Parallel Game Tree Search

Jason Chalom - 711985
Liron Mizrahi – 708810

Background Knowledge

Background

Game Trees

- A game tree is a directed graph where each node indicates a gamestate and each edge indicates an action.
- A game tree is complete if it contains all possible actions from each possible state.
- Game trees can be incredibly large for simple games.
- Game trees are highly parallelisable as each subtree can be seen as disjoint trees and so can be searched individually.

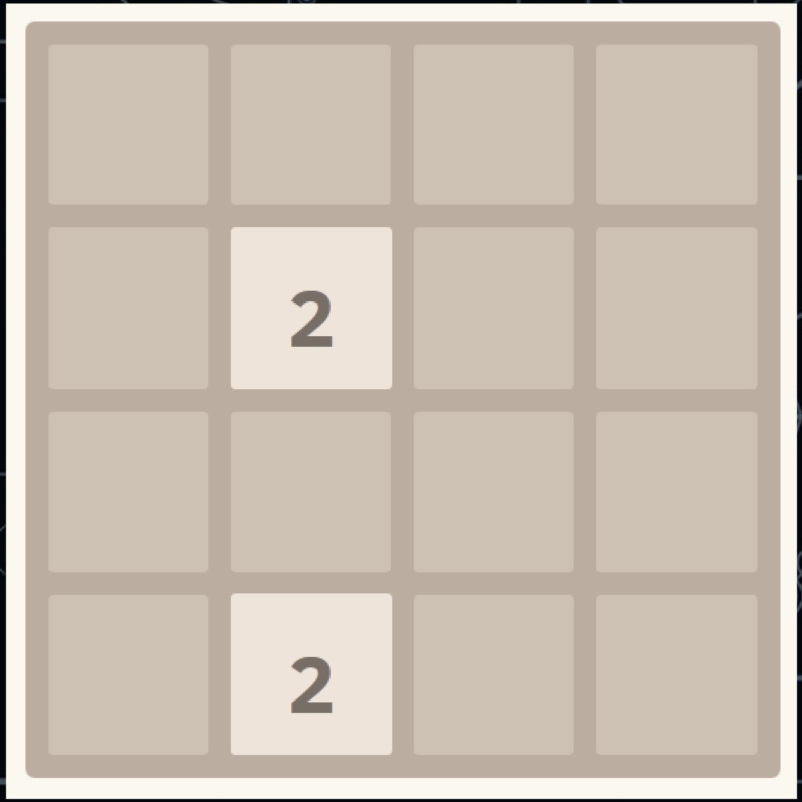
Background

The Game of 2048

- 2048 is a single-player simple, non-deterministic block sliding puzzle game.
- The rules are as follows:
    - The game is played on a square grid, i.e. 4x4 or 8x8 grid.
    - Every turn the player will choose a direction for the blocks to move, either up, down, left or right.
    - Tiles will slide as far as possible until stopped by the edge of the grid or another block.
    - If 2 tiles have the same number while colliding then they will combine into a single block and the new value will be the sum of their values.
    - After each action, a new tile will appear on the grid with a value of 2.
    - The game is won when a tile with value of 2048 appears on the board.

Background

Example of a 2048 Game



Technology Used

Technology

- The game was implemented using C++.
- The tree search was implemented in C++ and parallelised using Nvidia Cuda and Open MPI.
- HTML and CSS were used for the demo.

