Parallel Primitives for Computation with Large Graphs

Aydın Buluç
John R.Gilbert

UCSB

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

UCSB

An Architectural Approach - XMT

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

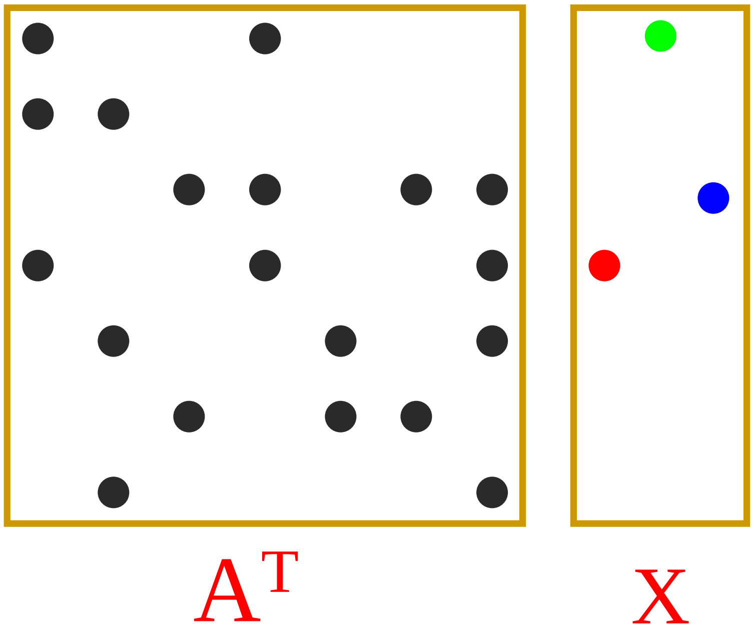
UCSB

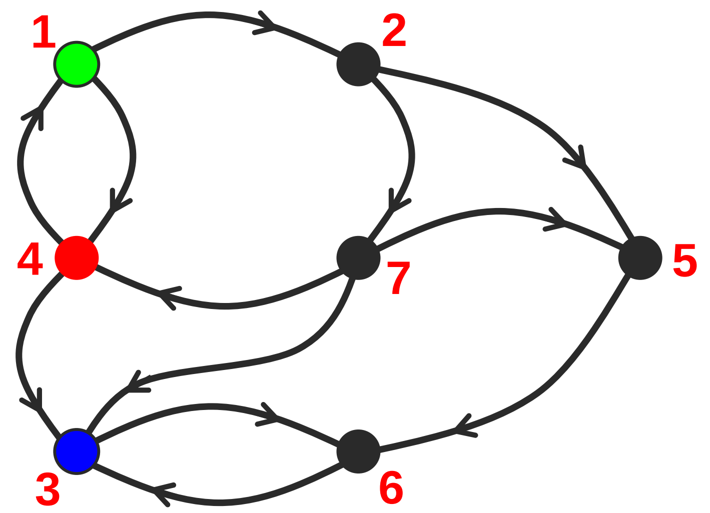
Our Approach – Sparse Matrices

- Sparse matrix primitives
  - On special semirings
  - (×,+) ; (and,or) ; (+,min) ; . . .
- Oblivious
  - Fixed communication patterns
  - Easier to overlap communication
- Coarse grained parallelism
  - Exploit memory hierarchy

