Parallelization of A*: A Graph Search Algorithm

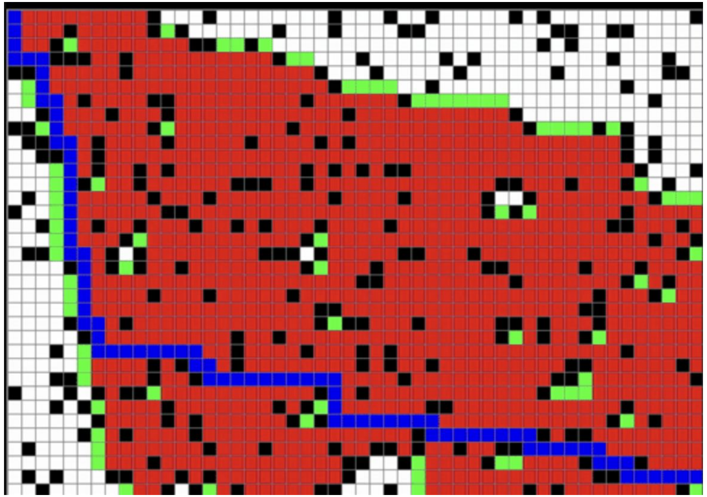
William Foy (wfoy) and Jacob Zych (jzych)

Summary

- Parallelization of A* Path Planning Algorithm
- C++ - Sequential, Centralized (pthreads), Decentralized (OpenMPI)
- Go - Sequential, Centralized (goroutines)

What is A*?

- Path Search Algorithm
- Minimization of path function f(n) = g(n) + h(n)
  - g(n) is the current cost
  - h(n) is a heuristic -> distance from node to target
- Use of priority queue to expand nodes w/ lowest costs

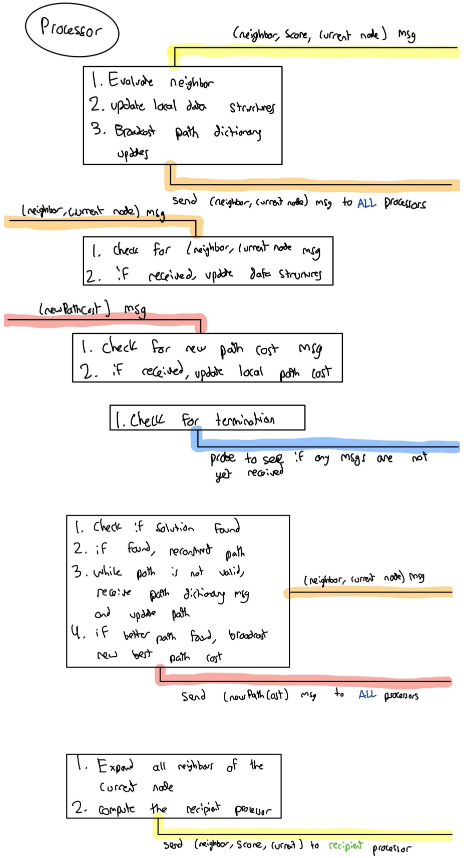


Centralized Parallelization

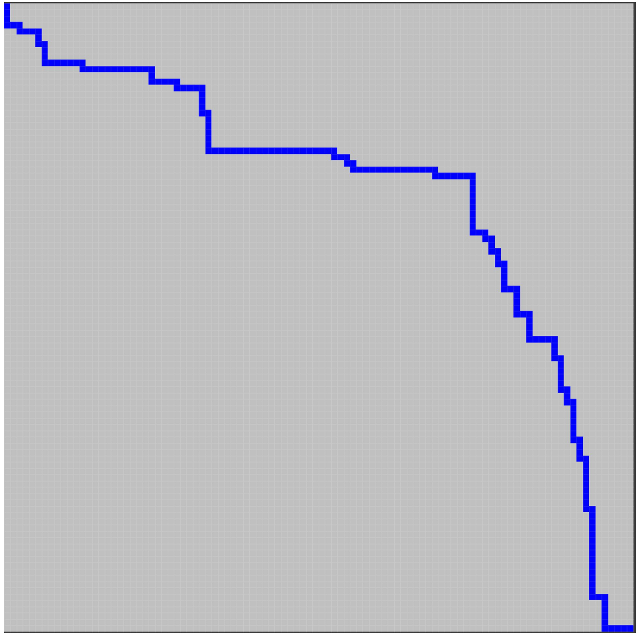
- All threads share an open set of nodes
- Each thread takes turns popping nodes from priority queue
- Synchronization overhead
- Implemented w/ C++ (Pthreads) and Go (goroutines)