Strategies Used

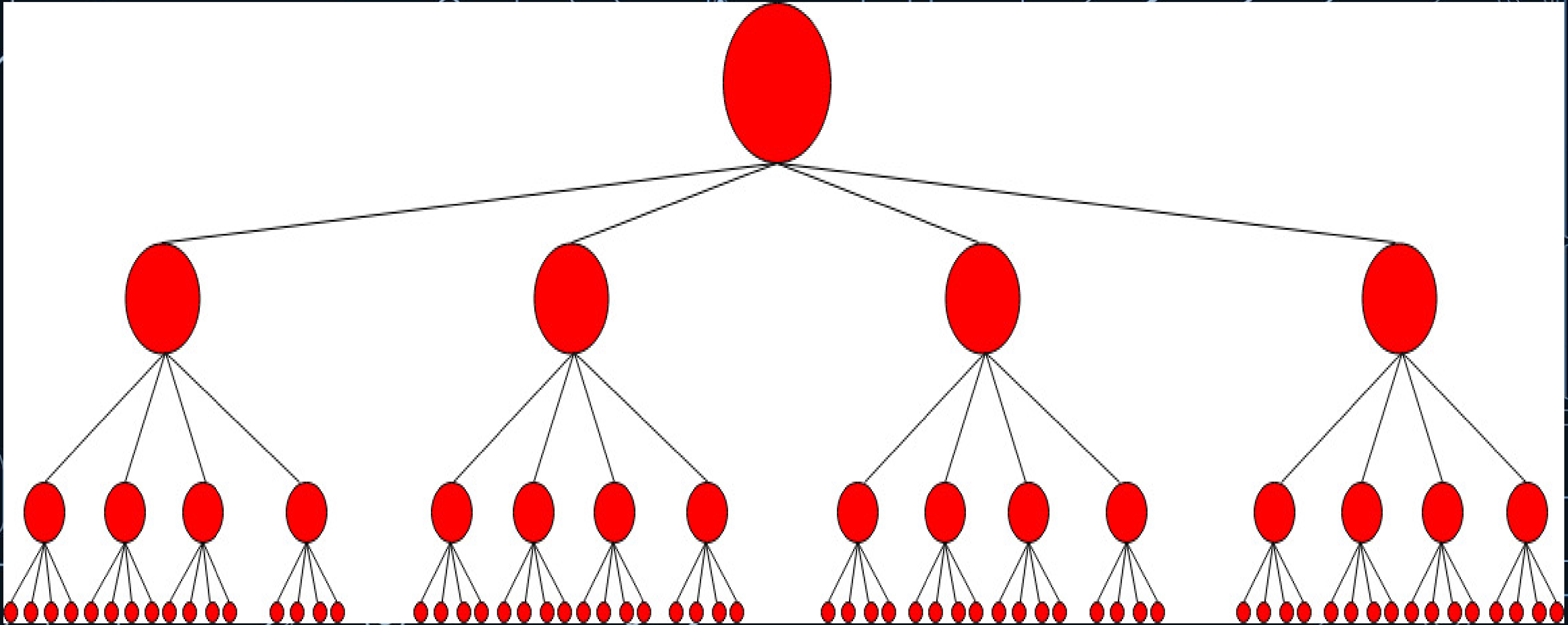
Strategies

Serial Approach

- We used a user controlled stack in order to build the tree.
- From the root node, all possible children are pushed to the stack.
- Then for each node at the top of the stack:
  - Each child is checked to see if its a dead state or if it is a solution state
  - A dead state is one where there are no possible moves left.
  - If not dead state or solution state then it is pushed to the top of the stack
  - If a solution is found then store it for later processing.
- However, since the game tree is so big for 2048, the tree is limited by the number of nodes.

