Hypergraph Partitioning for Computing Matrix Powers

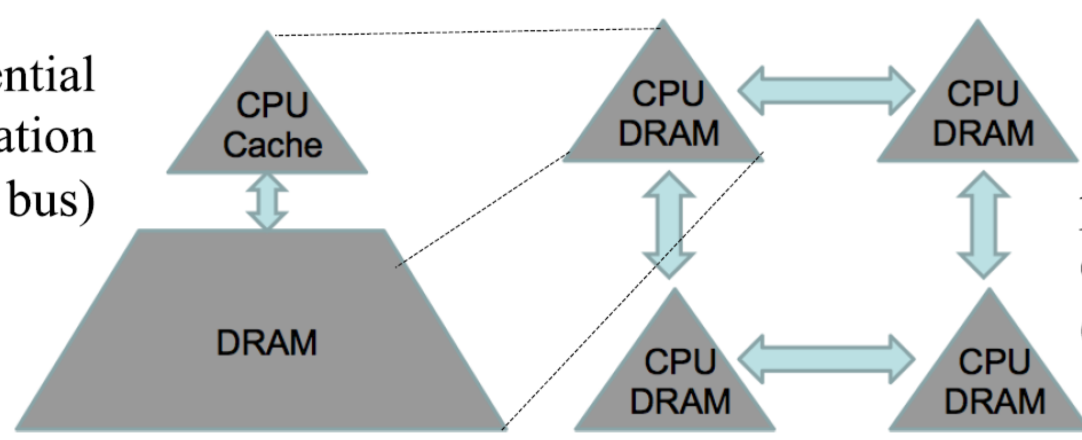
Nick Knight and Erin Carson

BERKELEY PARLAB

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from Par Lab affiliates National Instruments, NEC, Nokia, NVIDIA, and Samsung.

Communication is Expensive

sequential communication (eg, over a bus)

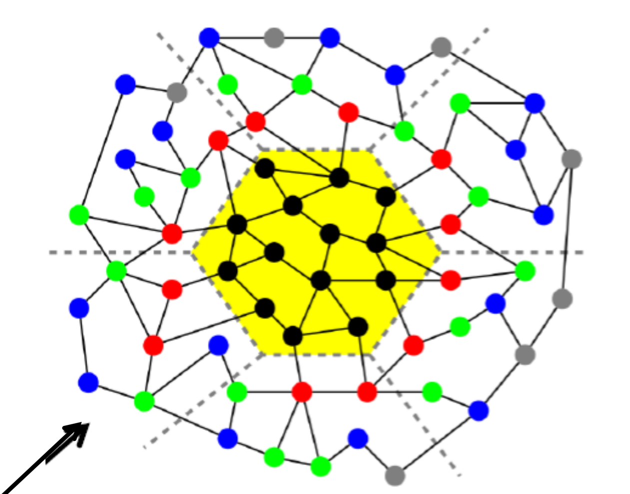


parallel communication (eg, over a network)

- Cost of an algorithm = computation + communication
- Time/flop << 1/bandwidth << latency – gap increasingly exponentially!
- Algorithms must *avoid communication* to improve efficiency

