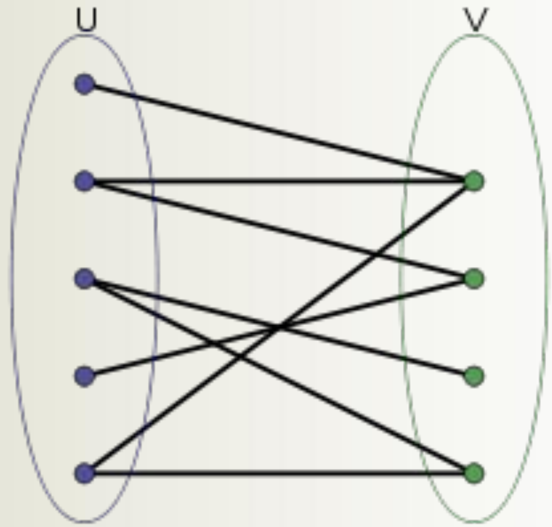
# Distributed Bipartite Matching

Brian A. Page
bpage1nd.edu
November 8, 2018

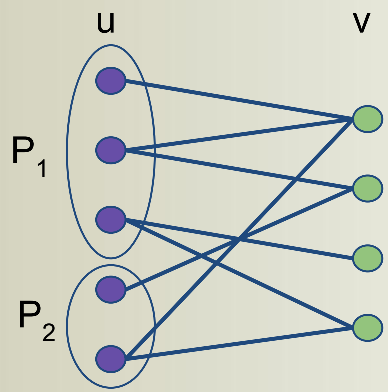
The College of Engineering at the University of Notre Dame

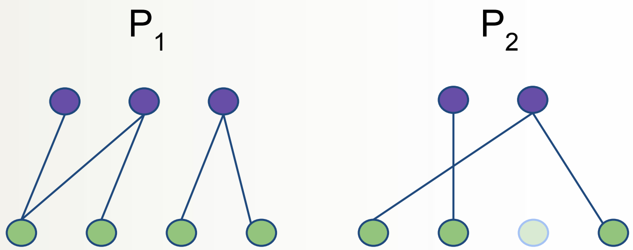
# Bipartite Matching




- Matching (M) is set of edges such that E(u,v)
- Vertices incident to only one edge in M

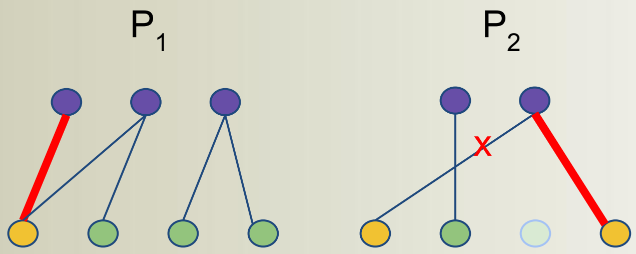
# DMBM: Kernel Cont.

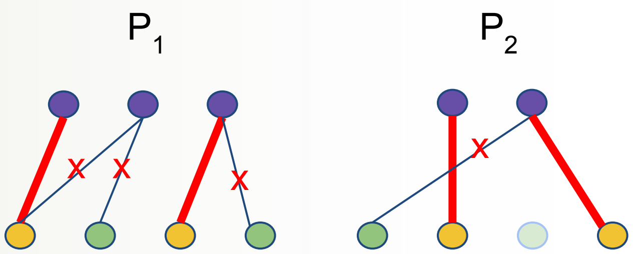



- Partition Graph based on one set of vertices
- Distribute vertices and associated edge lists to processes
- Better partitioning can be created but at greater cost

# DMBM: Kernel




|M| = 4

- Updates must be when vertex is matched
- Similar to Push-Relabel
- Lots of communication required