Strategies

Serial Approach

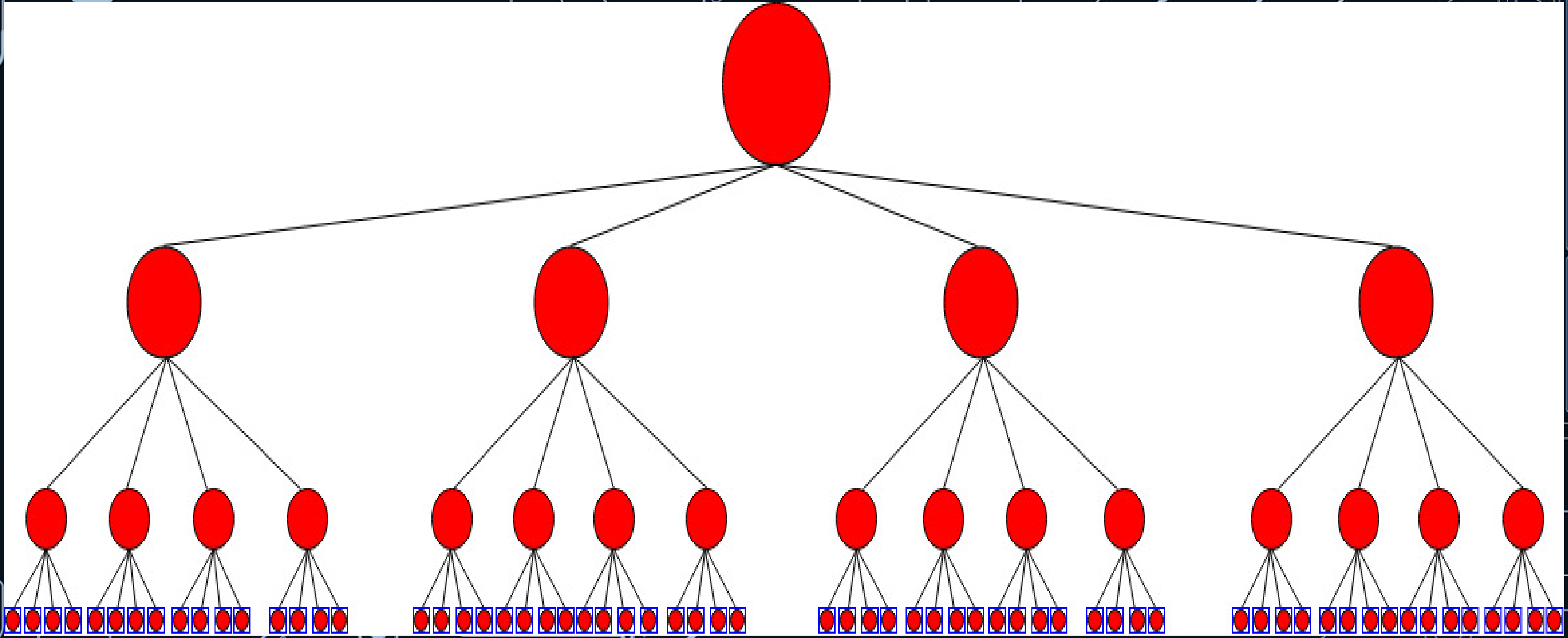


MPI Approach

- The MPI approach is very similar to the serial approach.
- The root process creates a small subtree until the number of leaves is equal to number of processes used.
- Each process is sent its initial state
- Each process then creates a serial search from the initial state.
- Then each process sends back its local optimal solution depth to the root process.
- The root process compares them and chooses the minimum.

