Communication-Avoiding Linear-Algebraic Primitives For Graph Analytics

Aydın Buluç
Berkeley Lab (LBNL)
May 19, 2014
GABB’14 at IPDPS
The Combinatorial BLAS implements these, and more, on arbitrary semirings, e.g. $(\times, +), (\text{and, or}), (+, \text{min})$
### Some Combinatorial BLAS functions

*Grossly simplified (parameters, semiring)*

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Returns</th>
<th>Math Notation</th>
</tr>
</thead>
</table>
| SpGEMM   | - sparse matrices A and B  
- unary functors (op) | sparse matrix | $C = \text{op}(A) \times \text{op}(B)$ |
| SpM{Sp}V | - sparse matrix A  
- sparse/dense vector x | sparse/dense vector | $y = A \times x$ |
| SpEWiseX | - sparse matrices or vectors  
- binary functor and predicate | in place or sparse matrix/vector | $C = A \circ B$ |
| Reduce   | - sparse matrix A and functors | dense vector | $y = \text{sum}(A, \text{op})$ |
| SpRef    | - sparse matrix A  
- index vectors p and q | sparse matrix | $B = A(p,q)$ |
| SpAsgn   | - sparse matrices A and B  
- index vectors p and q | none | $A(p,q) = B$ |
| Scale    | - sparse matrix A  
- dense matrix or vector X | none | check manual |
| Apply    | - any matrix or vector X  
- unary functor (op) | none | $\text{op}(X)$ |
Many irregular applications contain coarse-grained parallelism that can be exploited by abstractions at the proper level.

<table>
<thead>
<tr>
<th>Traditional graph computations</th>
<th>Graphs in the language of linear algebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data driven, unpredictable communication.</td>
<td>Fixed communication patterns</td>
</tr>
<tr>
<td>Irregular and unstructured, poor locality of reference</td>
<td>Operations on matrix blocks exploit memory hierarchy</td>
</tr>
<tr>
<td>Fine grained data accesses, dominated by latency</td>
<td>Coarse grained parallelism, bandwidth limited</td>
</tr>
</tbody>
</table>
Matrix/vector distributions, interleaved on each other.

Default distribution in Combinatorial BLAS.

Scalable with increasing number of processes

- 2D matrix layout wins over 1D with large core counts and with limited bandwidth/compute
- 2D vector layout sometimes important for load balance

2D parallel BFS algorithm

ALGORITHM:
1. Gather vertices in *processor column* [communication]
2. Find owners of the current frontier’s adjacency [computation]
3. Exchange adjacencies in *processor row* [communication]
4. Update distances/parents for unvisited vertices. [computation]
Outline

• Communication-avoiding graph/sparse-matrix algorithms
• Why do they matter in science?
  – Isomap (non-linear dimensionality reduction) needs APSP
  – Parallel betweenness centrality needs sparse matrix-matrix product
• What kind of infrastructure/software support they need?

Communication-avoiding algorithms: **Save time**

![Graph showing energy consumption](image)

Image courtesy of John Shalf (LBNL)

Communication-avoiding algorithms: **Save energy**
Two kinds of costs:
- Arithmetic (FLOPs)
- Communication: moving data

\[ \text{Running time} = \gamma \cdot \#\text{FLOPs} + \beta \cdot \#\text{Words} + (\alpha \cdot \#\text{Messages}) \]

Develop faster algorithms: minimize communication (to lower bound if possible)

2004: trend transition into multi-core, further communication costs

\( P \) processors
- Often no surface to volume ratio.
- Very little data reuse in existing algorithmic formulations *
- Already heavily communication bound

2D sparse matrix-matrix multiply emulating:
- Graph contraction
- AMG restriction operations

Scale 23 R-MAT (scale-free graph) times order 4 restriction operator

Cray XT4, Franklin, NERSC

Graph Analysis for the Brain

- Connective abnormalities in schizophrenia [van den Heuvel et al.]
  - Problem: understand disease from anatomical brain imaging
  - Tools: betweenness centrality (BC), shortest path length
  - Results: global statistics on connection graph correlate w/ diagnosis

BC measures “influence”

\[ C_B(v) = \frac{\sum_{s \neq v \neq t \in V} \sigma_{st}(v)}{\sigma_{st}} \]

Among all the shortest paths, what fraction passes through \( v \)?

\[ Cost = O(mn) \]

- unweighted-

Computationally intensive!