The Matrix Powers Kernel

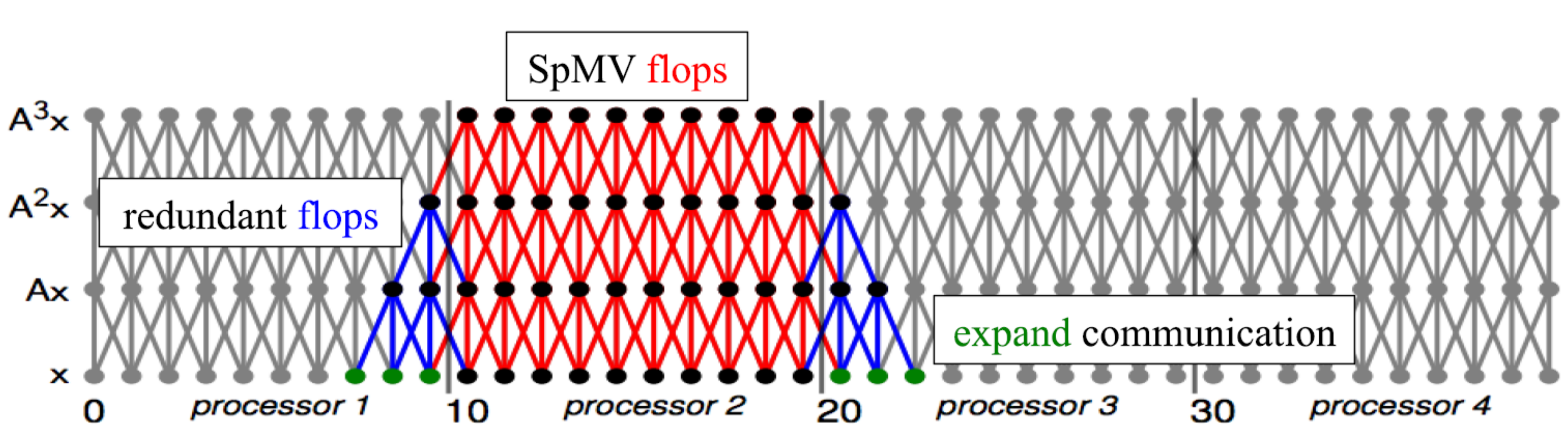
- Computes Asx(A, s, x) = [x, Ax, …, A^sx]
  - Only needs to read A once!
- Used to generate s Krylov basis vectors in Comm. Avoiding Krylov Subspace Methods
- In parallel, we avoid communication by doing s 'expand' phases upfront



Works for general graphs!

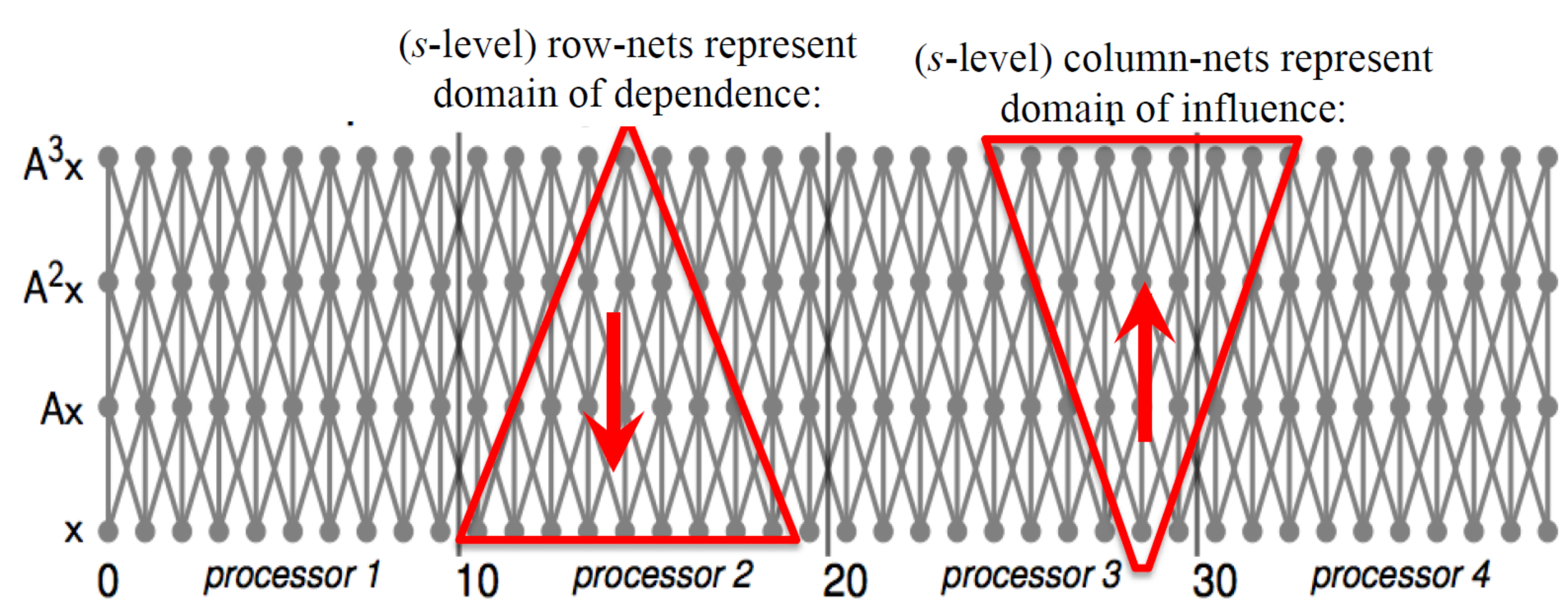
black = local elements
red = 1-level dependencies
green = 2-level dependencies
blue = 3-level dependencies

'Expand' communication upfront, no 'fold' communication.