Strategies

MPI Approach



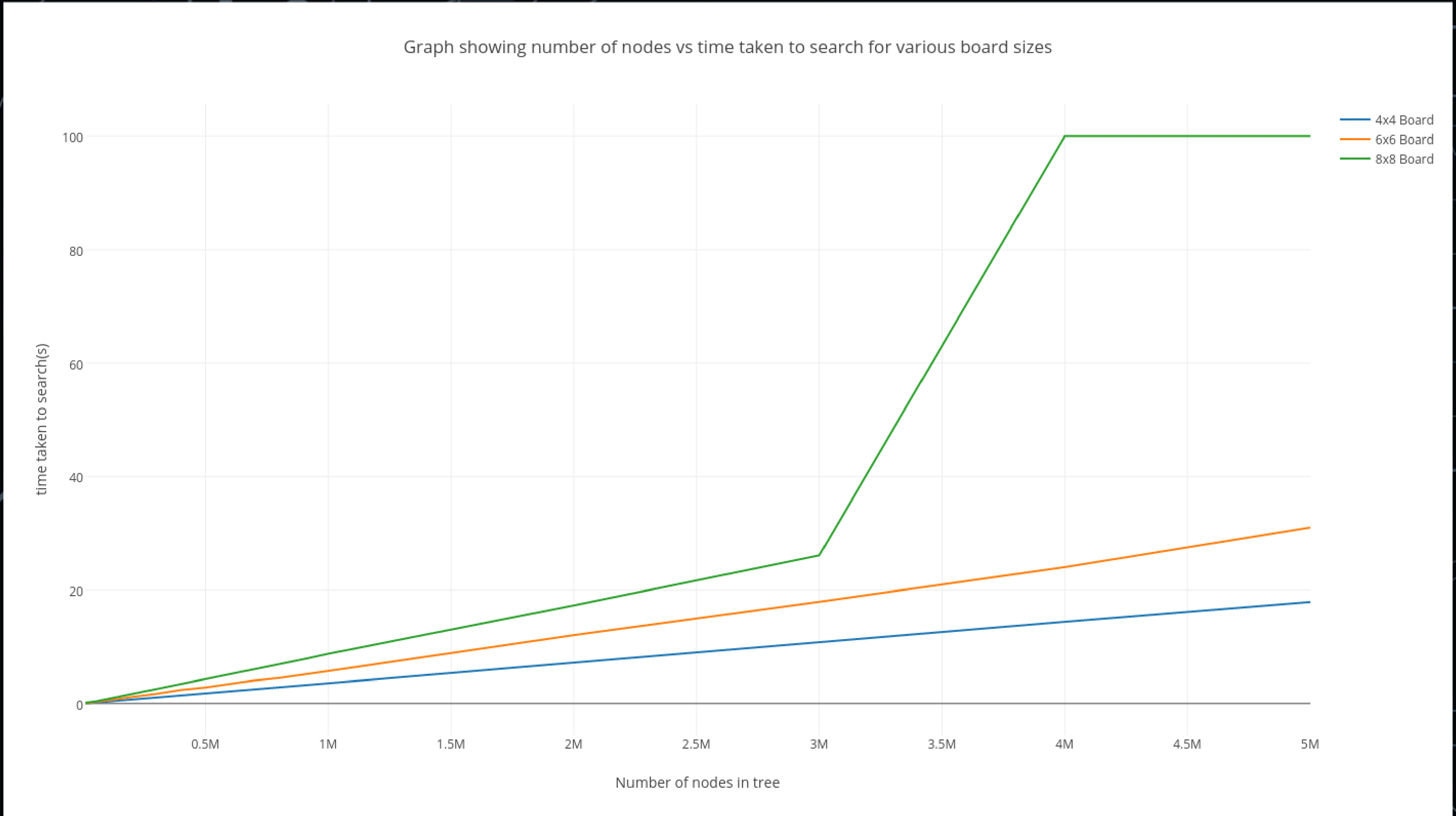
CUDA Approach

- The CUDA approach starts in a similar manner to the MPI approach.
- The host generates a small subtree and stores each initial state to be used in the first column of a matrix.
- Each row in this matrix is a subtree to be explored by a thread.
- For each entry in its row, each thread creates the nodes children and places them in the row of the matrix.
- For this approach to work, we must assume the tree is complete and symmetric.