<table>
<thead>
<tr>
<th>Input type</th>
<th>Resolution</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROIs</td>
<td>256</td>
<td>seconds</td>
</tr>
<tr>
<td>Voxels</td>
<td>60,000</td>
<td>hours</td>
</tr>
<tr>
<td>Neurons</td>
<td>1M+</td>
<td>years?</td>
</tr>
</tbody>
</table>
Distributed memory BC algorithm

Work-efficient parallel breadth-first search via parallel sparse matrix-matrix multiplication over semirings

Encapsulates three level of parallelism:
1. columns(B): multiple BFS searches in parallel
2. columns($A^T$)+rows(B): parallel over frontier vertices in each BFS
3. rows($A^T$): parallel over incident edges of each frontier vertex

Parallel sparse matrix-matrix multiplication algorithms

\[ C_{ij} += \text{HyperSparseGEMM}(A^{\text{recv}}, B^{\text{recv}}) \]

2D algorithm: Sparse SUMMA (based on dense SUMMA)
General implementation that handles rectangular matrices

B., Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. In ICPP’08
1D vs. 2D scaling for sparse matrix-matrix multiplication

In practice, 2D algorithms have the potential to scale, but not linearly.

\[ T_{\text{comm}}(2D) = \alpha p \sqrt{p} + \beta cn \sqrt{p} \]

\[ T_{\text{comp}}(\text{optimal}) = c^2 n \]
2D sparse matrix multiplication

Almost linear scaling until bandwidth costs starts to dominate

Scaling proportional to $\sqrt{p}$ afterwards

NERSC/Franklin Cray XT4

R-MAT, edgelfactor: 8

$a=0.6$, $b=c=d=0.4/3$
Matrix multiplication: \( \forall (i,j) \in n \times n, \quad C(i,j) = \Sigma_k A(i,k)B(k,j) \),

The computation (discrete) cube:

- A face for each (input/output) matrix
- A grid point for each multiplication

How about sparse algorithms?

1D algorithms

2D algorithms

3D algorithms
The computation cube and sparsity-independent algorithms

Matrix multiplication: \[ \forall (i,j) \in n \times n, \quad C(i,j) = \sum_k A(i,k)B(k,j), \]

**Sparsity independent** algorithms: assigning grid-points to processors is independent of sparsity structure.
- In particular: if \( C_{ij} \) is non-zero, who holds it?
- all standard algorithms are sparsity independent

Assumptions:
- Sparsity independent algorithms
- input (and output) are sparse:
- The algorithm is load balanced
Algorithms attaining lower bounds

Previous Sparse Classical:
\[ \Omega\left( \frac{\#FLOPs \cdot M}{(\sqrt{M})^3 \cdot P} \right) = \Omega\left( \frac{d^2 n}{P \sqrt{M}} \right) \]

No algorithm attain this bound!

[Ballard, et al. SIMAX’11]

New Lower bound for Erdős-Rényi(n,d):
\[ \Omega\left( \min\left\{ \frac{dn}{\sqrt{P}}, \frac{d^2 n}{P} \right\} \right) \]

Two new algorithms achieving the bounds (Up to a logarithmic factor)

i. Recursive 3D, based on [Ballard, et al. SPAA’12]
ii. Iterative 3D, based on [Solomonik & Demmel EuroPar’11]

No previous algorithm attain these.