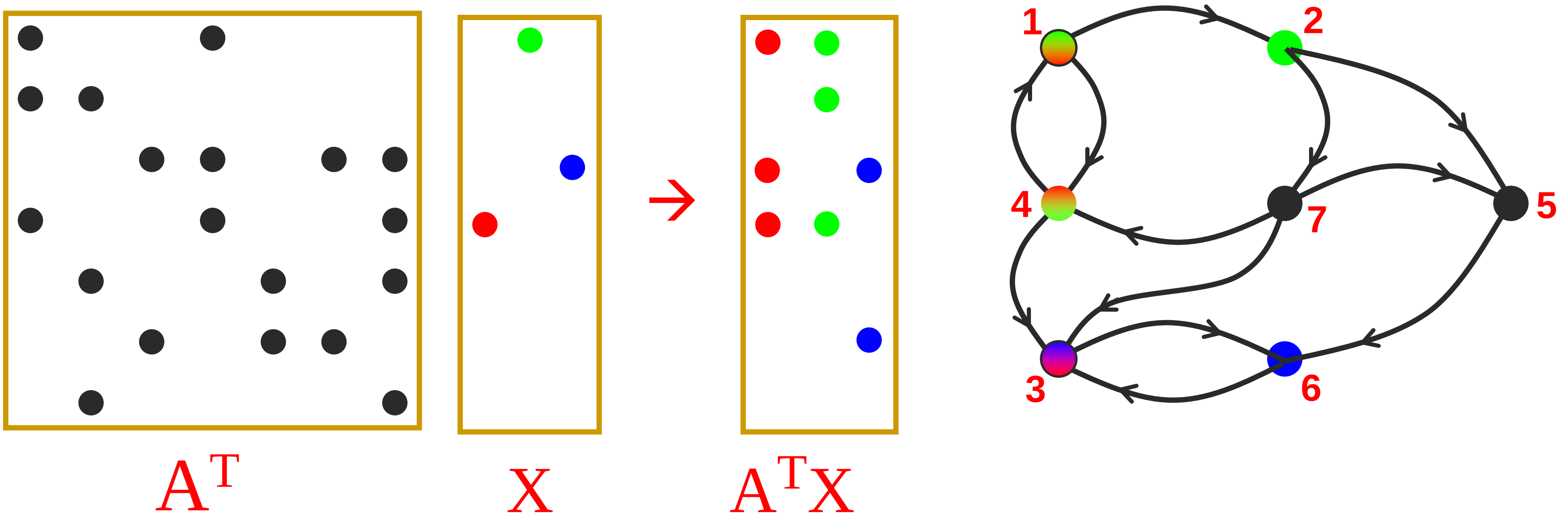
UCSB

BFS from multiple sources





BFS from multiple sources

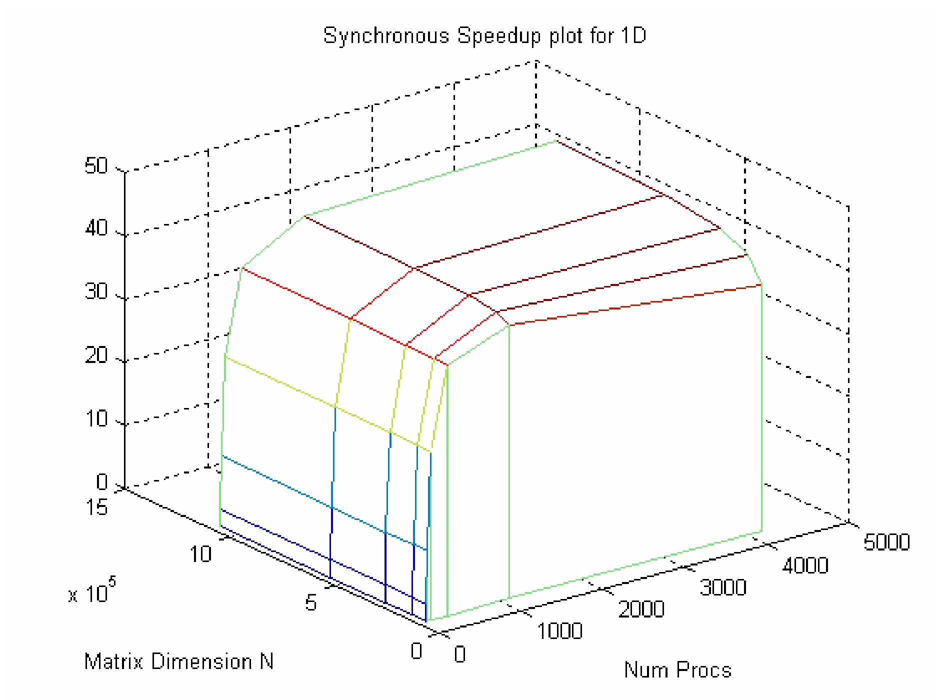