CUDA Approach

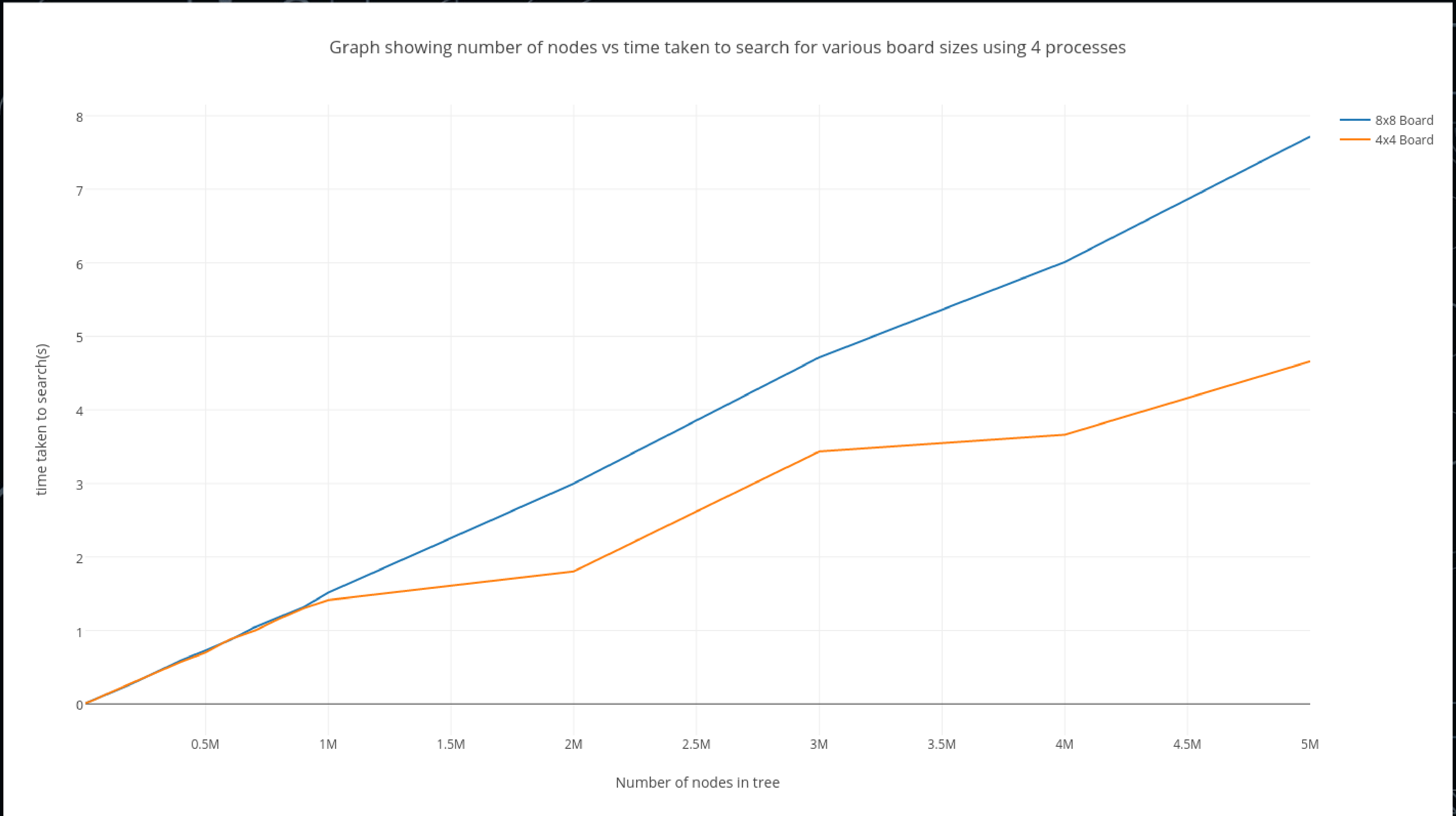
Results

Results

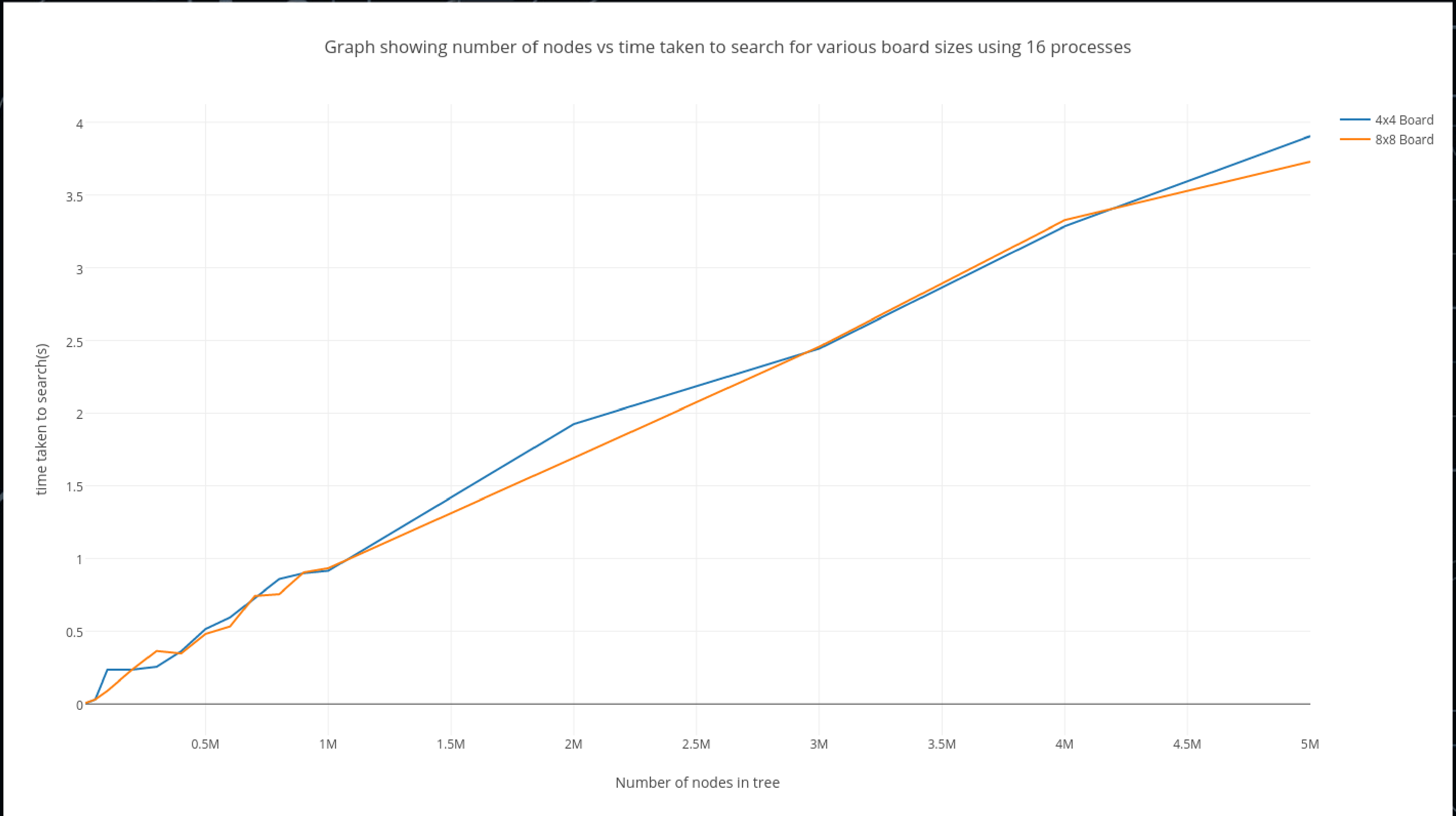


Graph showing number of nodes vs time taken to search for various board sizes
4x4 Board
6x6 Board
8x8 Board
time taken to search(s)
100
80
60
40
20
0
0.5M
1M
1.5M
2M
2.5M
3M
3.5M
4M
4.5M
5M
Number of nodes in tree

