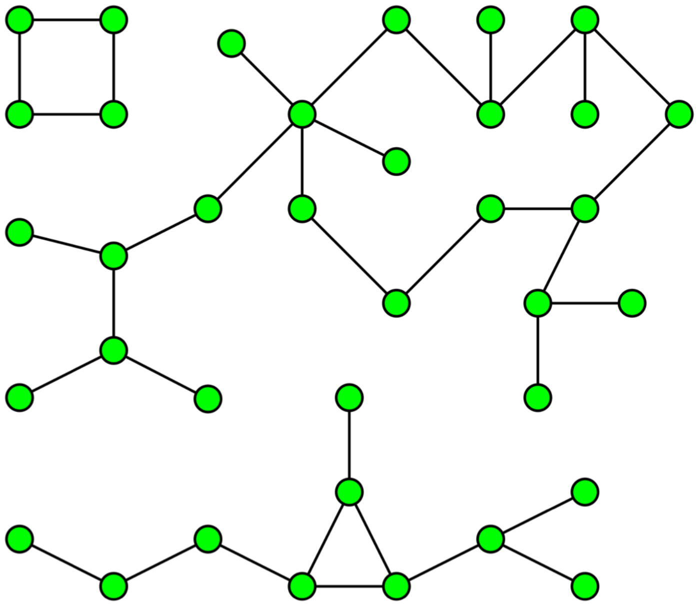
A New Parallel Algorithm for Connected Components in Dynamic Graphs

Tim Kralj

Outline

- Connected Components
- Motivation
- STINGER
- Data Structure
- Parent-Neighbor Relationship
- Initialization
- Results
- Discussion

Connected Component



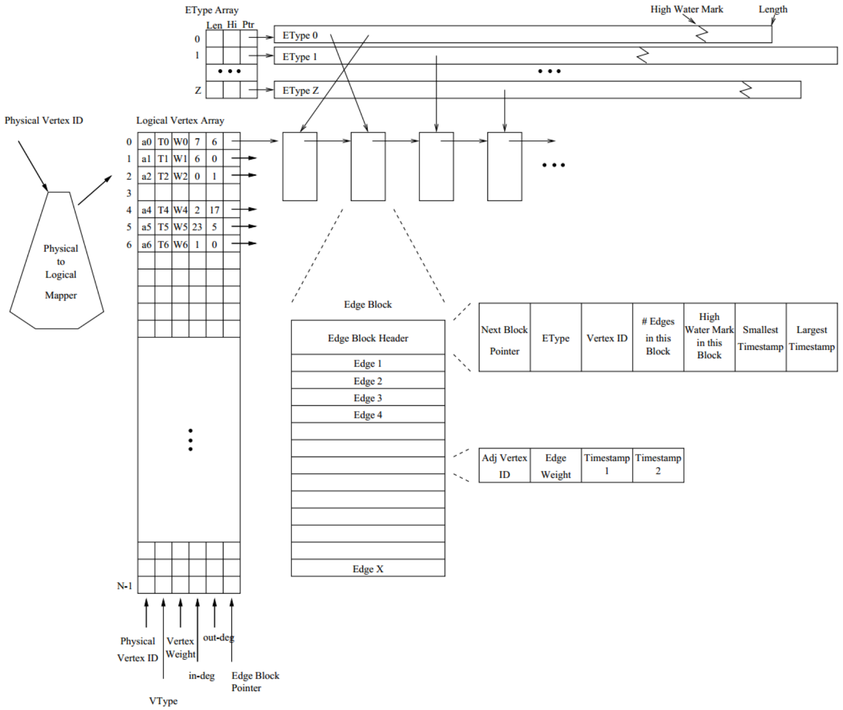
By David Eppstein - Own work, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=2838835

- Subgraph of original graph
- All vertices connected
- Usually calculated with BFS or DFS
- All vertices connected with paths

Uses of Connected Components

- Betweenness centrality
- Community detection
- Image processing
- Largely used in social networks

STINGER



- Data structure for dynamic graph problems
- Adjacency lists
    - Fast updates
- CSR
    - Low storage component
- Fast insertions
- Good locality
- Parallelism
- Streaming Graphs

Dynamic vs Static

- Dynamic Graph
  - Constantly updating
  - Steam of updates
  - Analysis of continuously changing state
- Static
  - Snapshot of dynamic graph
  - Analysis of a state
  - How most algorithms taught (006 etc)