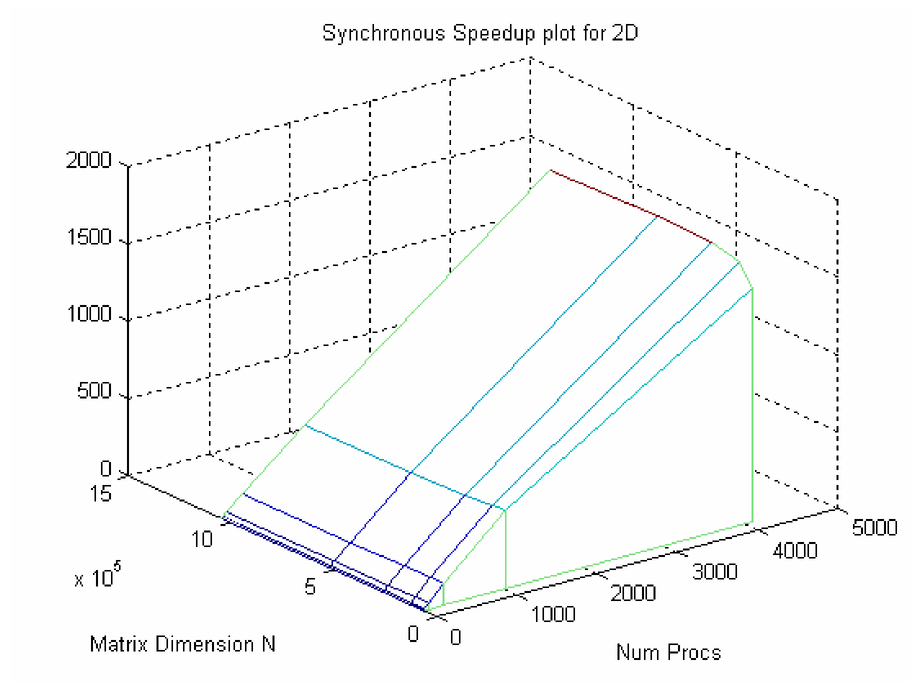
- Work efficient implementation using sparse matrix-matrix multiplication (SpGEMM)

