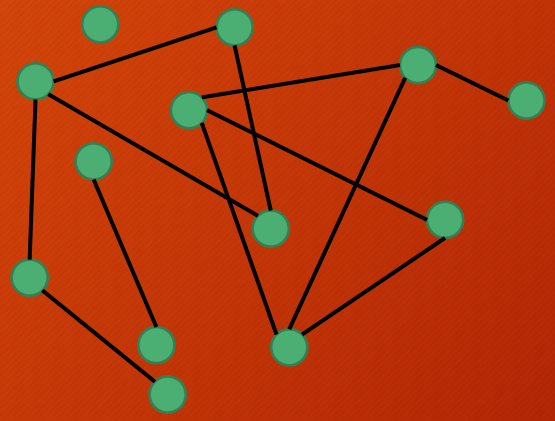
Parallel Connected Components

The Parallelepipeds:
Fabian Meier
Gustavo Segovia
Seraiah Walter

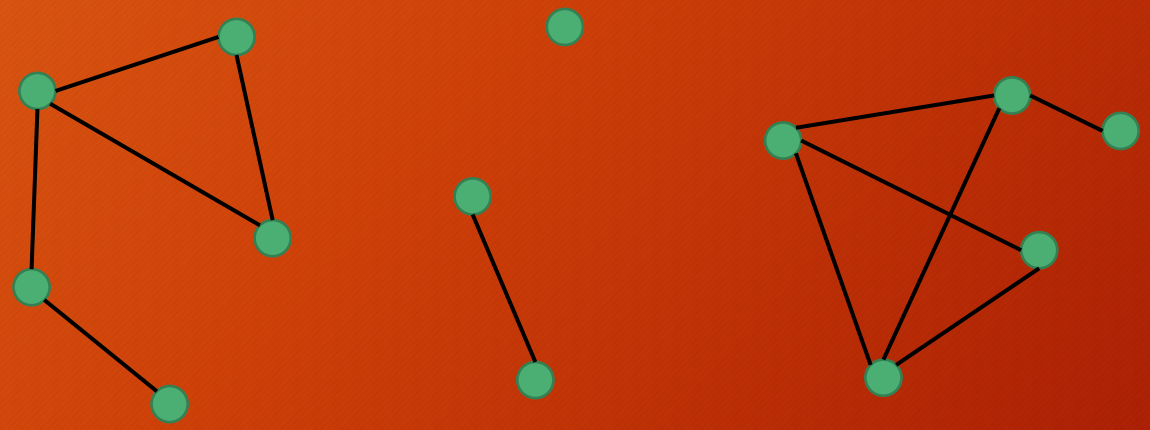
The problem

- Connected components on an undirected static graph
- Whole graph fits in memory



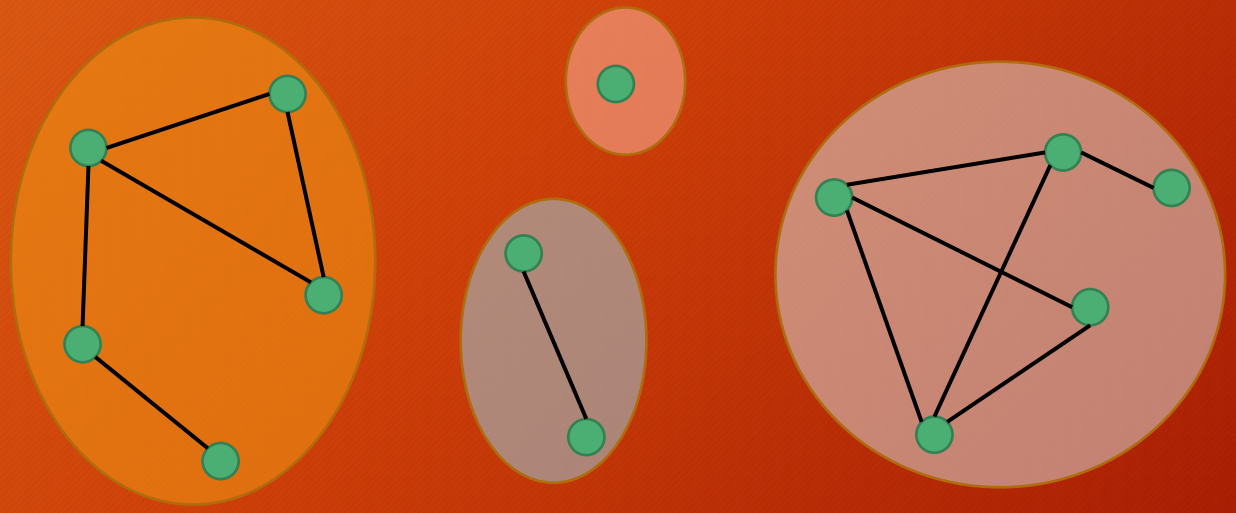
The problem

- Connected components on an undirected static graph
- Whole graph fits in memory



The problem

- Connected components on an undirected static graph
- Whole graph fits in memory



