Parallelization and analysis of all-shortest paths algorithms using OpenMP

University of Nebraska Lincoln
Computer Science & Engineering

Introduction

Applications

- Route guidance
  - Determining shortest routes dependent on a variety of weights. Constantly needs updated
- Network Router Pathing
  - Determining shortest route for network to take in order to deliver packets. Unoptimized routes increase network latency. Important for time critical applications
- Video Game AI path finding
  - Artificial Intelligence players need to react to known human players and possibly determine shortest path from a constantly moving object.

Two All-shortest paths algorithms:
Floyd-Warshall and Dijkstra

Implementation

Floyd Warshall's Algorithm

- Runs through all possible paths from one vertex to another to find shortest
- Outer k loop is not parallelizable because k determines the intermediate vertices in order

Analysis:
Serial: $\Theta(n^3)$
OpenMP: $\Theta(n^3/p)$

Implementation of OpenMP algorithm

```
for (k = 0; k < n; ++k)
    #pragma omp parallel for private(i,j)
    for (i = 0; i < n; ++i){
    nt = omp_get_num_threads();
        for (j = 0; j < n; ++j)
            if ((dist[i][k] * dist[k][j] != 0) && (i != j))
                if ((dist[i][k] + dist[k][j] < dist[i][j]) || (dist[i][j] == 0))
                    dist[i][j] = dist[i][k] + dist[k][j];
    }
```

Implementation of Serial algorithm

```
for(int k = 0; k < n; k++)
    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            if(dist[i][j]>dist[i][k]+dist[k][j])
                dist[i][j]=dist[i][k]+dist[k][j];
```

Data Sets:
Input:
Randomly generated edge weights from 0-50 for V vertices
Output:
Table of shortest paths for each pair

Dijkstra's Algorithm

- Determines the distance from a source vertex to each other vertex
- Checks to make sure a vertex isn't visited twice and finds the shortest path by checking if the distance to next thorough v from u is shorter than the previous shortest

Analysis:
Serial: $\Theta(E^*V^2)$ -- Using adjacency matrix instead of list means E=V
OpenMP: $\Theta((E^*V^2)/p)$

Implementation of Serial algorithm

```
dist[src] = 0;

for (int count = 0; count < V-1; count++)
{
  int u = minDistance(dist, sptSet);

  sptSet[u] = true;

  for (int v = 0; v < V; v++)

    if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                                  && dist[u]+graph[u][v] < dist[v])
      dist[v] = dist[u] + graph[u][v];
}
```

Implementation of OpenMP algorithm

```
#pragma omp parallel for
for(i=0; i < V; i++){
    dijkstra(i);
    x = omp_get_num_threads();
}
```

Analysis and Evaluation

- Both of the OpenMP algorithms result in a reduced running time for more than 1 thread.
- Floyd Warshall's algorithm shows excellent strong scaling average 50% reduction in run time
- Dijkstra's algorithm shows poor strong scaling with an average 35% reduction in run time
- Both show the same weak scaling performance

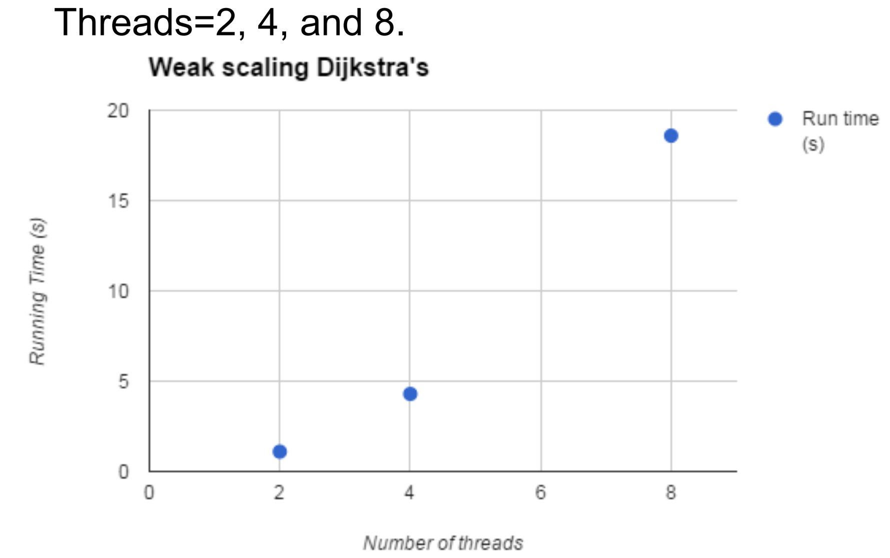
Seq. running times:
500 Vertices
0.356 s FW
0.353 s DS