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

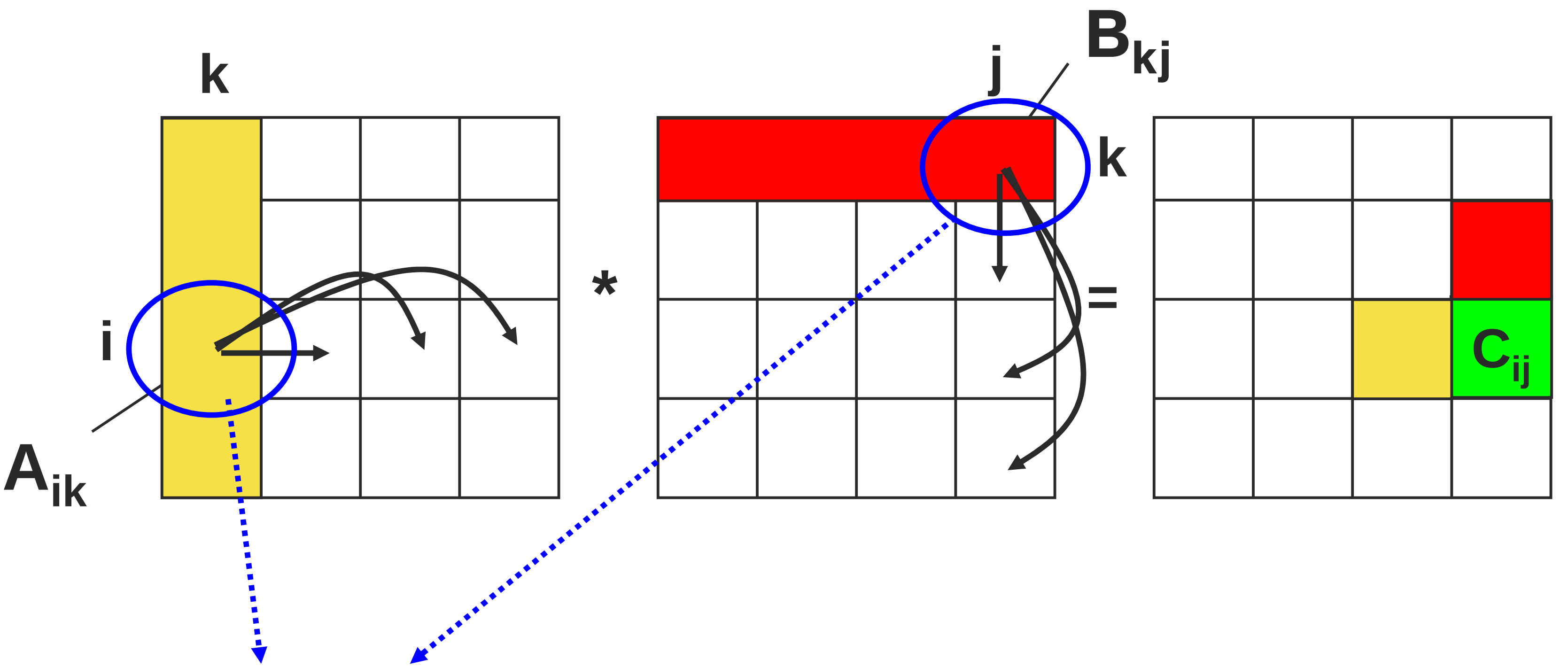
SpGEMM Data Distribution





- 1D algorithms can not scale beyond 40x
- Break-even point is around 50 processors.

2D Example: Sparse SUMMA



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

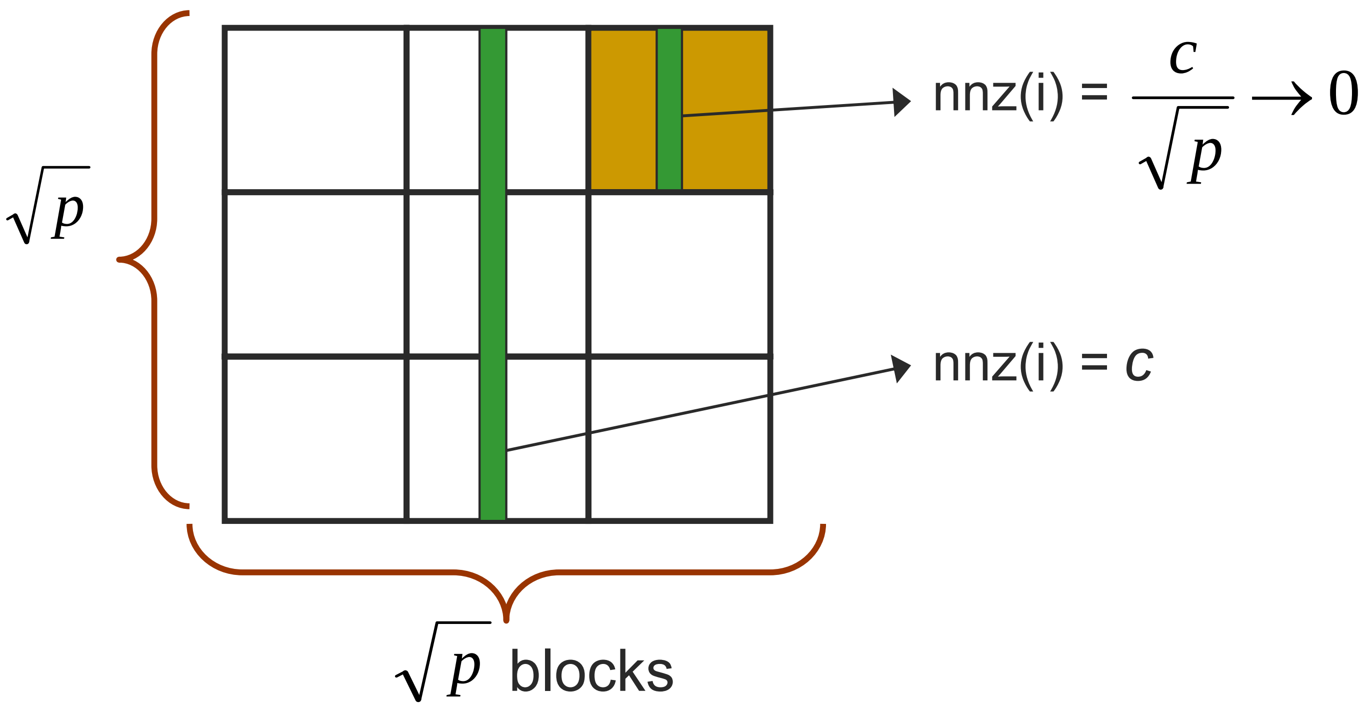
- Based on SUMMA (block size = n/sqrt(p))
- Easy to generalize nonsquare matrices, etc.