# DMBM: Kernel Cont.

Given a graph $G(V(u,v), E(u_i,v_j))$:

```
bool augment_path(uint uid) {
    visited[uid] = true;

    for (uint i = 0; i < graph[uid].size(); i++) {
        uint neighbour = graph[uid][i];
        if (visited[neighbour]) {
            continue;
        }

        // Base-case. We've reached a node at the end of an alternating path that
        // ends in a freenode.
        if (matched[neighbour] == UNMATCHED) {
            matched[uid] = neighbour;
            matched[neighbour] = uid;
            return true;
        } else if (matched[neighbour] != uid) {
            // This is not your standard DFS. Because we're DFSing along an
            // alternating path, when we choose the next vertex to visit, we MUST
            // then go along its matching edge. So we say we've visited the neighbour
            // trivially and then recursing on matched[neighbour].

            visited[neighbour] = true;
            if (augment_path(matched[neighbour])) {
                matched[uid] = neighbour;
                matched[neighbour] = uid;
                return true;
            }
        }
    }

    return false;
}
```

# DMBM: Kernel Cont.

Given a graph $G(V(u,v), E(u_i,v_j))$, and process count P:

distribute vertex and edge list assignments

omp for $i < u_p$.size() do

    augmentPath($u_i$)

        if $E(v_j)$ contains $u_i$ st $u_i \notin u_p$

            notify $P_k$ assigned $u_i$ of $v_j$ visitation

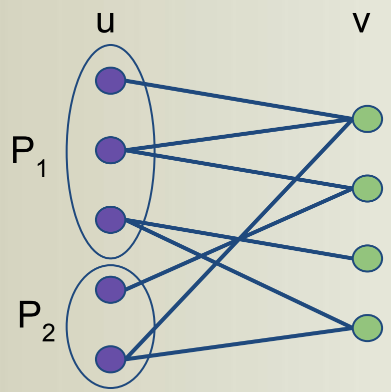
    if notified ($v_j$)

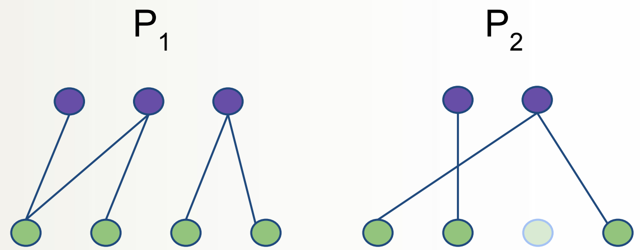
        augmentPath($v_j$)

gather matchings

end

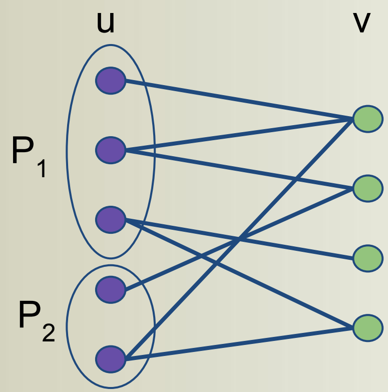
# Problem 1: Partitioning

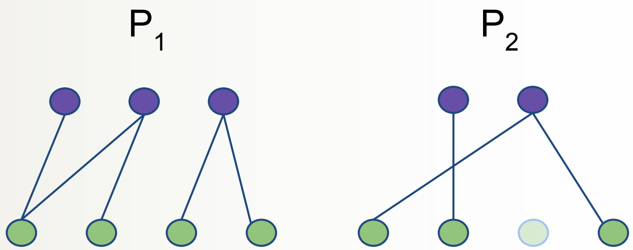



- If Scale Free graphs are used, the partitioning can become highly skewed
- Imbalance causes communication hotspots
- Concerned with vertex count AND edge count

# Problem 2: Communication




- Communication at core of compute phase
- Message volume and interconnect becomes dominant factor
- Does NOT scale well!!

# DMBM: Time Complexity

- Ford-Fulkerson: $O(VE^2)$
- Hopcroft-Karp: $O(|E|\sqrt{(V)})$
- Distributed MBM: $O(O(VE^2)+V\lg(P))$
  - not 100% on this...