Serial approaches

- BFS/DFS approach — O(n+m)
  - Iterate through every vertex
  - If vertex has no assigned component id
    - Create new component id
    - Search vertex's connected graph with BFS or DFS
    - Assign every vertex found with component id
- Union-Find approach — O(m log(n))
  - Maintain a Union-Find data-structure
  - Iterate over all edges

Related work

- Parallel 1-D block mapping
  - http://www.cs.rice.edu/~vs3/comp422/lecture-notes/comp422-lec24-s08-v2.pdf
  - Splits adjacency matrix into n blocks, processes in parallel and merges in the end
- Randomized and Deterministic Parallel Connected Components
  - http://www3.cs.stonybrook.edu/~rezaul/Spring-2012/CSE613/CSE613-lecture-11.pdf
  - Contracts graphs into smaller graphs at every iteration
- Boost Parallel Connected Components implementation
  - http://www.boost.org/doc/libs/1_56_0/libs/graph_parallel/doc/html/connected_components.html
  - Same contraction idea

The grand plan

- Implement a graph generator
  - Number of components
  - Component size
  - Mean vertex connectivity
- Implement several Connected components algorithms
  - Basic serial implementation
  - From related work
  - Our own ideas using OpenMP and MPI
- Compare them all with each other and with Boost's serial and parallel implementation

MPI

- BFS approach
  - Random starting points
  - Update other processes in intervals of k steps
  - Use non-blocking messages
- Union-Find approach
  - Split edges evenly
  - Find components of each part
  - Merge the result
  - Use blocking messages

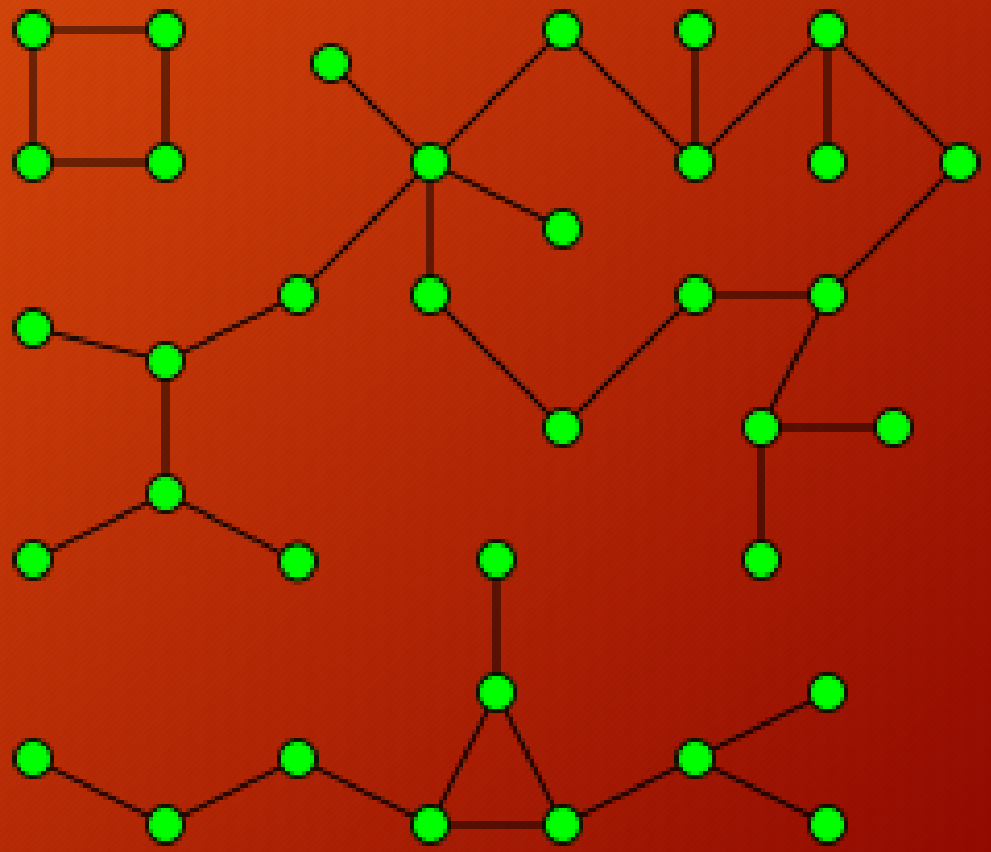
MPI Discussion

- Messaging rounds could be a bottleneck
- No shared memory will decrease runtime

- Merge operation with union find
- Expected Runtime: $O(m/p + n \log(p))$

OpenMP

- Take advantage of Shared-Memory
- First approach:
  - each Core starts at a different Node BFS
  - mark visited nodes to prevent duplicate work.
- Second approach
  - Union find



OpenMP Discussion

- How to make sure each cache has the same data?
  - Don’t! (Hardware will take care of it eventually)
    - Might lead to duplicate work.
  - Use atomic operations
    - Might be slow.
  - Use Locks
- Which one is fastest?