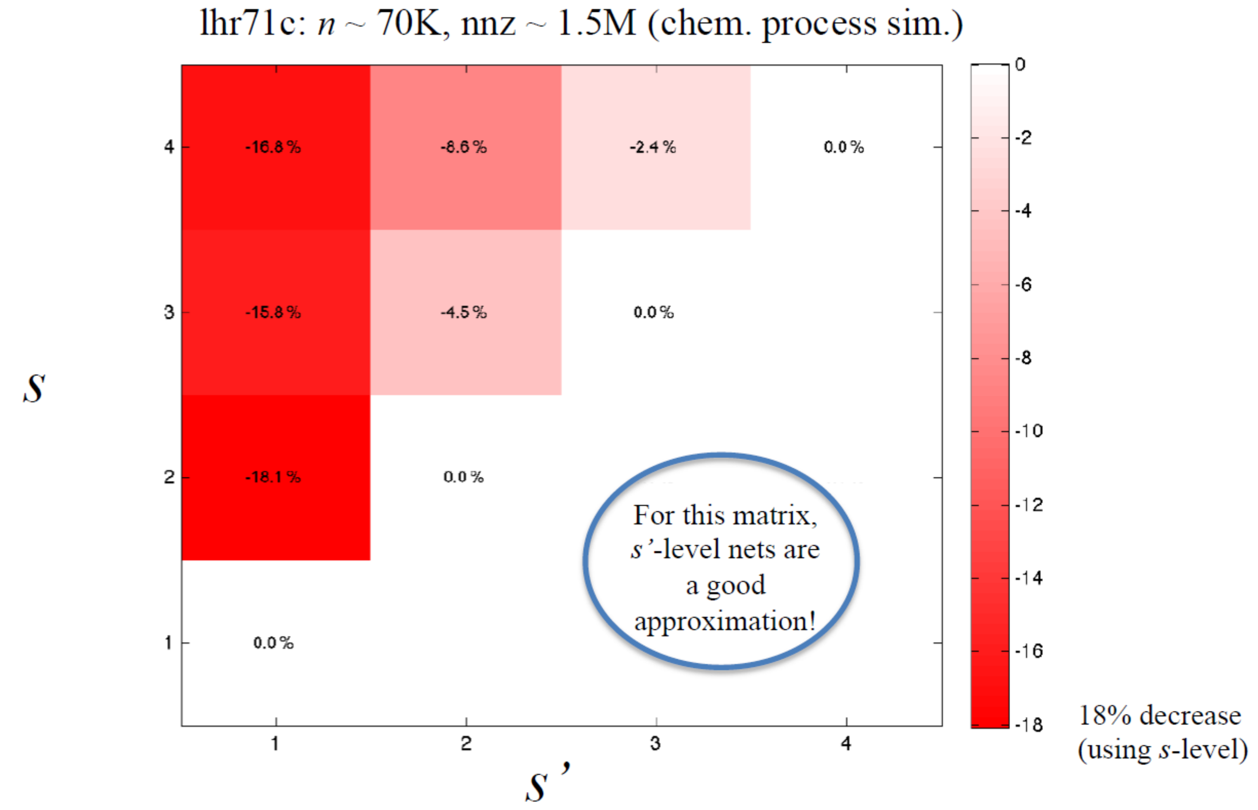
- Previous implementation: graph partitioning of A+A^T
  - Graph partitioning doesn't accurately count comm. [Catalyurek 99]
  - Poor approximation for unsymmetric matrices
  - Doesn't taken into account structure of A^s
- **Hypergraph partitioning** solves the first two problems…
- How can we extend the hypergraph model for SpMV to solve the third?

Modeling Communication in Matrix Powers



PROBLEM: Computing these s-level nets is expensive (s Boolean SpMVs!) Is an approximation good enough?

lhr71c: n ~ 70K, nnz ~ 1.5M (chem. process sim.)