Decentralized Parallelization

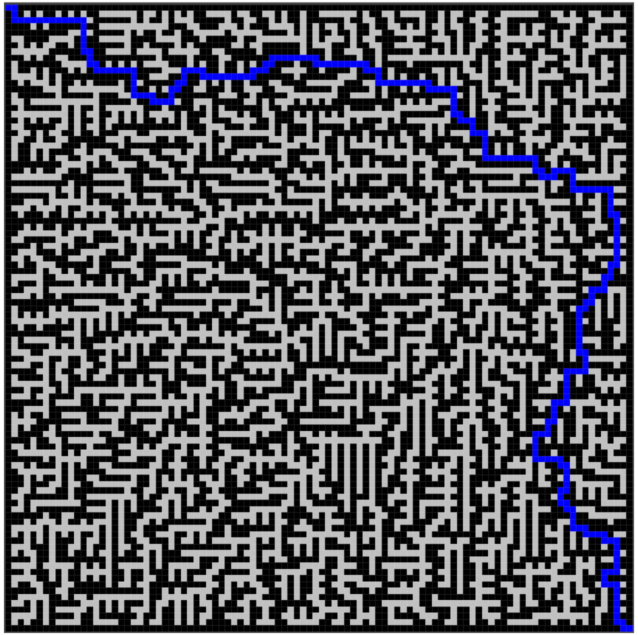
- Each thread has its own open set of nodes
- No synchronization needed
- Messages passed from thread to thread
- Implemented w/ C++ (OpenMPI)


Processor
(neighbor, score, current node) msg
1. Evaluate neighbor
2. update local data structures
3. Broadcast path dictionary updates
send (neighbor, current node) msg to ALL processors
(neighbor, current node) msg
1. check for (neighbor, current node msg
2. if received, update data structures
(newPathcost) msg
1. check for new path cost msg
2. if received, update local path cost
1. Check for termination
probe to see if any msgs are not yet received
1. Check if solution found
2. if found, reconstruct path
3. while path is not valid, receive path dictionary msg and update path
4. if better path found, broadcast new best path cost
(neighbor, current node) msg
send (newPathCost) msg to ALL processors
1. Expand all neighbors of the current node
2. compute the recipient processor
send (neighbor, score, current) to recipient processor

