Parallel Sparse Matrix Indexing and Assignment

Aydın Buluç
Lawrence Berkeley National Laboratory

John R. Gilbert
University of California, Santa Barbara
• Every graph is a sparse matrix and vice-versa
• Adjacency matrix: sparse array w/ nonzeros for graph edges
• Storage-efficient implementation from sparse data structures
Sparse matrix-matrix Multiplication (SpGEMM)

Element-wise operations

Sparse matrix-sparse vector multiplication

Sparse Matrix Indexing

Matrices on semirings, e.g. (×, +), (and, or), (+, min)
Indexed reference and assignment

Matlab internal names: \texttt{subsref}, \texttt{subsasgn}
For sparse special case, we use: \texttt{SpRef}, \texttt{SpAsgn}

\begin{align*}
\textbf{SpRef:} & \quad B = A(I, J) \\
\textbf{SpAsgn:} & \quad B(I, J) = A
\end{align*}

\(A, B:\) sparse matrices
\(I, J:\) vectors of indices

\texttt{SpRef} using mixed-mode sparse matrix-matrix multiplication (\texttt{SpGEMM}). Ex: \(B = A([2,4], [1,2,3])\)
Why are **SpRef/SpAsgn** important?

Subscripting and colon notation:

⇒ Batched and vectorized operations
⇒ High Performance and parallelism.

\[
\begin{align*}
A &= \text{rmat}(15) \\
A(r,r) : r &\text{ random} \\
A(r,r) : r &= \text{symrcm}(A)
\end{align*}
\]

\begin{itemize}
  \item Load balance hard
  \item + Some locality
  \item – No locality
  \item Load balance easy
  \item + Good locality
  \item – Load balance hard
\end{itemize}
More applications

Prune isolated vertices; plug-n-play way (Graph 500)

```
sa = sum(A); // A is symmetric, for undirected graph
nonisov = find(sa>0);
A= A(nonisov, nonisov); // keep only connected vertices
```
More applications

Extracting (induced) subgraphs

- Per-area analysis on power grids
- Subroutine for recursive algorithms on graphs
**Sequential algorithms**

```matlab
function B = spref(A,I,J)
R = sparse(1:length(I),I,1,length(I),size(A,1));
Q = sparse(J,1:length(J),1,size(A,2),length(J));
B = R*A*Q;

T_{spref} = flops(R \cdot A) + flops(RA \cdot Q) = nnz(R \cdot A) + nnz(RA \cdot Q) = O(nnz(A))
```

```matlab
function C = spasgn(A,I,J,B)
[ma,na] = size(A);
[mb,nb] = size(B);
R = sparse(I,1:mb,1,ma,mb);
Q = sparse(1:nb,J,1,nb,na);
S = sparse(I,I,1,ma,ma);
T = sparse(J,J,1,na,na);
C = A + R*B*Q - S*A*T;

A + \begin{pmatrix}
0 & 0 & 0 \\
0 & B & 0 \\
0 & 0 & 0
\end{pmatrix} - \begin{pmatrix}
0 & 0 & 0 \\
0 & A(I,J) & 0 \\
0 & 0 & 0
\end{pmatrix}

T_{spasgn} = O(nnz(A))
```
1. Forming R from I in parallel, on a 3x3 processor grid

- Vector distributed only on diagonal processors; for illustration.
- Full (2D) vector distribution: SCATTER \rightarrow ALLTOALLV
- Forming $Q^T$ from J is identical, followed by $Q=Q^T$.Transpose()
2. **SpGEMM** using memory-efficient Sparse SUMMA.

Minimize temporaries by:
- Splitting local matrix, and broadcasting multiple times
- Deleting P (and A if in-place) immediately after forming $C=P*A$
2D vector distribution

Matrix/vector distributions, interleaved on each other.

Default distribution in Combinatorial BLAS.

- Performance change is marginal (dominated by SpGEMM)
- Scalable with increasing number of processes
- No significant load imbalance
Complexity analysis

**SpGEMM:**

\[ T_{\text{comp}} \approx \Theta \left( \frac{\text{nnz}(A)}{p} \cdot \log \left( \frac{\text{length}(I)}{p} + \frac{\text{length}(J)}{p} + \sqrt{p} \right) \right) \]

\[ T_{\text{comm}} = \Theta \left( \alpha \cdot \sqrt{p} + \beta \cdot \frac{\text{nnz}(A)}{\sqrt{p}} \right) \]

**Matrix formation:**

\[ \Theta \left( \alpha \cdot \log(p) + \beta \cdot \frac{\text{length}(I) + \text{length}(J)}{\sqrt{p}} \right) \]

Assumptions:
- The triple product is evaluated from left to right: \( B = (R \times A) \times Q \)
- Nonzeros uniformly distributed to processors (chicken-egg?)

Dominated by SpGEMM
Bottleneck: bandwidth costs
Speedup: \( \Theta \left( \sqrt{p} \right) \)
Strong scaling of **SpRef**

random symmetric permutation $\Leftrightarrow$ relabeling graph vertices
- RMAT Scale 22; edge factor=8; $a=.6$, $b=c=d=.4/3$
- Franklin/NERSC, each node is a quad-core AMD Budapest
Strong scaling of \textbf{SpRef}

Extracts 10 random (induced) subgraphs, each with $|V|/10$ vert.
Higher span $\rightarrow$ Decreased parallelism $\rightarrow$ Lower speedup
Conclusions

- Parallel algorithms for \texttt{SpRef} and \texttt{SpAsgn}
- Systemic algorithm structure imposed by \texttt{SpGEMM}
- Analysis made possible for the general case
- Good strong scaling for 1000-way concurrency
- Many applications on sparse matrix and graph world.

\textbf{Caveat:} Avoid load imbalance by indexing \textit{non-monotonically}

\[ I = \begin{bmatrix}
    1 \\
    3 \\
    4 \\
    6 \\
    7 \\
    8
\end{bmatrix} \quad \text{and} \quad I = \begin{bmatrix}
    7 \\
    3 \\
    8 \\
    1 \\
    6 \\
    4
\end{bmatrix} \]