Scalable Parallel Sparse Matrix Computations

Ahmed Sameh
Computer Science, Purdue University

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

after HSL-MC73 after TraceMIN-Fiedler

obtaining the Fiedler vector via the eigensolver: TraceMIN
(Wisniewski and A.S. -- SINUM, ’82)
Relative bandwidth (RBW):

$$\frac{\sum |a(i,j)| \text{ within band}}{\text{total } \sum |a(i,j)|}$$

Reordering

A := BCSSTK22

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

half-bandwidth: $k$
A Hybrid Sparse Linear System Solver: \textit{PSPIKE}
Target Computational Loop

- **Integration**
- **Newton iteration**
- **Linear system solvers**
  - require relative residuals of $O(10^{-5}$ or $10^{-6}$)

**Equations:**
- $\eta_k$
- $\epsilon_k$
- $\Delta t$
PSPIKE

Stage 1 –
• Extraction of an effective “banded” preconditioner $M$ – with or without reordering.
• $\text{norm}(M,’fro’) = (1- \varepsilon) \text{norm}(A,’fro’)$;
• “bandwidth” $\leq \beta(n)$.

Stage 2 –
• Use an outer Krylov subspace method ($\text{BiCGstab}$) to solve $A \times = f$ with preconditioner $M$.
• A modified $\text{Pardiso}$ (by O. Schenk) is a major kernel for solving $M \times = r$.
• Classes of banded preconditioners:
  narrow-banded, generalized-banded, wide-banded
a. PSPIKE-NBP

&

b. PSPIKE-MBP
UFL: f2 -- structural mechanics

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

Original matrix  After MC73  After TraceMin-Fiedler

TraceMIN-Fiedler: Murat Manguoglu et. al.
Before reordering

After reordering via TraceMIN-Fiedler

UFL – f2
\[ M \mathbf{z} = \mathbf{r} \quad (M \text{ is “banded”}) \]

Each \( M_{kk} \) is a general sparse matrix

\[ P = M + \delta(M) = D' \times S' \]

(i) Solve \( D'y = r \)

(ii) Solve \( S'z = y \)

Solving systems involving

The preconditioner \( P \mathbf{z} = \mathbf{r} \)

\( P = \text{PSPIKE: Pardiso} \) by Olaf Schenk
\( S = \text{SPIKE: Polizzi-Sameh} \)
Spike Matrix $S$ for 3 partitions

$D^{-1}A$ $X$ $D^{-1}f$

$I$ $*$ $= $

$\hat{\mathbf{S}}$ $\hat{x}$ $\hat{g}$ $*$ $= $

reduced system
Generating tips of the spikes

\[ M_{kk} \]

Obtain the upper and lower tips of the solution block via the modified direct sparse system solver “Pardiso”.
Parallel Scalability of PSPIKE_NBP vs. direct solvers
UFL – Rajat31 (circuit simulation)

\[ N \sim 4.7 \, M \]
\[ \text{nnz} \sim 20 \, M \]
non-symmetric
PSPIKE – WSMP – MUMPS

Time in seconds

100

~37

~25

~17

8 MPI proc./node

Total time for WSMP & MUMPS

factorization

PSPIKE: 1 MPI proc./node
8 OMP threads/MPI proc.

PSPIKE: β = 5

rel. res. = O(10^{-5})

# of cores (8 cores per node)

(128 nodes)
Pardiso vs. PSPIKE-MBP 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

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

PSPIKE: rel. residual $\leq 10^{-8}$
### PSPIKE vs. Pardiso

(rel. res. $\leq 10^{-8}$)

<table>
<thead>
<tr>
<th>MPI proc.</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>16</th>
<th>$T(\text{Pardiso}) \div T(\text{PSPIKE})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores: 1</td>
<td>400</td>
<td></td>
<td></td>
<td></td>
<td>.95</td>
</tr>
<tr>
<td>2</td>
<td>228</td>
<td>163</td>
<td></td>
<td></td>
<td>1.23</td>
</tr>
<tr>
<td>4</td>
<td>141</td>
<td>102</td>
<td>43</td>
<td></td>
<td>2.46</td>
</tr>
<tr>
<td>64</td>
<td>62</td>
<td>39</td>
<td>16</td>
<td>8</td>
<td>3.88</td>
</tr>
</tbody>
</table>
System 2: Matrix – vanHeukelum/cage13
(DNA electrophoresis, polymer. A. van Heukelum, Utrecht U)