Centralized Results

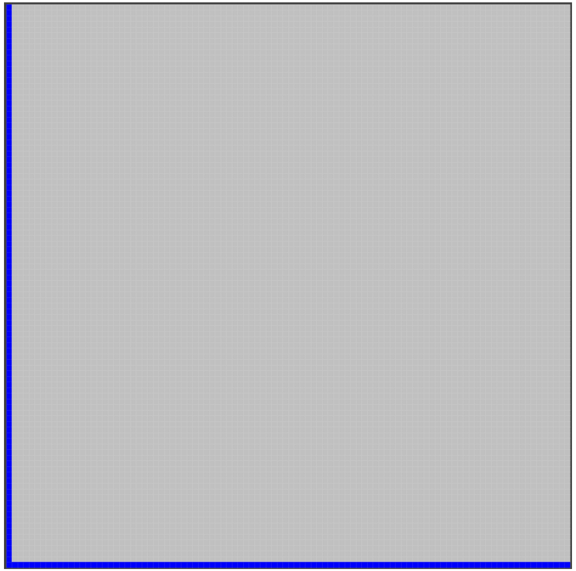


PThread - No Obstacles 100x100

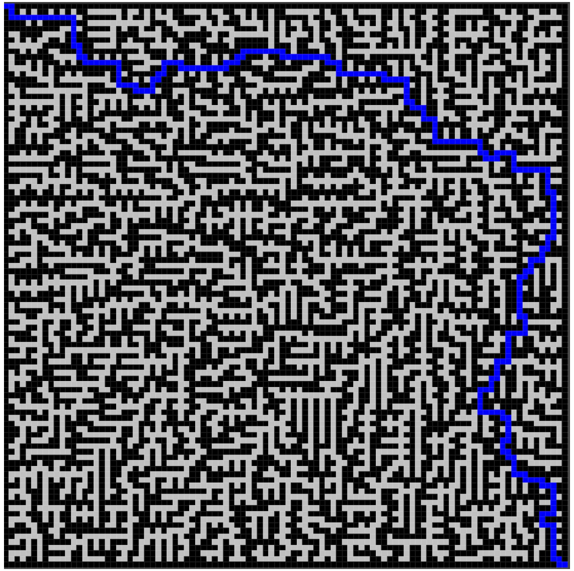


Go - Maze 100x100

Decentralized Results

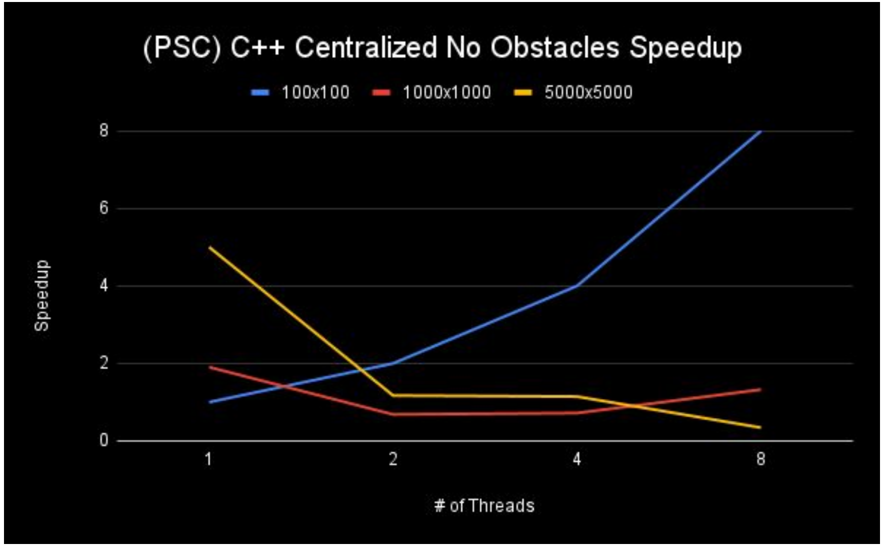


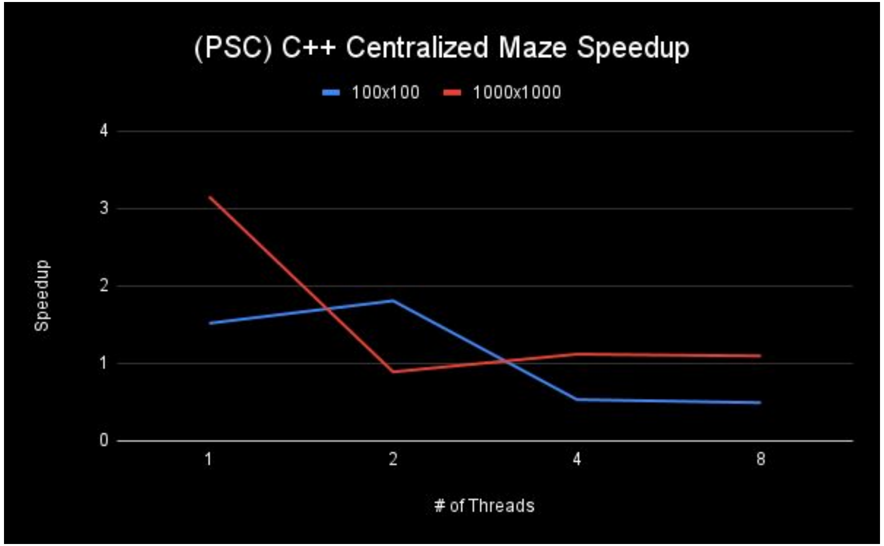
MPI - No Obstacles 100x100



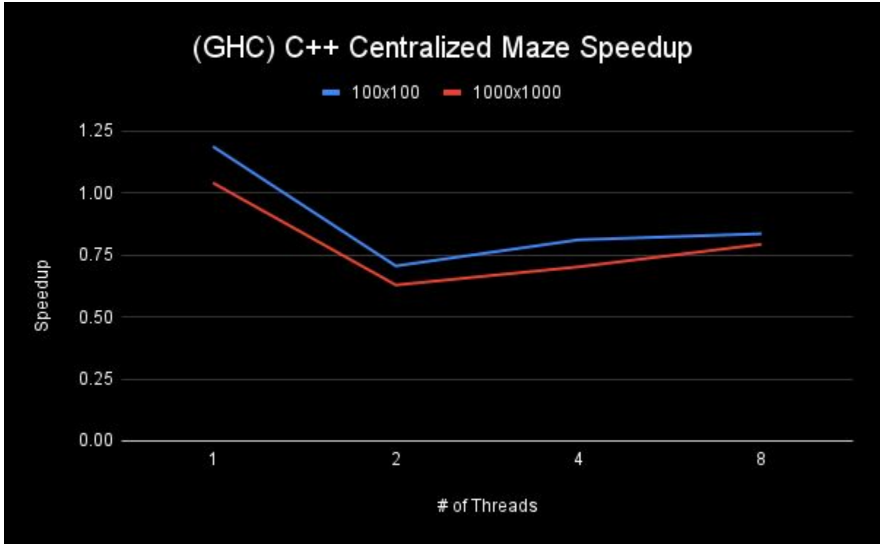
MPI - Maze 100x100