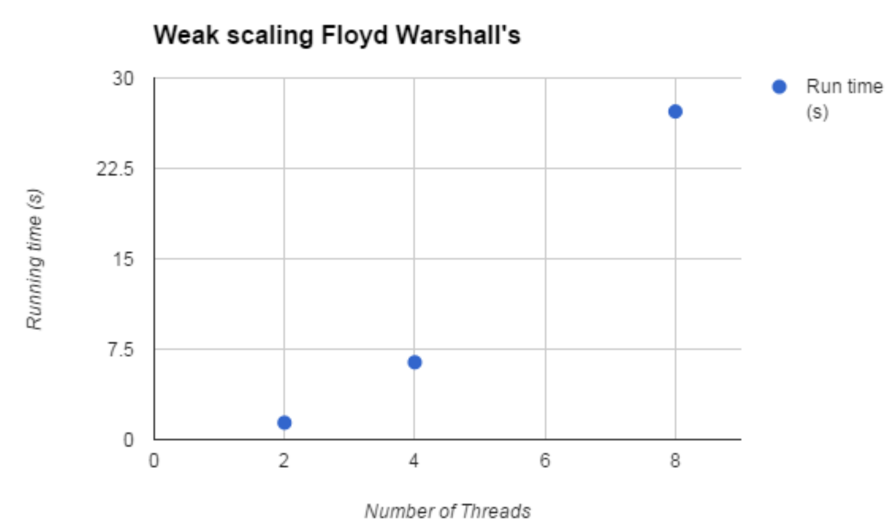
Speed ups for each algorithm dependent on number of threads (Problem size at 500 vertices):
Floyd Warshall's
- 2 threads result in a 50% reduction in running time (1.7)
- 4 threads result in a 50% reduction in running time (3.5)
- 8 threads result in a 51% reduction in running time (6.9)

Dijkstra's
- 2 threads result in a 40% reduction in running time (1.3)
- 4 threads result in a 37% reduction in running time (3.3)
- 8 threads result in a 28% reduction in running time (4.5)

Weak Scaling

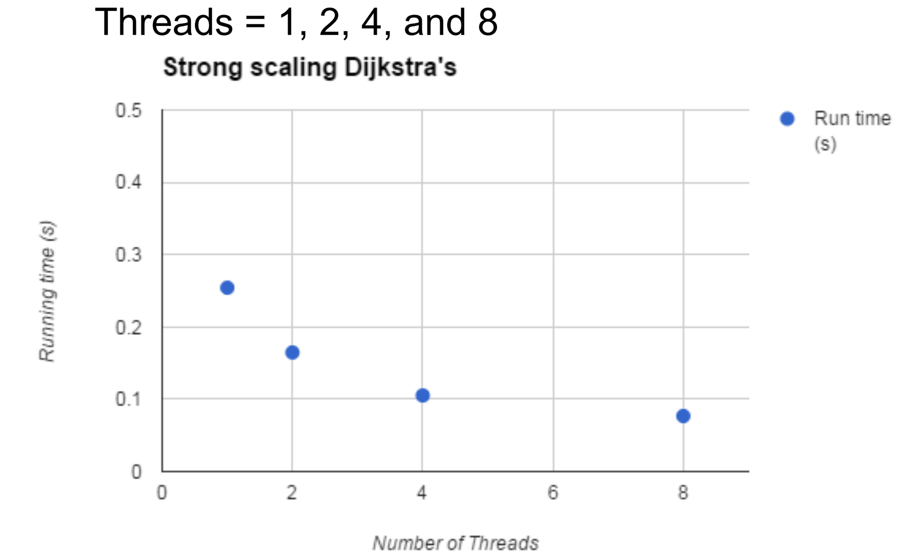
Weak scaling shows the effect of increasing both the problem size and the number of threads at the same time. Variables used where V=1000, 2000, and 4000 and Threads=2, 4, and 8.

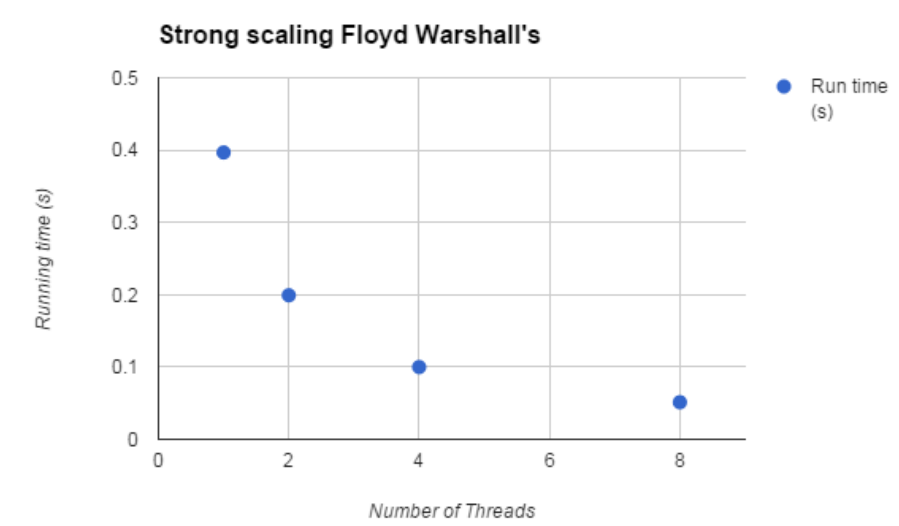




Strong Scaling

Strong scaling shows the effect of only increasing the number of threads. Variables used where V=500 and Threads = 1, 2, 4, and 8





Future Work and Conclusion

MPI and PGAS versions

Blocking of both algorithms (parallelize inner working instead of just splitting based on vertices)

Additional Shortest path algorithms

Overall:
Both algorithms provided great speedups compared to the sequential versions. Floyd Warshall provides best speedup with completely dense graph.

Zachary Sims

CSCE 456