An Interactive Environment for Combinatorial Scientific Computing

Viral B. Shah
John R. Gilbert
Steve Reinhardt

With thanks to: Brad McRae, Stefan Karpinski, Vikram Aggarwal, Min Roh
HPC today is exciting!
## Complex software stack

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational ecology, CFD, data exploration</td>
</tr>
<tr>
<td>Preconditioned Iterative Methods</td>
</tr>
<tr>
<td>CG, BiCGStab, etc. + combinatorial preconditioners (AMG, Vaidya)</td>
</tr>
<tr>
<td>Graph Analysis &amp; PD Toolbox</td>
</tr>
<tr>
<td>Graph querying &amp; manipulation, connectivity, spanning trees, geometric partitioning, nested dissection, NNMF, . . .</td>
</tr>
<tr>
<td>Distributed Sparse Matrices</td>
</tr>
<tr>
<td>Arithmetic, matrix multiplication, indexing, solvers (, eigs)</td>
</tr>
</tbody>
</table>
A = rand(4000*p, 4000*p);
x = randn(4000*p, 1);
y = zeros(size(x));
while norm(x-y) / norm(x) > 1e-11
    y = x;
    x = A*x;
    x = x / norm(x);
end;
Parallel sorting

- Simple, widely used combinatorial primitive
- \([V, \text{perm}] = \text{sort} (V)\)
- Used in many sparse matrix and array algorithms: sparse(), indexing, concatenation, transpose, reshape, repmat etc.
- Communication efficient
Sorting performance

Time spent in different phases of Psort
(192 processors on SGI Altix)

- Sequential sorting
- Splitters using medians
- Communication
- Merging
- Total time

Problem Size

1E+06 1E+07 1E+08 1E+09 1E+10 1E+11
Each processor stores:

- # of local nonzeros (# local edges)
- range of local rows (local vertices)
- nonzeros in a compressed row data structure (local edges)
Sparse matrix operations

- `dsparse` layout, same semantics as `ddense`
- Matrix arithmetic: `+`, `max`, `sum`, etc.
- `matrix * matrix` and `matrix * vector`
- Matrix indexing and concatenation
  \[
  A (1:3, [4 5 2]) = [B(:, J) C] ;
  \]
- Linear solvers: `x = A \ b`; using MUMPS/SuperLU (MPI)
- Eigensolvers: `[V, D] = eigs(A);` using PARPACK (MPI)
Sparse matrix multiplication

See A. Buluc (MS42, Fri 10am)

for \( j = 1:n \)

\[
C(:, j) = A \times B(:, j)
\]

All matrix columns and vectors are stored compressed except the SPA.
Interactive data exploration

A graph plotted with relaxed Fiedler co-ordinates
A 2-D density spy plot

Density spy plot of an R-MAT power law graph
Breadth-first search: sparse matvec

- Multiply by adjacency matrix $\rightarrow$ step to neighbor vertices
- Work-efficient implementation from sparse data structures
Maximal independent set

\[
\text{degree} = \text{sum}(G, 2);
\]
\[
\text{prob} = 1 ./ (2 \times \text{deg});
\]
\[
\text{select} = \text{rand} (n, 1) < \text{prob};
\]

if ~isempty (select & (G * select);
    \% keep higher degree vertices
end

\[
\text{IndepSet} = [\text{IndepSet} \text{select}];
\]

neighbor = neighbor | (G * select);
remain = neighbor == 0;
G = G(remain, remain);
A graph clustering benchmark

Fine-grained, irregular data access

- Many tight clusters, loosely interconnected
- Vertices and edges permuted randomly
Clustering by BFS

- Grow local clusters from many seeds in parallel
- Breadth-first search by sparse matrix * matrix
- Cluster vertices connected by many short paths

```matlab
% Grow each seed to vertices
% reached by at least k
% paths of length 1 or 2

C = sparse(seeds, 1:ns, 1, n, ns);
C = A * C;
C = C + A * C;
C = C >= k;
```
Clustering by peer pressure

\[
[\text{ignore, leader}] = \max(G);
\]

\[
S = \text{sparse(leader,1:n,1,n,n)} \ast G;
\]

\[
[\text{ignore, leader}] = \max(S);
\]

- Each vertex votes for its highest numbered neighbor as its leader
- Number of leaders is roughly the same as number of clusters
- Matrix multiplication gathers neighbor votes
- \(S(i,j)\) is the number of votes for \(i\) from \(j\)'s neighbors
Scaling up

- Graph with 2 million nodes, 321 million directed edges, 89 million undirected edges, 32 thousand cliques
- Good scaling observed from 8 to 120 processors of an SGI Altix

![Graph with 2 million nodes, 321 million directed edges, 89 million undirected edges, 32 thousand cliques. Good scaling observed from 8 to 120 processors of an SGI Altix.](#)
Graph of Poisson’s Equation on a 2D grid

\[ G = \text{grid5 (10)}; \]
Spanning trees

Maximum weight spanning tree

\[ T = \text{mst}(G, \text{max}); \]
A combinatorial preconditioner
V. Aggarwal

Augmented Vaidya’s preconditioner

\[ V = \text{vaidya\_support} \left( G \right); \]
Quadtree meshes and AMG
V. Aggarwal and M. Roh
• Non-negative matrix factorizations (NNMF) for wireless traffic modeling

• NNMF algorithms combine linear algebra and optimization methods

• Basic and “improved” NMF factorization algorithms implemented:
  – euclidean (Lee & Seung 2000)
  – K-L divergence (Lee & Seung 2000)
  – semi-nonnegative (Ding et al. 2006)
  – left/right-orthogonal (Ding et al. 2006)
  – bi-orthogonal tri-factorization (Ding et al. 2006)
  – sparse euclidean (Hoyer et al. 2002)
  – sparse divergence (Liu et al. 2003)
  – non-smooth (Pascual-Montano et al. 2006)
A meta-algorithm

spherical k-means

ANLS

K-L div.

same as CDFs
• Landscape connectivity governs the degree to which the landscape facilitates or impedes movement.

• Need to model important processes like:
  – Gene flow (to avoid inbreeding)
  – Movement and mortality patterns

• Corridor identification, conservation planning
Pumas in southern California

Habitat quality model

Joshua Tree National Park

Los Angeles

Palm Springs
Model as a resistive network
Processing landscapes

Combinatorial methods
Graph construction
Graph contraction
Connected components

Numerical methods
Linear systems
Combinatorial preconditioners
Results

• Solution time reduced from 3 days to 5 minutes for typical problems
• Aiming for much larger problems: Yellowstone-to-Yukon (Y2Y)
## Multi-layered software tools

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational ecology, CFD, data exploration</td>
</tr>
<tr>
<td>Preconditioned Iterative Methods</td>
</tr>
<tr>
<td>Graph querying &amp; manipulation, connectivity, spanning trees, geometric partitioning, nested dissection, NNMF, . . .</td>
</tr>
</tbody>
</table>

Graph Analysis & PD Toolbox

<table>
<thead>
<tr>
<th>Distributed Sparse Matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic, matrix multiplication, indexing, solvers (, eigs)</td>
</tr>
</tbody>
</table>
Thanks for coming