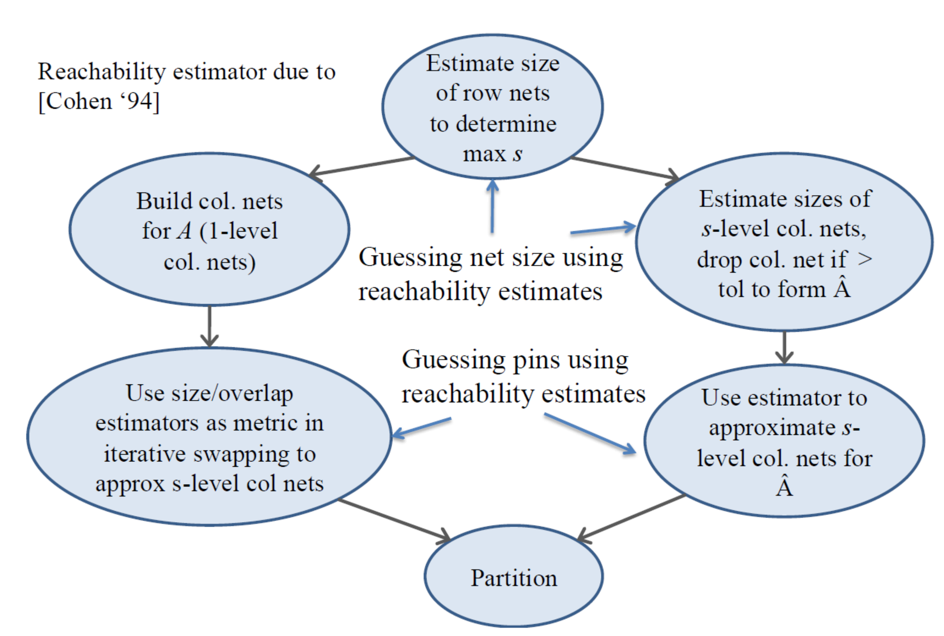


Heuristically Estimating Reachability

- Edith Cohen's Reachability Estimation ('94)
  - $O(n*nnz)$ time algorithm for estimating size of transitive closure (the size of each hyperedge in A^k)
    - Previous best was $O(n*sqrt(m))$ (Lipton and Naughton)
  - Adapted to compute col/row sizes in matrix product (for k repeated SpMVs, equivalent to k steps of the transitive closure)
- Motivation: DB-query size estimations, data mining, optimal matmul ordering, efficient memory allocation
  - **New motivation: Reducing hypergraph partitioning time (partitioning time and building the column nets) for computing matrix powers**