# DMBM: Data Sets

Suite Sparse Matrix Collection

https://sparse.tamu.edu

Largest undirected biparite graph:

- 12,471 x 872,622 (885,093 total vertices)
- 22,624,727 edges

# DMBM: Present and Future

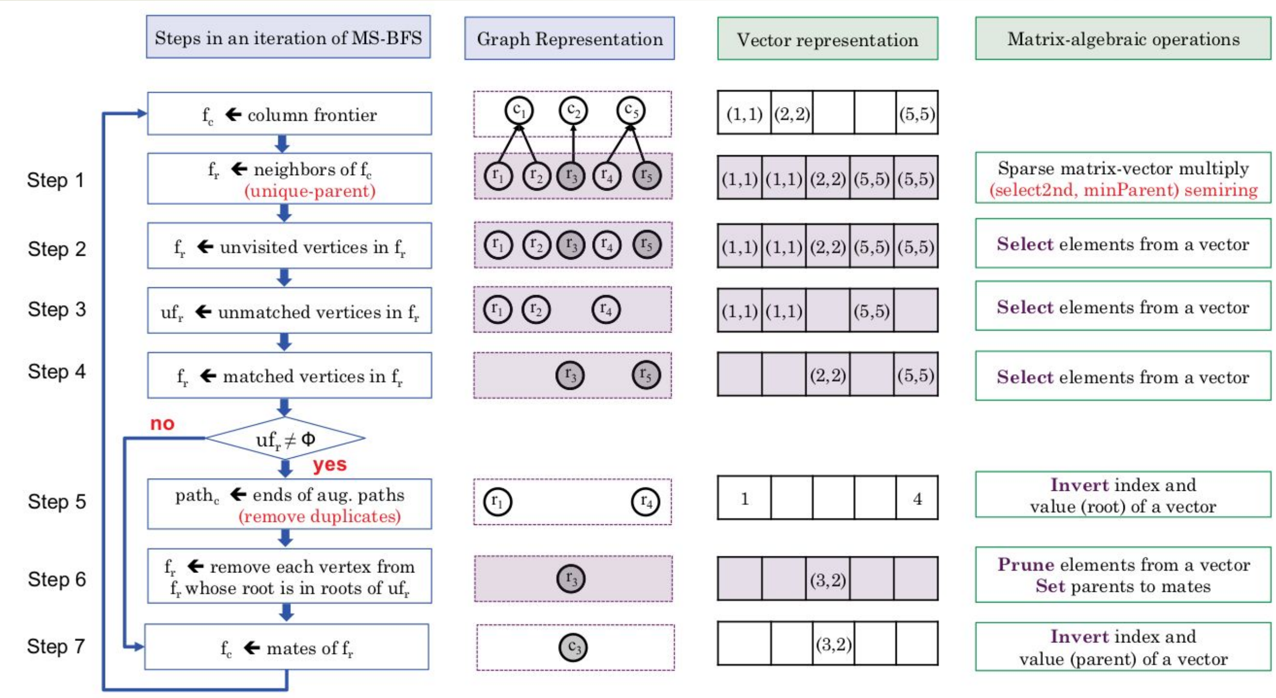
Presently:

- It works!
- Performance is abysmal

Future:

- Implement Hopcroft-Karp to see if communication is reduced
- HavoqGT vertex-centric framework
- Communication may be unavoidable

# DMBM: Ray of Hope




Ariful Azad, Aydin Buluc (LBNL)