UCSB

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

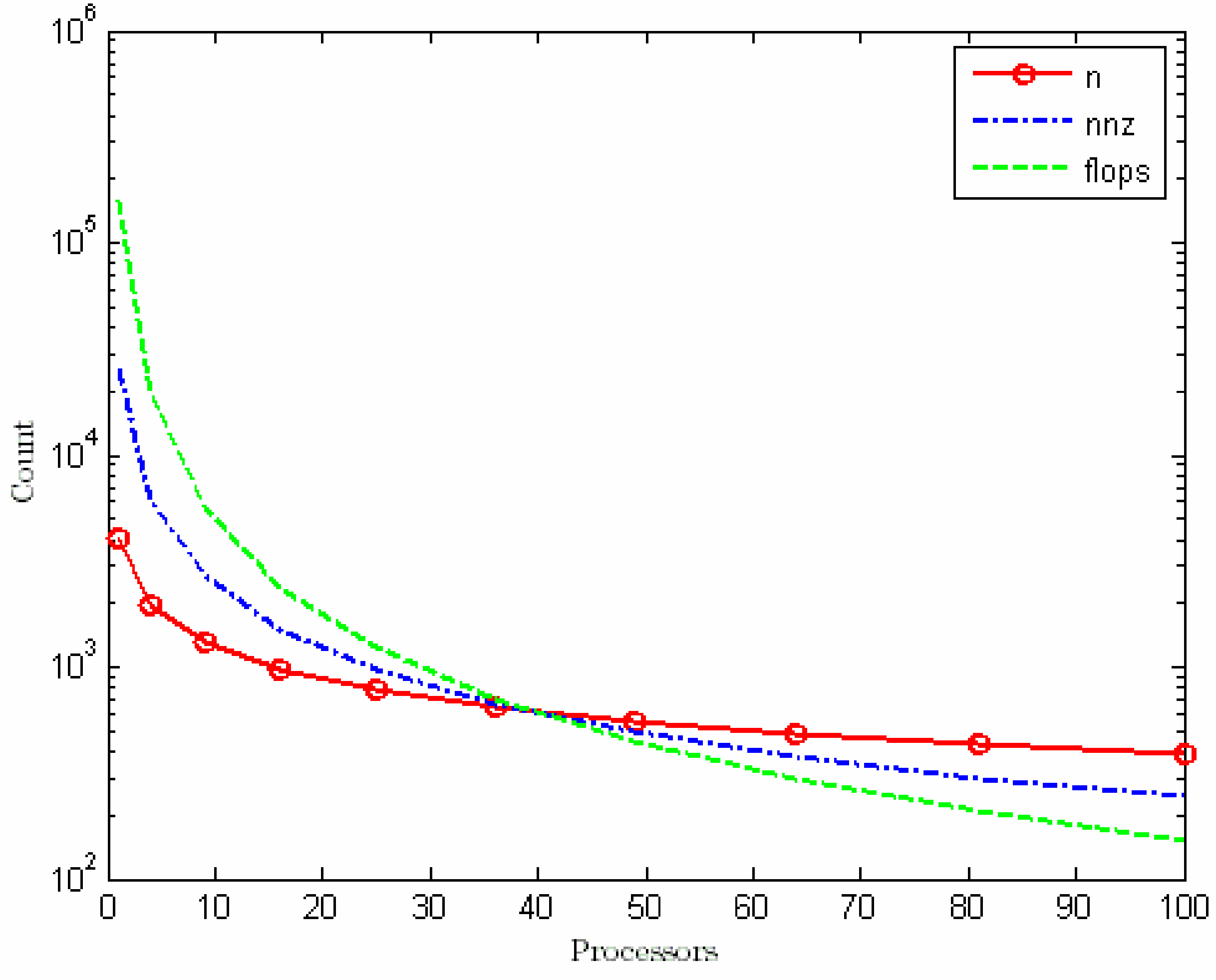
Submatrices are *hypersparse* !