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

PSPIKE:
rel. residual ≤ 10^{-8}
**PSPIKE vs. Pardiso**

*(rel. res. ≤ 10^{-8})*

<table>
<thead>
<tr>
<th>MPI proc.</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>16</th>
<th>$T(Pardiso) ÷ T(PSPIKE)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores = 1</td>
<td>21,266</td>
<td></td>
<td></td>
<td></td>
<td>~ 1</td>
</tr>
<tr>
<td>2</td>
<td>12,034</td>
<td>5,309</td>
<td></td>
<td></td>
<td>~ 2</td>
</tr>
<tr>
<td>4</td>
<td>6,223</td>
<td>3,033</td>
<td>851</td>
<td></td>
<td>~ 6</td>
</tr>
<tr>
<td>64</td>
<td>1,055</td>
<td>584</td>
<td>165</td>
<td>12</td>
<td>~ 182</td>
</tr>
</tbody>
</table>
Robustness & Parallel Scalability of PSPIKE-NBP vs. preconditioned iterative solvers
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

<table>
<thead>
<tr>
<th># of nodes</th>
<th>Time (Trilinos-ML) ÷ Time (PSPIKE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
</tr>
</tbody>
</table>

**Time (Trilinos-ML) ÷ Time (PSPIKE)**

**PSPIKE: k threads per MPI process**

**break-even @ 4 nodes**

**Preconditioner bw: β = 5**

**Intel Harpertown**

**MEMS benchmark 1**

<table>
<thead>
<tr>
<th># of nodes k</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 4</td>
</tr>
<tr>
<td>8 to 16</td>
</tr>
<tr>
<td>&gt; 16</td>
</tr>
</tbody>
</table>

**Speed**
Strong Scalability on Intel Nehalem for a MEMS system of order ~ 23M (benchmark 2)

Speed improvement: ~ 7.7
Efficiency: ~ 48%

Time (sec)

Number of cores (128 nodes)
c. PSPIKE-WBP
Circuit Simulation

Original Matrix

after TraceMIN-Fiedler

Reordered Matrix

Wide-Banded Preconditioner
WBP: wide-banded preconditioners

preconditioner consists of overlapped blocks

Solve: $Ax = f$ ; $A :=$ nonsingular

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22} & A_{23} \\
A_{32} & A_{33}
\end{bmatrix}
\begin{bmatrix} x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix} f_1 \\
f_2 \\
f_3
\end{bmatrix}
\]

\[
\begin{align*}
(A^{(1)}_{11} & A^{(1)}_{12}) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ \alpha f_2 + y \end{bmatrix}, & \quad 0 < \alpha < 1 \\
(A^{(2)}_{11} & A^{(2)}_{12}) \begin{bmatrix} \hat{x}_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} (1 - \alpha) f_2 - y \\ f_3 \end{bmatrix}
\end{align*}
\]

\[
A_{22} = A^{(1)}_{22} + A^{(2)}_{11} \in \mathbb{R}^{m \times m}
\]

$m << n$
Determine that $\dot{y}$ which assures that $x_2 = \hat{x}_2$.

$\dot{y}$ is the solution of the balance system (order = size of overlap)

$$My = g$$

$M$ and $g$ are not available explicitly.
The Balance System

• Solve the system $M y = g$ using a Krylov subspace scheme (CG or Bicgstab).

• Two time-consuming kernels:
  – $q = M^* p$
  – $r(p) = g - M^* p$
Observations

• $r(p) = g - M*p$
  
  $= \hat{x}_2(p) - x_2(p)$

• $r(0) = g$
  
  $= \hat{x}_2(0) - x_2(0)$

• $q = M*p$
  
  $= r(0) - r(p)$
**PSPIKE for ****ASIC680K system**

*relative residual*

<table>
<thead>
<tr>
<th>Method</th>
<th>Relative Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pardiso</td>
<td>$O(10^{-9})$</td>
</tr>
<tr>
<td>ILU+BiCGstab</td>
<td>$O(10^{-7})$</td>
</tr>
<tr>
<td>PSPIKE-WBP</td>
<td>$O(10^{-7})$</td>
</tr>
</tbody>
</table>