**3D Iterative Sparse GEMM**

[Dense case: Solomonik and Demmel, 2011]

Optimal replicas: \( c = \Theta \left( \frac{p}{d^2} \right) \) (theoretically)

**3D Algorithms:**
Using extra memory (c replicas) reduces communication volume by a factor of \( c^{1/2} \) compared to 2D
Performance results (Erdős-Rényi graphs)
Iterative 2D-3D performance results (R-MAT graphs)

Seconds

<table>
<thead>
<tr>
<th>Number of grid layers (c)</th>
<th>Imbalance</th>
<th>Merge</th>
<th>HypersparseGEMM</th>
<th>Comm_Reduce</th>
<th>Comm_Bcast</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

- **p=4096 (s20)**
- **p=16384 (s22)**
- **p=65536 (s24)**

- 2X Faster
- 5X faster if load balanced
All-pairs shortest-paths problem

- **Input:** Directed graph with “costs” on edges
- Find least-cost paths between all reachable vertex pairs
- Classical algorithm: Floyd-Warshall

```plaintext
for k=1:n  // the induction sequence
  for i = 1:n
    for j = 1:n
      if(w(i→k)+w(k→j) < w(i→j))
        w(i→j):= w(i→k) + w(k→j)
```

- It turns out a previously overlooked **recursive version** is more parallelizable than the triple nested loop
\[
\begin{bmatrix}
0 & 5 & 9 & \infty & \infty & 4 \\
\infty & 0 & 1 & -2 & \infty & \infty \\
3 & \infty & 0 & 4 & \infty & 3 \\
\infty & \infty & 5 & 0 & 4 & \infty \\
\infty & \infty & -1 & \infty & 0 & \infty \\
-3 & \infty & \infty & \infty & 7 & 0
\end{bmatrix}
\]

+ is "min", \(\times\) is "add"

\[
A = A*; \quad % \text{recursive call}
B = AB; \quad C = CA;
D = D + CB;
D = D*; \quad % \text{recursive call}
B = BD; \quad C = DC;
A = A + BC;
\]
Communication-avoiding APSP on distributed memory

Bandwidth: \( W_{bc-2.5D}(n, p) = O\left(\frac{n^2}{\sqrt{cp}}\right) \)

Latency: \( S_{bc-2.5D}(p) = O\left(\sqrt{cp \log^2(p)}\right) \)

Optimal for any memory size!
Communication-avoiding APSP on distributed memory

65K vertex dense problem solved in about two minutes

Software for communication-avoiding graph kernels and beyond

- Combinatorial BLAS used a static 2D block data distribution
  - Significantly better than 1D in almost all cases
  - Most *communication-avoiding algorithms are 3D*
  - Infrastructure for *recursion in distributed-memory* is needed
  - Support for *automatic switching between dense/sparse formats*

- Combinatorial BLAS used text I/O and non-portable binary
  - Move onto future proof self-describing format such as HDF5
  - Build a community file format on top of HDF5

- Combinatorial BLAS accepts **graphs** as input
  - Often the graph needs to be constructed from data (G=ATT)

- Combinatorial BLAS API developed organically
  - Need to contribute to a standard interface
Conclusions

- Parallel graph libraries are crucial for analyzing big data.
- In addition to being high performance and scalable, the library should be flexible and simple to use.
- Linear-algebraic primitives should provide the core library.
- Avoiding communication improves performance and energy.

- KDT + Combinatorial BLAS was an excellent proof of concept.
- A more elegant, simpler, complete system to emerge that:
  - Is developed *concurrently with the Graph BLAS standard*
  - Implements *communication-avoiding algorithms*
  - Interoperates seamlessly with non-linear algebraic pieces such as the native graph systems like GraphLab and graph databases.
Acknowledgments

• Grey Ballard (UCB)
• Jarrod Chapman (JGI)
• Jim Demmel (UC Berkeley)
• John Gilbert (UCSB)
• Evangelos Georganas (UCB)
• Laura Grigori (INRIA)
• Ben Lipshitz (UCB)
• Adam Lugowski (UCSB)
• Steve Reinhardt (Cray)
• Dan Rokhsar (JGI/UCB)

• Lenny Oliker (Berkeley Lab)
• Oded Schwartz (UCB)
• Edgar Solomonik (UCB)
• Veronika Strnadova (UCSB)
• Sivan Toledo (Tel Aviv Univ)
• Dani Ushizima (Berkeley Lab)
• Kathy Yelick (Berkeley Lab/UCB)

This work is funded by:
Questions?