Hypergraph Partitioning for Computing Matrix Powers
Nick Knight and Erin Carson

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 (e.g., over a bus)
- Parallel communication (e.g., 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 $A^s(x, s, x) = [x, Ax, …, A^s x]$
- 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

Heuristically Estimating Reachability

- Edith Cohen’s Reachability Estimation (‘94)
  - $O(n^2 m)z$ time algorithm for estimating size of transitive closure (the size of each hyperedge in $A$)
  - Previous best was $O(n^2 p^2 (k^2))$ (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 $v$ vector of rankings (sampled from exponential $\lambda V$, $\lambda = 1$) to each vertex $v$
- In each iteration (up to $k$), for each vertex $v$, take the coordinate-wise minima of the $v$ vectors reachable from $v$ (denoted $R(v)$, 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 $A$
- 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