**Time on 1 node:**
- Pardiso: 23 (sec).
- ILU-BiCGstab: 34 (sec).

**Time on 2 nodes:**
- PSPIKE: 6 (sec).
Preconditioner

Overlapped diagonal blocks

PSPIKE-WBP
ASIC_680k_8
nonsymmetric
size: ~5.5 M
nnz: ~31.0 M

Preconditioner:
16 overlapped diagonal blocks
overlap size = 100

N = 16 m – 15 β
PSPIKE-WBP vs. ILU-preconditioned BiCGstab

- **ILUT-BiCGstab (one core):**
  - \( \text{drop tolerance} = 10^{-3} \)
  - \( \text{fill\_in per row} = 10\% \)
  - 16 iterations, \( \text{rel. res.} = O(10^{-6}) \)
  - Time \( \sim 276 \text{ sec.} \)

- **PSPIKE-WBP:**
  - 1 iteration, \( \text{rel. res} = O(10^{-9}) \)
**PSPIKE-WBP (multiple nodes)**

vs.

**ILU-preconditioned BiCGstab (one core)**

<table>
<thead>
<tr>
<th># of cores</th>
<th>32 (4 nodes)</th>
<th>64 (8 nodes)</th>
<th>128 (16 nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time (sec.)</strong></td>
<td>113</td>
<td>31</td>
<td>9</td>
</tr>
<tr>
<td><strong>Speed improv. over ILU-BiCGstab (one core)</strong></td>
<td>2.4</td>
<td>8.9</td>
<td>30.7</td>
</tr>
</tbody>
</table>
A Parallel Symmetric Eigenvalue Problem Solver: TraceMIN
The Trace minimization scheme:

\[ Ax = \lambda Bx \; ; \; \text{obtain the } p \text{ smallest eigenpairs} \]

\[ A = A^T \; ; \; B: \text{s.p.d} \]

\[
\begin{align*}
\min_{Y^TBY=I_p} & \quad \text{tr}(Y^TAY) = \sum_{i=1}^{p} \lambda_i \\
\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p < \lambda_{p+1} \leq \cdots \leq \lambda_n \\
Y & \in \mathbb{R}^{n \times p} \; ; \; p \ll n.
\end{align*}
\]

A.S. & J. Wisniewski: SINUM, 1982
\begin{align*}
Y_k^T AY_k &= \Sigma_k = \text{diag}(\sigma_1^{(k)}, \ldots, \sigma_p^{(k)}) \\
Y_k^T BY_k &= I_p \\
Y_{k+1} &= (Y_k - \Delta_k)S_k \\
\begin{array}{l}
\min \quad \text{tr} \left[ (Y_k - \Delta_k)^T A(Y_k - \Delta_k) \right] \\
\text{s.t.} \quad Y_k^T B\Delta_k = 0
\end{array}
\end{align*}

\textbf{Note: if } A \text{ were s.p.d. we have } p \text{ indep. problems of the form:}

\begin{align*}
\min \quad & (y_j^{(k)} - d_j^{(k)})^T A(y_j^{(k)} - d_j^{(k)}) \\
\text{s.t.} \quad & Y_k^T Bd_j^{(k)} = 0 \quad j = 1, 2, \ldots, p
\end{align*}
TraceMin (Outer iterations)

- relative residual $\leq \varepsilon_{out}$

  - form a section

  \[ Y^TAY = \Sigma; Y^TBY = I_p \]

  - 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

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

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

• convergence rate is ultimately cubic.

• $\nu_j$’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)
$A x = \lambda x$; lowest 4 eigenpairs

$A$: Poisson operator

Matrix size: 64 million

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 $B$

$n \sim 1.5$ Million
Car Body Problem

\[ A x = \lambda B x \]

Matrix size: 1.5 million

B is computationally singular

TraceMIN on one node: 8070 sec
TraceMIN on 64 nodes: 248 sec
Speed improv. = 32.5

Both LOBPCG & BKS failed for this problem!
Sampling the spectrum via TraceMIN

4 eigenpairs closest to $\alpha_j$, $j = 1, 2, \ldots, 100$

(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

<table>
<thead>
<tr>
<th>Time in seconds</th>
<th>Relative Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>$10^{-5}$</td>
</tr>
<tr>
<td>21</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>22</td>
<td>$10^{-9}$</td>
</tr>
</tbody>
</table>
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 (2 to 64 nodes): \(~ 17\)
- Efficiency: \(~ 53\%\)
Thank you!