PARALLEL APPROACH TO DIJKSTRA'S ALGORITHM

LIBING WU

PROFESSOR: RUSS MILLER

SPRING 2017, CSE 633, UNIVERSITY AT BUFFALO
Dijkstra's algorithm to find the shortest path between a and b. It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbor, and updates the neighbor's distance if smaller. Mark visited when done with neighbors.
Dijkstra's algorithm to find the shortest path between a and b. It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbor, and updates the neighbor's distance if smaller. Mark visited when done with neighbors.

Sequential running time of...

- Dijkstra's algorithm with list $O(V^2)$
- Dijkstra's algorithm with modified binary heap $O((E + V) \log V)$
- Dijkstra's algorithm with Fibonacci heap $O(E + V \log V)$
function Dijkstra(Graph, source):
    create vertex set Q

    for each vertex v in Graph:          // Initialization
        dist[v] ← INFINITY             // Unknown distance from source to v
        prev[v] ← UNDEFINED           // Previous node in optimal path from source
        add v to Q                     // All nodes initially in Q (unvisited nodes)

    dist[source] ← 0                      // Distance from source to source

    while Q is not empty:
        u ← vertex in Q with min dist[u] // Node with the least distance will be selected first
        remove u from Q

        for each neighbor v of u:         // where v is still in Q.
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:           // A shorter path to v has been found
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
DRAWBACK

- Sequential running time of...
- Dijkstra's algorithm with list \( O(V^2) \)
- Dijkstra's algorithm with modified binary heap \( O((E + V) \log V) \)
- Dijkstra's algorithm with Fibonacci heap \( O(E + V \log V) \)
- If nodes are too many (scale is too large), it will become very inefficient.
A PARALLEL APPROACH

• Divide nodes into clusters
• Each cluster calculate local node closest to the starting node
• Identify global closest node using parallel prefix
• Broadcast to all cluster
• Perform list update for each cluster
VOID DIJKSTRA(INT *LOCAL_MAT, INT *LOCAL_DIST, INT *LOCAL_PRED, INT N, INT LOCAL_N, INT MY_RANK, MPI_COMM COMM)

// INITIAL LOCAL MINIMUM DISTANCE

//SET NODE 0 WITH VISITED MARK

// FIND A LOCAL NODE WHICH IS NOT VISITED WITH Minimum DISTANCE

/*

* SEND:

* MIN_DIST_LOC[0]: THE Minimum DISTANCE TO A LOCAL NODE

* MIN_DIST_LOC[1]: THE GLOBAL POSITION OF THE LOCAL NODE

* RECEIVE:

* MIN_DIST[0]: THE Minimum DISTANCE TO A NODE

* MIN_DIST[1]: THE GLOBAL POSITION OF THE NODE

*/

//COMBINES VALUES FROM ALL PROCESSES AND DISTRIBUTES THE RESULT BACK TO ALL PROCESSES

MPI_ALLREDUCE(MIN_DIST_LOC, MIN_DIST, 1, MPI_2INT, MPI_MINLOC, COMM);

//CALCULATE NEW Minimum DISTANCE
RUNNING TIME

- \( P \) = number of processor
- During each iteration, update value of \( O(V) \) nodes with \( P \) processor \( \rightarrow O(V/P) \)
- During each iteration, find global closest node \( \rightarrow O(\log P) \)
- \( O(V) \) iteration to finish routing table
- Running time: \( O \left( \frac{V^2}{P} + V \log P \right) \)
- More efficient to handle large scale nodes.
SEQUENTIAL VS. PARALLEL

time (sec)

number of nodes

❓❓❓

parallel
sequential
SEQUENTIAL VS. PARALLEL (LARGER SCALE...)

![Graph showing time (sec) vs. number of nodes for parallel and sequential processing.](image)
SEQUENTIAL VS. PARALLEL (LARGER SCALE...)
2 PROCESSORS VS. 4 PROCESSORS

Time (sec) vs. Number of Nodes

- Sec(2 cores)
- Sec(4 cores)

MinDist:
- 0: 18
- 1: 15
- 2: 7
- 3: 3
- 4: 3
- 5: 2
- 6: 12
- 7: 6
- 8: 16
- 9: 0
- 10: 2

Time cost: 1012067 usec
2 PROCESSORS VS. 4 PROCESSORS (FIXED BY INCREASING WEIGHTS ON EDGES)
NUMBER OF PROCESSORS (2, 4, 8, 16)

time (sec)

number of nodes
SCENARIO OF 32 PROCESSORS...
(NOT PERFORM BETTER THAN 16 PROCESSORS)
PERFORM BEST WITH 16 PROCESSORS
SPEED-UP

percentage of parallel faster than sequential

number of nodes
• **LIMITATION:** In Dijkstra’s algorithm, only the inner loop can be implemented with parallel algorithm.

• **FINDING:** When number of processors is increasing, the efficiency of parallel algorithm drops, cost of communication increases, so that it slows down the whole program’s speed.

• **FURTHER:** More nodes... formula to calculate best amount of processors
THANKS