- Sparse algebra based Distributed MCM
- SpMV plays significant role

# DMBM: Ray of Hope Cont.

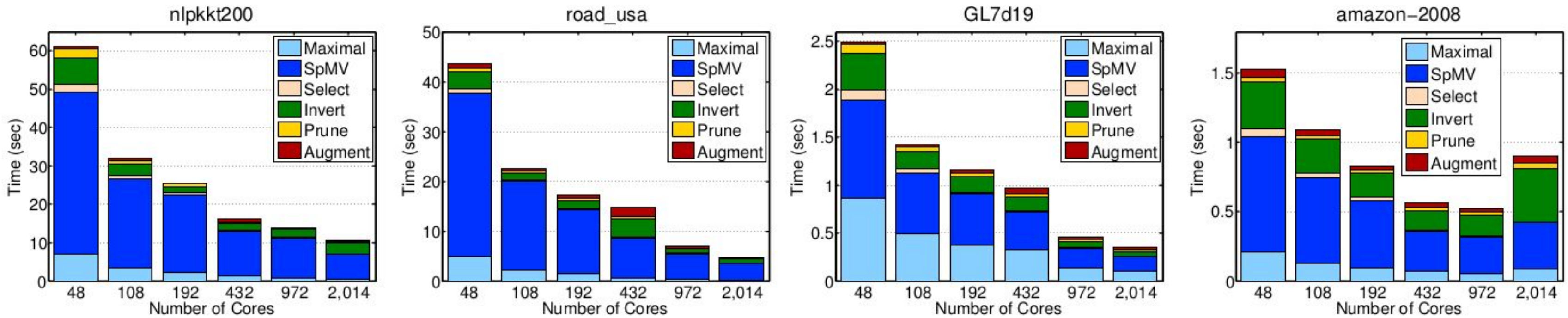



Fig. 5: Runtime breakdown of MCM-DIST for four representative graphs using 12 threads per MPI process on Edison.

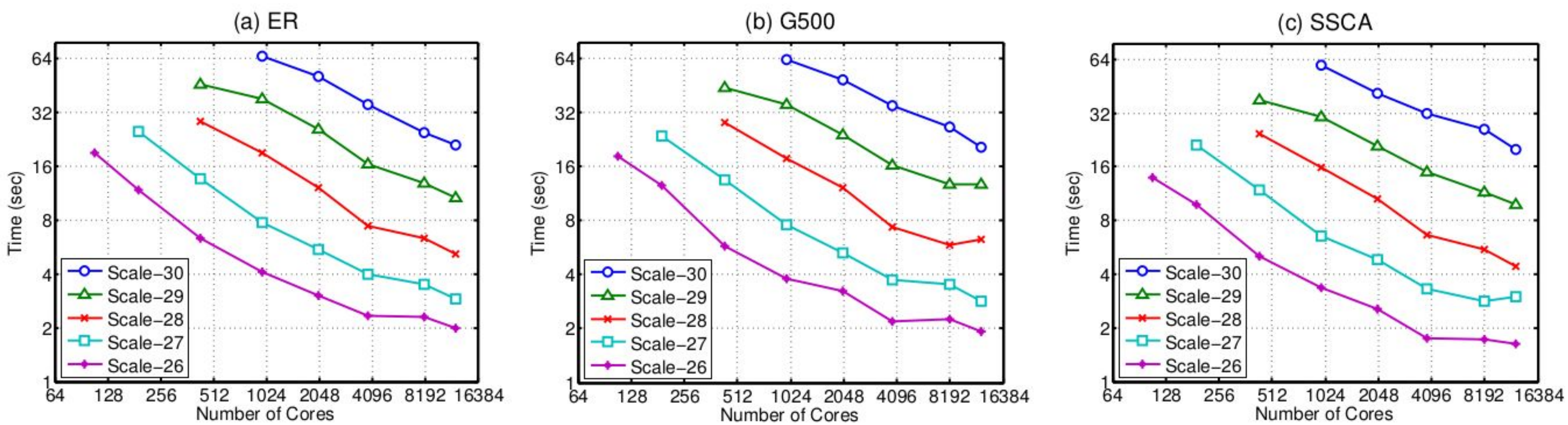



Fig. 6: Strong scaling of MCM-DIST when computing maximum matching on three classes of randomly generated graphs with five different scales on Edison.