- Any data structure that depends on the matrix dimension n (such as CSR or CSC) is asymptotically too wasteful for submatrices

UCSB

Trends of different components



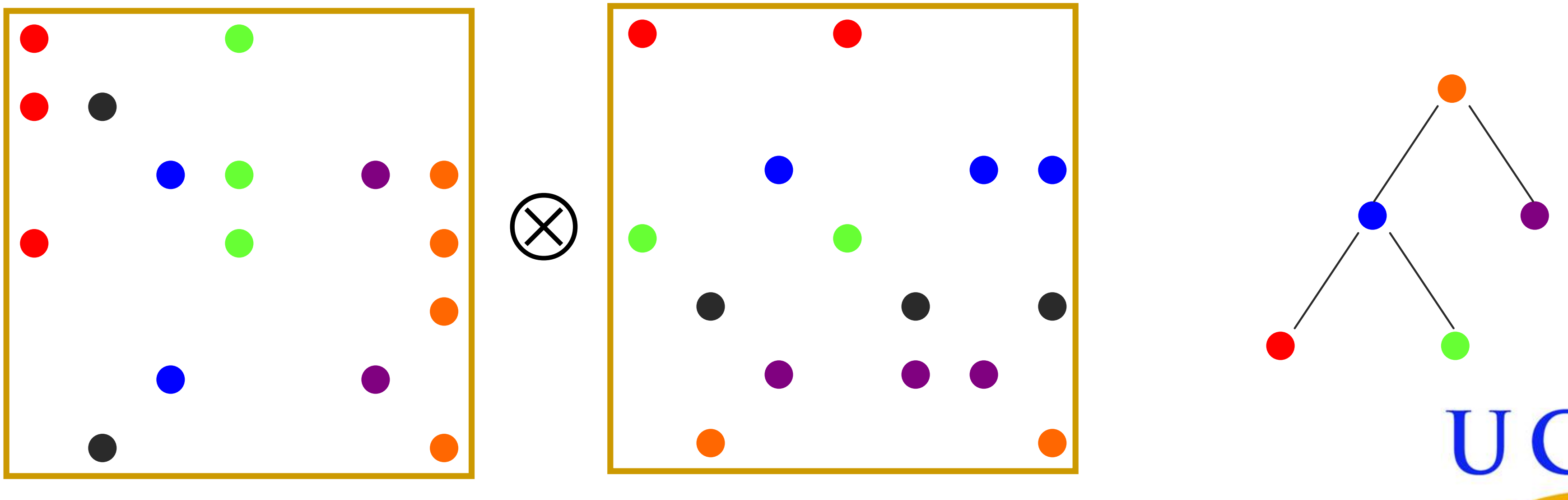
$$n' \approx \frac{n}{\sqrt{p}}$$

$$nnz' \approx \frac{nnz}{p}$$