Parent-Neighbor Relationship

- Memory Requirement of O(V)
- Directed subgraph
- Each vertex has list of vertices above “parents”
- List of vertices at same level “neighbors”
- Limit number of parents and neighbors with threshold
  - Helps maintain O(V) instead of O(V + E)

Data Structure

- Maintained in real time
- Initialization step

Name	Description	Type	Size (Elements)
C	Component labels	array	$O(V)$
$Size$	Component sizes	array	$O(V)$
$Level$	Approximate distance from the root	array	$O(V)$
PN	Parents and neighbors of each vertex	array of arrays	$O(V \cdot thresh_{PN}) = O(V)$
$Count$	Counts of parents and neighbors	array	$O(V)$
$thresh_{PN}$	Maximum count of parents and neighbors for a given vertex	value	$O(1)$
$\tilde{E}_I$	Batch of edges to be inserted into graph	array	$O(batch\ size)$
$\tilde{E}_R$	Batch of edges to be deleted from graph	array	$O(batch\ size)$

Initialization

- Uses Parallel BFS
- Start with
  - Level at INF
  - Counter = 0
  - Component size = 0
- Dequeue vertex and add parents and neighbor relationships
  - Only add neighbors if below Threshold
  - All parents will be found before neighbors added
- Atomic compare-and-swap, fetch-and-add

Updates

- Insertions-simple
- Deletions-harder

Insertions

- 2 options:
  - Edge within a connected component
  - Edge joins two components
- Levels of s and d checked
- Update Data structure

Deletions

- Need to check parents of deleted edge
- Search for remaining parent
  - If remains, connection to the root of component must exist-safe
  - If no parents, check for neighbors if levels > 0 (meaning path to the root)
  - If neither is true, deletion was not safe
- Need to check from both s and d of <s,d> deletion

Results

- Varying threshold variable
  - Used for number of parents/neighbors maintianed

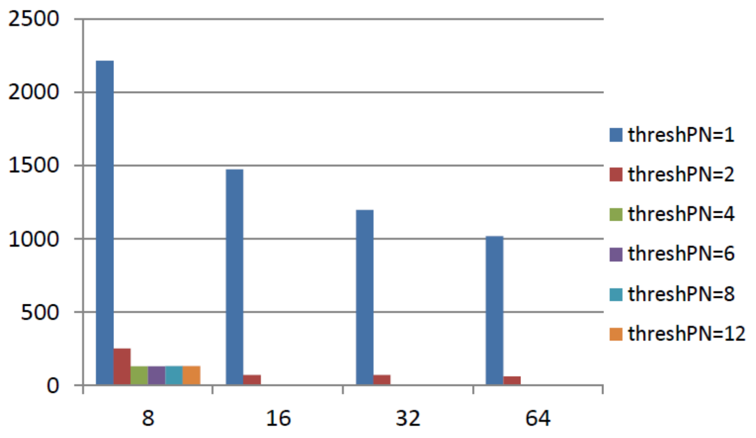


Figure 2. Average number of unsafe deletes in PN data structure for batches of $100K$ updates as a function of the average degree (x-axis) and $thresh_{PN}$ (bars).