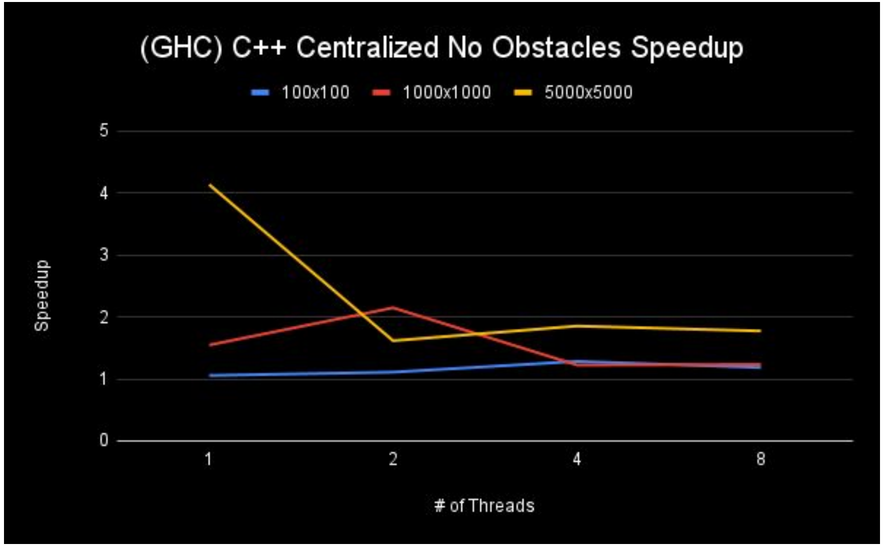
C++ Pthread Performance - PSC



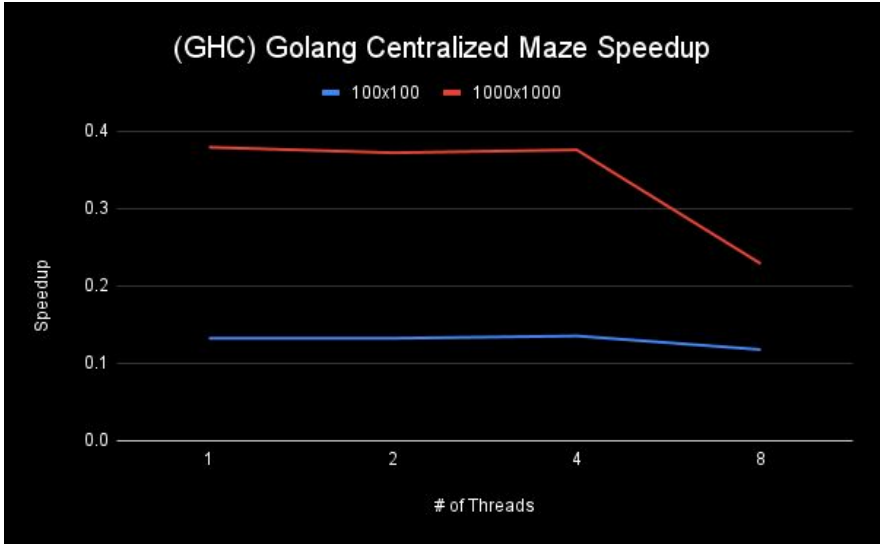


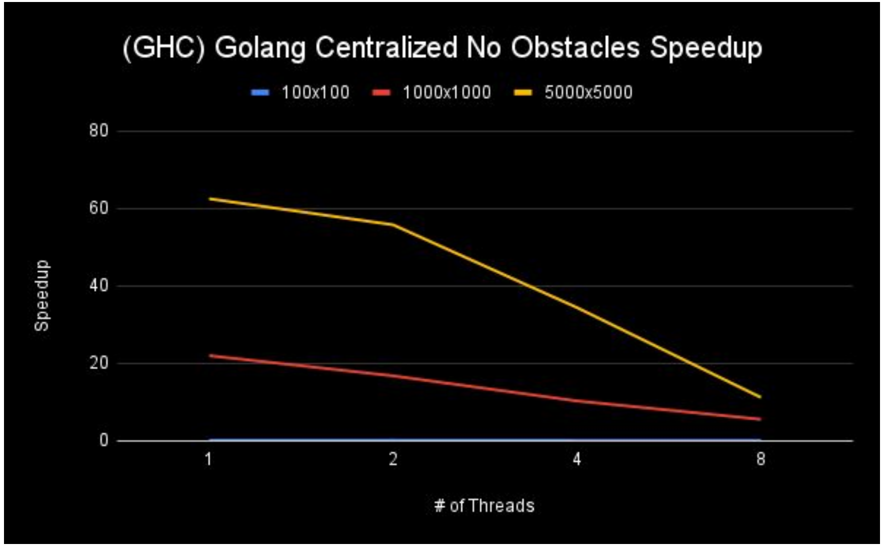
C++ Pthread Performance - GHC





Go Goroutine Performance - GHC





C++ OpenMPI Results

- 50-70% of execution time spent generating paths
- Majority of messages are entire dictionary updates
- Hashing could be used to cut down on path generation time

References

- https://arxiv.org/pdf/1708.05296.pdf
- https://en.wikipedia.org/wiki/A*_search_algorithm
- https://www.mpich.org/documentation/guides/
- http://www.mathcs.emory.edu/~cheung/Courses/561/Syllabus/92-MPI/async.html
- https://github.com/cmu15418s22/Assignment-4
- http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html