Results

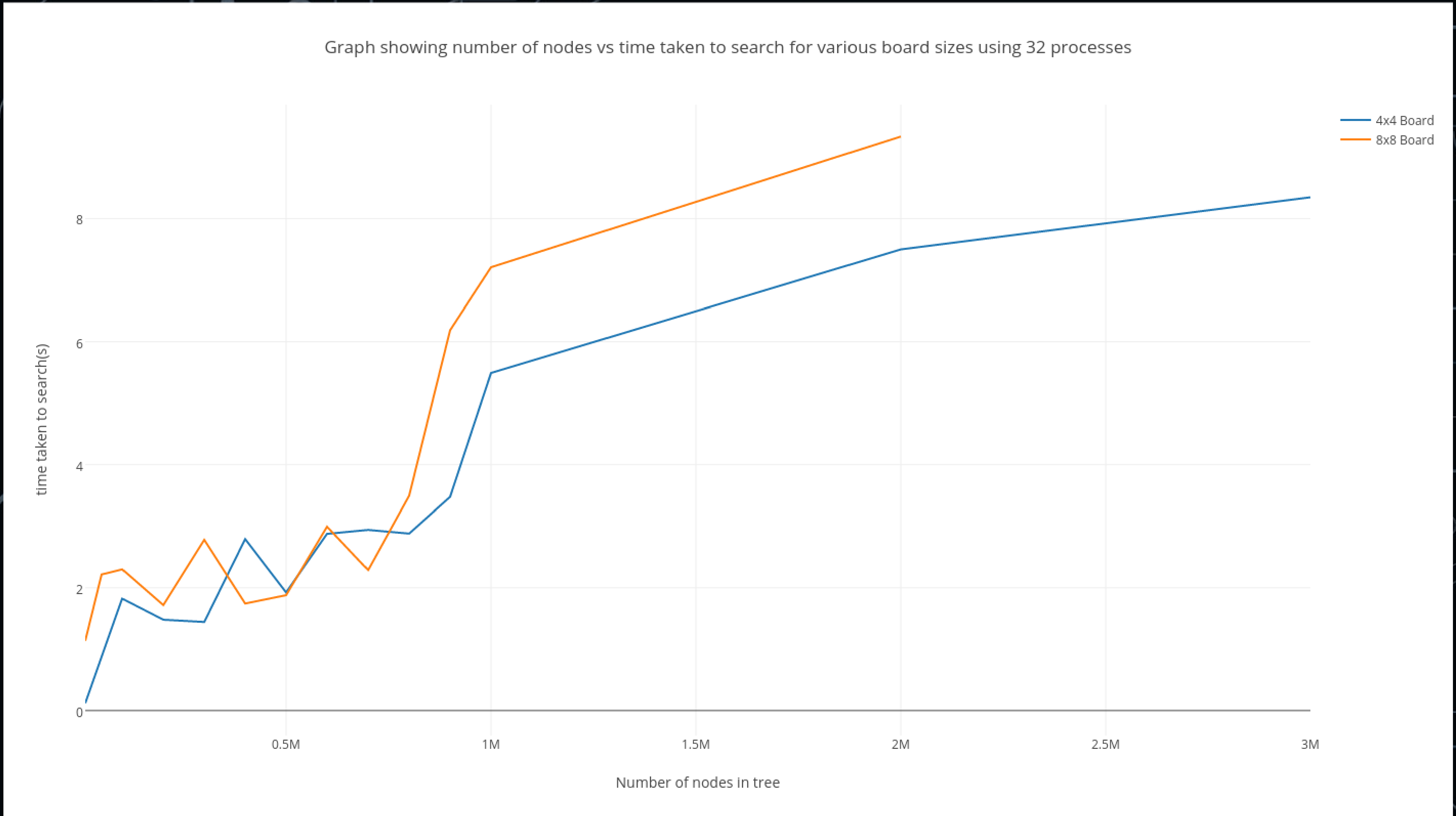

Graph showing number of nodes vs time taken to search for various board sizes using 4 processes
8x8 Board
4x4 Board
time taken to search(s)
0
1
2
3
4
5
6
7
8
0.5M
1M
1.5M
2M
2.5M
3M
3.5M
4M
4.5M
5M
Number of nodes in tree

Results



Graph showing number of nodes vs time taken to search for various board sizes using 16 processes
4x4 Board
8x8 Board
time taken to search(s)
0
0.5
1
1.5
2
2.5
3
3.5
4
0.5M
1M
1.5M
2M
2.5M
3M
3.5M
4M
4.5M
5M
Number of nodes in tree

Results



Graph showing number of nodes vs time taken to search for various board sizes using 32 processes
4x4 Board
8x8 Board
time taken to search(s)
0
2
4
6
8
0.5M
1M
1.5M
2M
2.5M
3M
Number of nodes in tree

Problems Encountered

Problems

- Inplace DFS and Recursive DFS
- Enforcing terminating conditions for tree generation.
- Using strings instead of ints.
- Underestimated game tree size
- Trying to code in such a way that code can be re-used for all 3 approaches.
- Most code had to be re-done for CUDA.
- The CUDA implementation was very challenging.
  - Indexing
  - Class function used pointers
  - Random number generation
- Creating initial subtrees for MPI and CUDA.
- Co-ordinating processes with MPI

Conclusions

Conclusions

- MPI is a simple way to gain a massive speed performance over the serial approach by sharing the work between processes.
- However, while CUDA may be a lot more complex to implement and may be slow, the limits of the tree size can greatly be expanded if memory is used efficiently.

Works Cited

- Sager, J. (2015, October 1). Free PowerPoint Template. Retrieved 2016, from http://sage-fox.com/
- En.wikipedia.org. *2048 (video game)*. [online] Available at: https://en.wikipedia.org/wiki/2048_(video_game) [Accessed 17 May 2017].
- NVIDIA Developer. *CUDA Zone*. [online] Available at: https://developer.nvidia.com/cuda-zone [Accessed 17 May 2017].
- Open-mpi.org. *Open MPI: Open Source High Performance Computing*. [online] Available at: https://www.open-mpi.org/ [Accessed 17 May 2017].
- Softonic. *2048 Game*. [image] Available at: https://features.en.softonic.com/how-to-win-at-2048 [Accessed 17 May 2017].