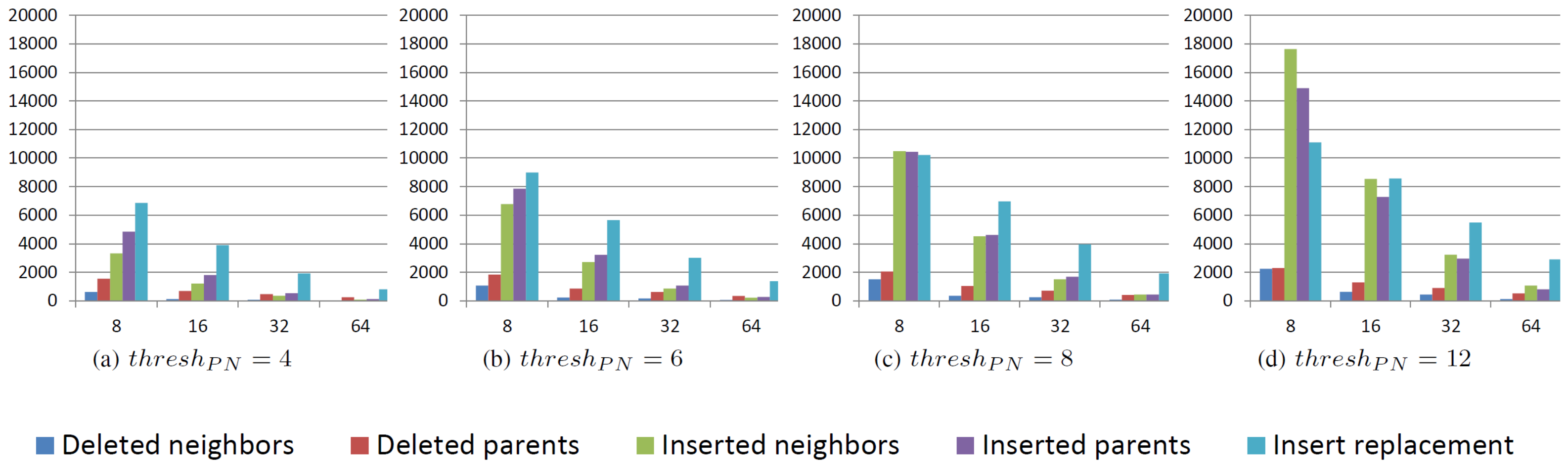


Figure 1. Average number of inserts and deletes in PN array for batches of $100K$ updates for RMAT-22 graphs. The subfigures are for different values of $thresh_{PN}$. Note that the ordinate is dependent on the specific bar chart. The charts for RMAT-21 graphs had very similar structure and have been removed for the sake of brevity.

Results

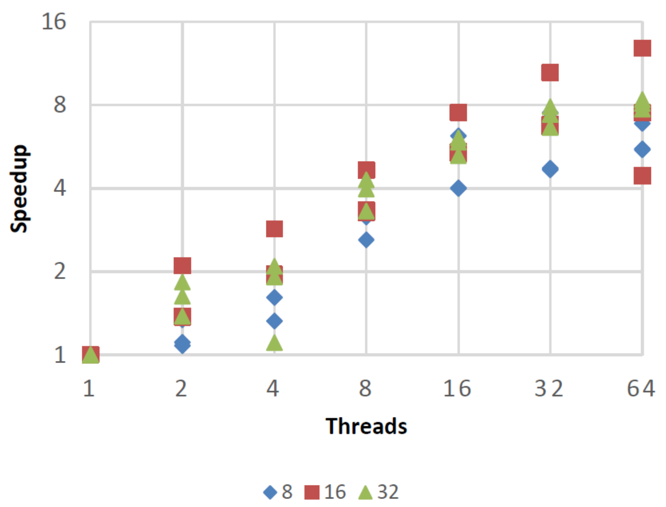


Figure 4. Speed up of the new algorithm over performing parallel static recomputation after each batch on three different RMAT-22 graphs with each average degree as a function of the number of threads.

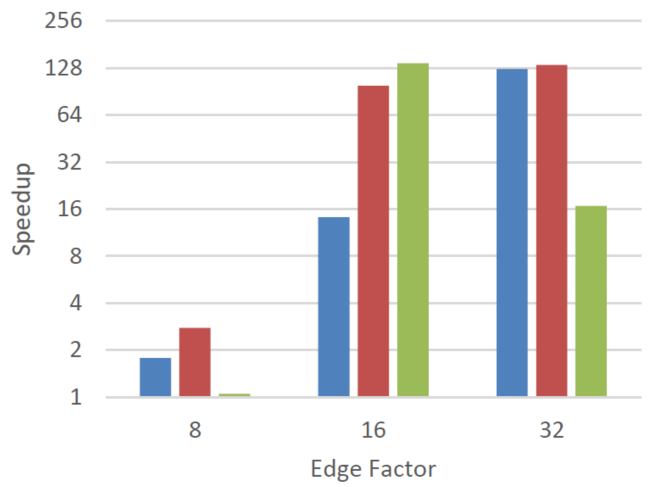


Figure 6. Speed up over performing static recomputation after each batch on scale 24 graphs for three graphs at each edge factor using 64 threads.

Discussion Questions/Commentary

- Mentions of increasing core counts but none of distributed systems, would this be feasible and improve performance?
- Use BFS in the initial stage and during steaming, be better if switched to a faster algorithm
- Largest graph had 2^{24} vertices (16 million) but social graphs much larger
- First parallel computing of connected components with processor oblivious runtime