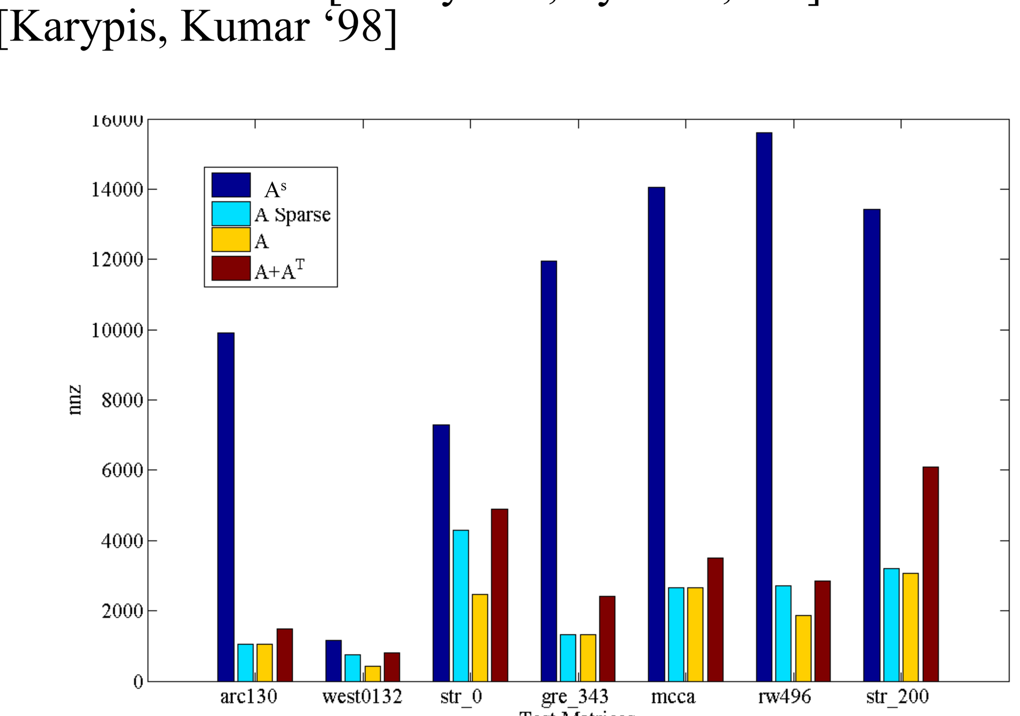
Algorithm Overview

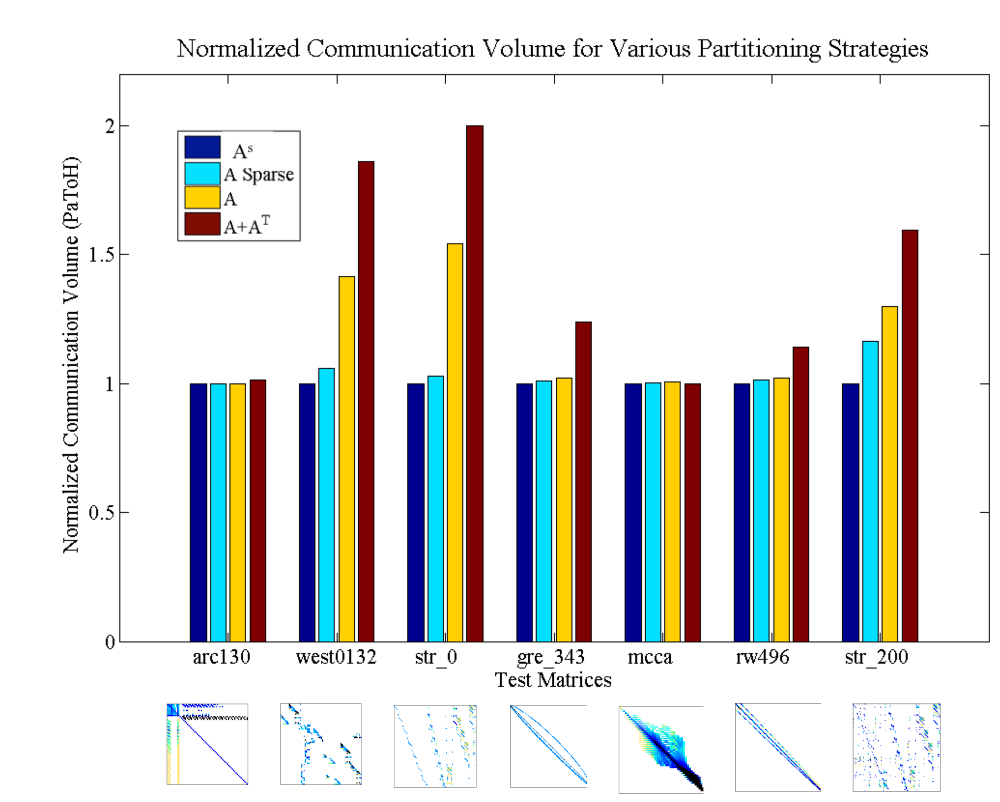
- Initially assign r-vector of rankings (sampled from exponential R.V., λ = 1) to each vertex v
- In each iteration (up to k), for each vertex v, take the coordinate-wise minima of the r-vectors reachable from v (denoted $S(v)$, non-zeros in column of A corresponding to v)



Preliminary Results

- Set of small test matrices from UFSMC [Davis '94]
- tol = 0.5 (half-dense), 4 parts, $s \in \{2, 3, 4\}$ depending on fill in A^s
- Comparison of hypergraph size and communication volume for four strategies:
  - s-level column nets
  - Sparsified column nets (somewhere between s- and 1-level)
  - 1-level column nets
  - Graph partitioning (A+A^T)
- Software: PaToH [Catalyurek, Aykanat, '99] and Metis [Karypis, Kumar '98]





Results and Observations

- Sparsified nets lead to comparable partition quality for **significantly** reduced hypergraph size
- Tuning parameter tol gives flexibility to trade off:
  - Quality of partition
  - Computation and storage costs