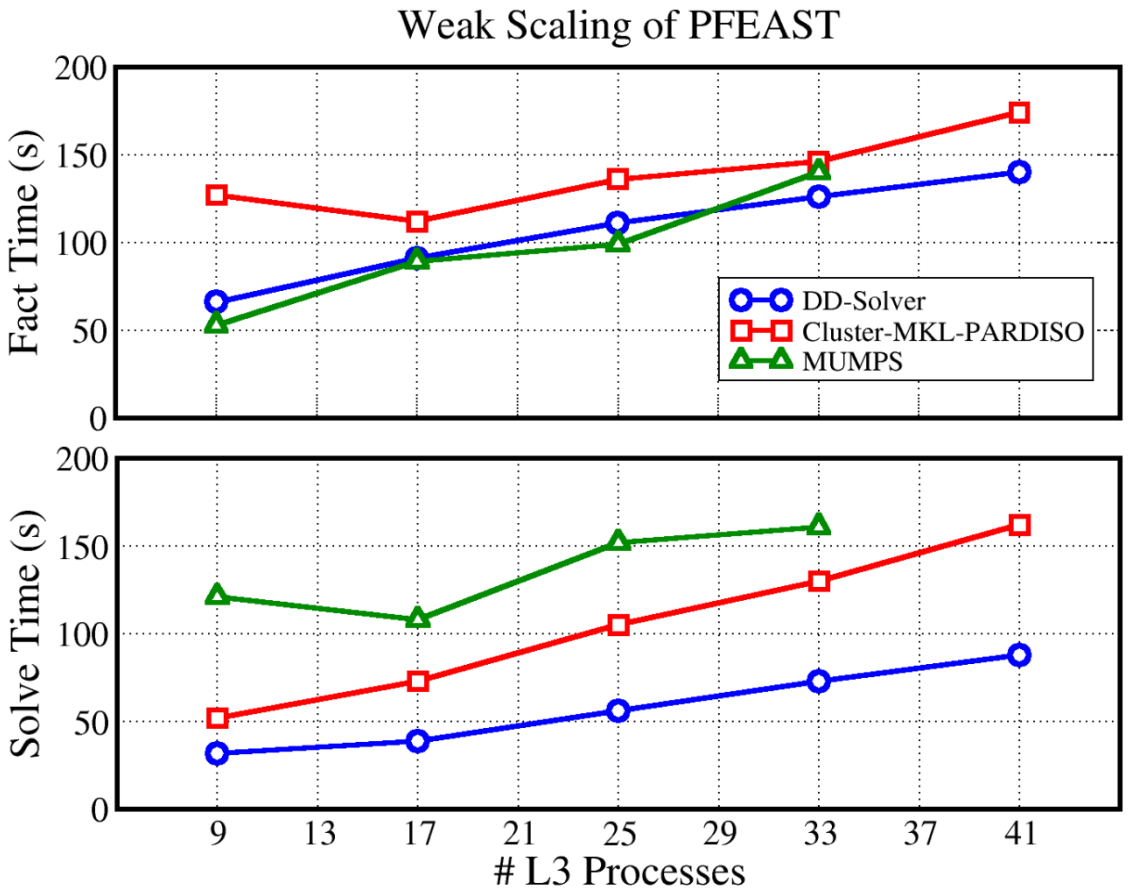
N=302,295; M0=200; 16 Quadrature Point

Results: Weak Scalability

# MPI	9	17	25	33	41
# Atoms	54	102	150	198	246
System Size	211,110	392,650	575,760	757,934	942,157



M0=200, 16 Contour Points (no **L2**); 1 FEAST Iteration



Example 246 Atom CNT

Need 750 eigenvectors
#**L1**=5 → 5 slices M0=200

#**L2**=16 (Optimal Case)
#**L3**=41

(#**L1**)x(#**L2**)x(#**L3**)
= **3,280** MPI
= **39,360** Cores

Conclusion

FEAST v4.0 (PFEAST) Coming Soon!

- 3 Levels of MPI Parallelism
- Reduce Memory with **L1** and **L3**
- 1D Eigenvector Distribution
- RCI for Custom Solver
- Predefined Interfaces for MUMPS and MKL-Cluster-Pardiso

- Scale to Larger Molecules (up to 1000 atoms) with Electronic Structure Application