Parallel Primitives for Computation with Large Graphs

Aydın Buluç
John R. Gilbert
Challenges [Lumsdaine et al. 2007]

- Graph computations are data-driven
  - Unpredictable communication patterns
- Irregular and unstructured nature
  - Poor locality
- Fine grained data accesses
  - Latency dominated
An Architectural Approach - XMT

- Massively multithreaded machines
- No (or shallow) memory hierarchy
- Slower clock rates
- Uniform access time
- Highly scalable but not ubiquitous.
Our Approach – Sparse Matrices

- Sparse matrix primitives
  - On special semirings
  - \((\times, +)\); \((\text{and,or})\); \((+, \text{min})\); \ldots
- Oblivious
  - Fixed communication patterns
  - Easier to overlap communication
- Coarse grained parallelism
  - Exploit memory hierarchy
BFS from multiple sources

\[ A^T \]

\[ X \]
BFS from multiple sources

\[ A^T X \Rightarrow A^T X \]

- Work efficient implementation using sparse matrix-matrix multiplication (SpGEMM)
SpGEMM Applications

- Shortest path calculations (APSP)
- Betweenness centrality
- BFS from multiple source vertices
- Multigrid interpolation / restriction
- Subgraph / submatrix indexing
- Graph contraction
- Cycle detection
- Colored intersection searching
- Context-free parsing
1D algorithms can not scale beyond 40x
Break-even point is around 50 processors.
2D Example: Sparse SUMMA

- $C_{ij} += A_{ik} \times B_{kj}$
- At worst doubles local storage

- Based on SUMMA (block size = $n/\sqrt{p}$)
- Easy to generalize nonsquare matrices, etc.
Challenges of Parallel SpGEMM

- Scalable sequential kernel \( A_{ik} * B_{kj} \)
- Load balancing
  - Especially for real world graphs
- Communication costs
  - Communication to computation ratio is much higher than dense GEMM
- Updates (additions)
  - scalar additions ≠ scalar multiplications
Submatrices are *hypersparse*!

Any data structure that depends on the matrix dimension $n$ (such as CSR or CSC) is asymptotically too wasteful for submatrices.
Trends of different components

\[ n' \approx \frac{n}{\sqrt{p}} \]

\[ \text{nnz}' \approx \frac{\text{nnz}}{p} \]

\[ f' \approx \frac{f}{p\sqrt{p}} \]
Sequential Kernel [B&G 2008]

- Strictly $O(\text{nnz})$ data structure
- Complexity independent of matrix dimension
- Revival of outer-product formulation
- Heap assisted multi-way merging
Experiments with RMAT

Only submatrix multiplications are timed

\[ \sum_{i=0}^{\sqrt{p}} \sum_{j=0}^{\sqrt{p}} \sum_{k=0}^{\sqrt{p}} \text{time}(A_{ik} \times B_{kj}) \]
Addressing the Load Balance

- Random permutations are useful.
- Bulk synchronous algorithms may still suffer:

Asynchronous algorithms have **no notion of stages**.

![Worst load imbalance by stage](image1)

![Total A*B flops by processor](image2)

\[ m = 10277423, \ min = 8319099, \ \text{avg} = 9.1669e+006, \ \text{total} = 2.346726e+009; \ max/\text{avg} = 1.1 \]
Overlapping Communication

- Asynchronous, one sided communication (Again!)
- Can drop o from LogP model

GASNET, ARMCI
(Truly one-sided) Communication layers

Myrinet, Infiniband, etc
Hardware supporting zero copy RDMA
Conclusions

- SpGEMM is a key primitive
- Much harder than dense GEMM
- No fixed recipe
  - It won’t solve all your graph problems (as SpMV does not solve all your scientific problems)
- Highly scalable solution where applicable
- Widespread implementation on modern architectures (GPUs, Cell) would help.