$$f' \approx \frac{f}{p\sqrt{p}}$$

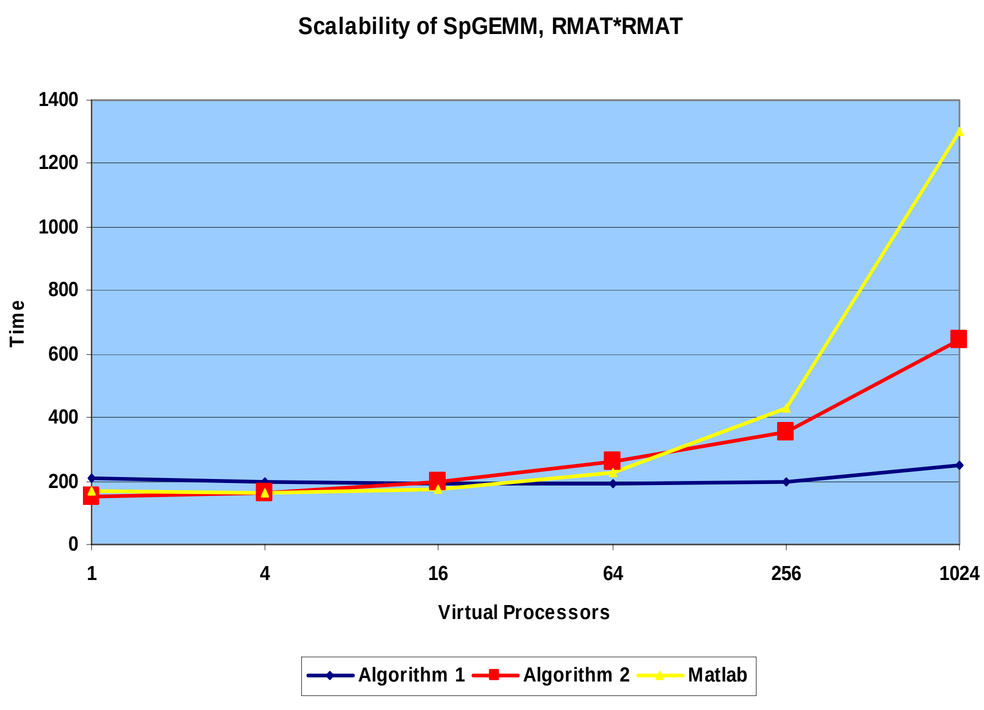
Sequential Kernel [B&G 2008]

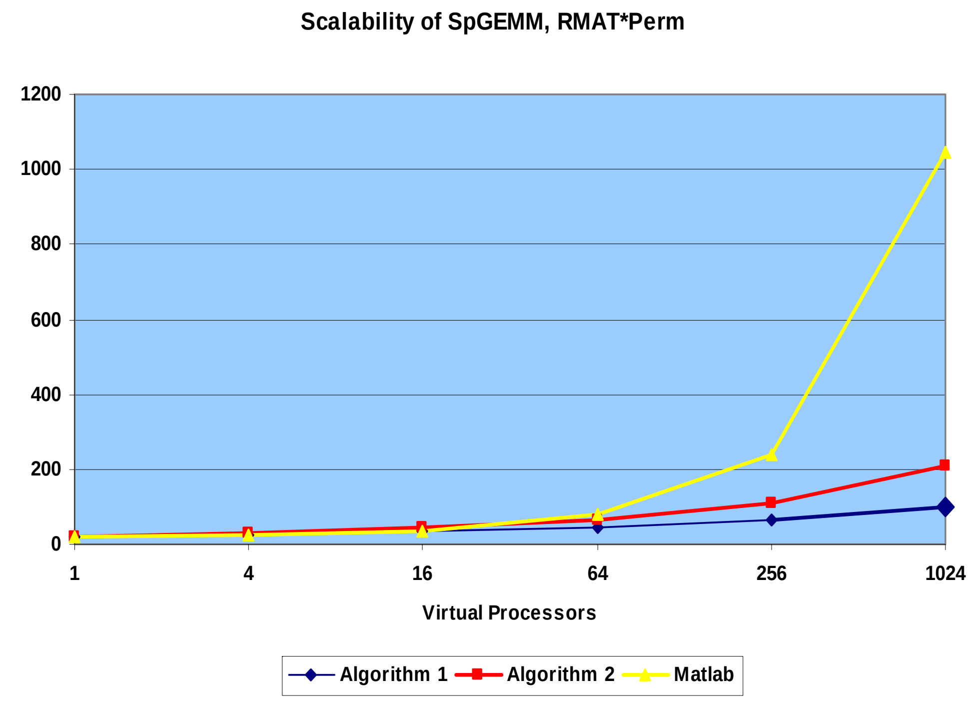
- Strictly *O(nnz)* data structure
- Complexity independent of matrix dimension
- Revival of outer-product formulation
- Heap assisted multi-way merging



UCSB

Experiments with RMAT





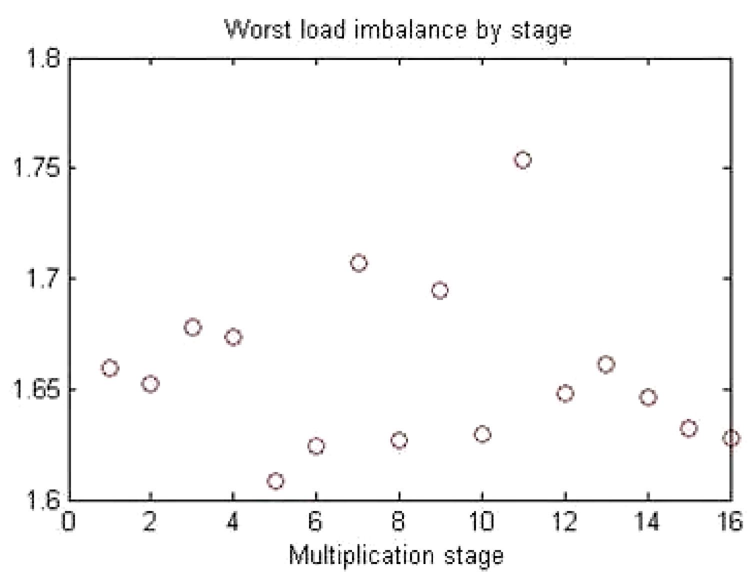
- Only submatrix multiplications are timed

$$\sum_{i=0}^{\sqrt{p}} \sum_{j=0}^{\sqrt{p}} \sum_{k=0}^{\sqrt{p}} time(A_{ik} \times B_{kj})$$

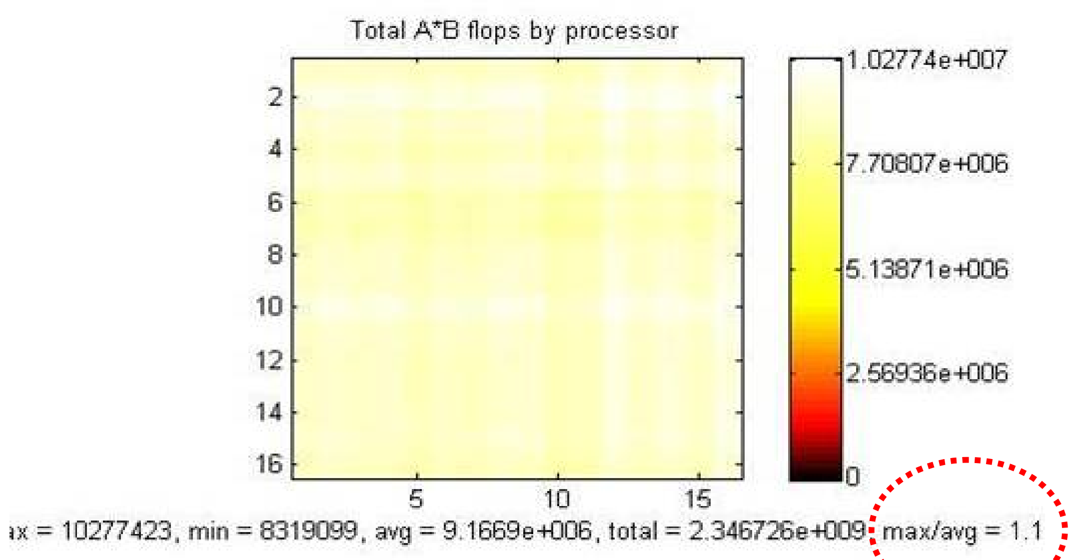
UCSB

Addressing the Load Balance

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



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



UCSB

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

UCSB

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.