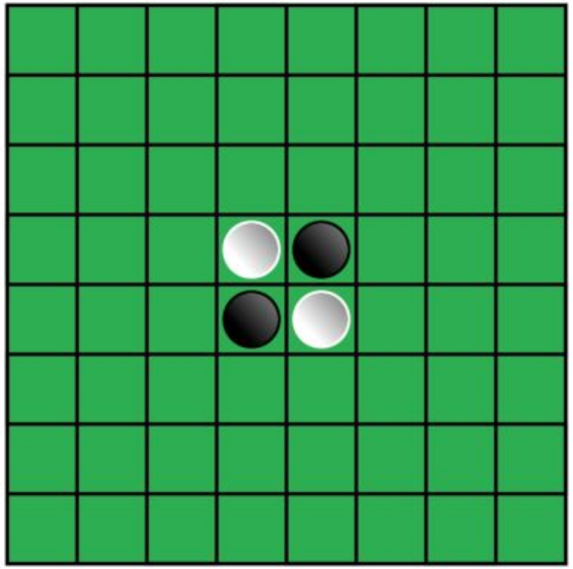
Parallel Functional Programming Final Project

Othello

Noam Hirschorn Dan Ivanovich

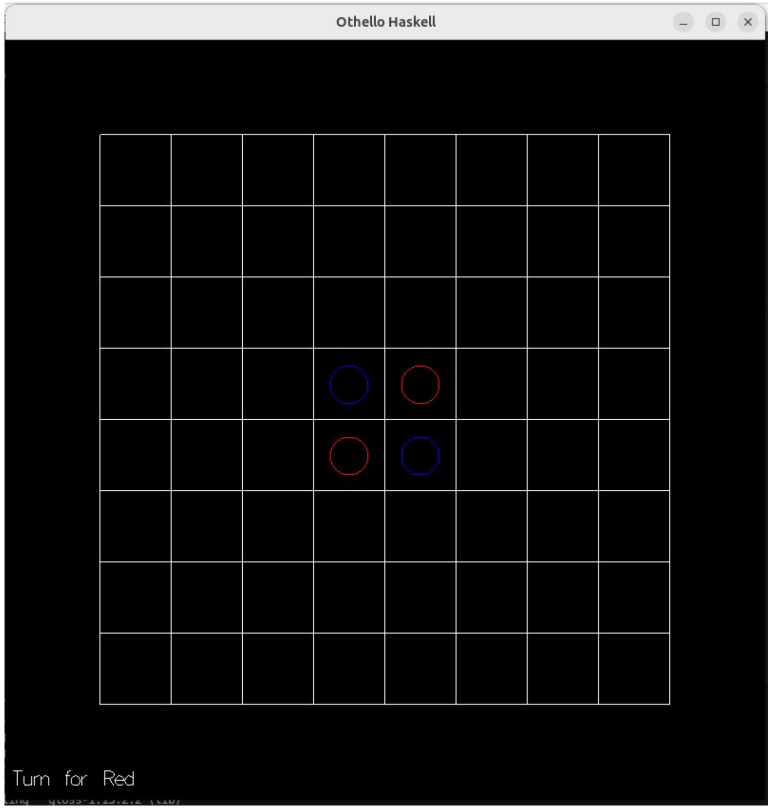
Background

- **Othello is two players placing different colors of tiles on a board**
  - **More tiles of your color = more points**
- **Starting code had minmax search tree for an AI option to play moves**
  - **Had a full GUI to allow player vs computer**
  - **No alpha beta pruning**



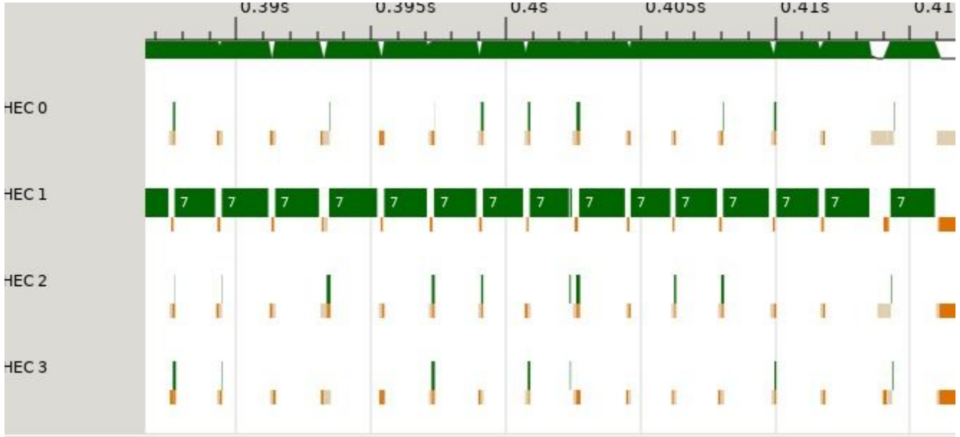
Initial Steps

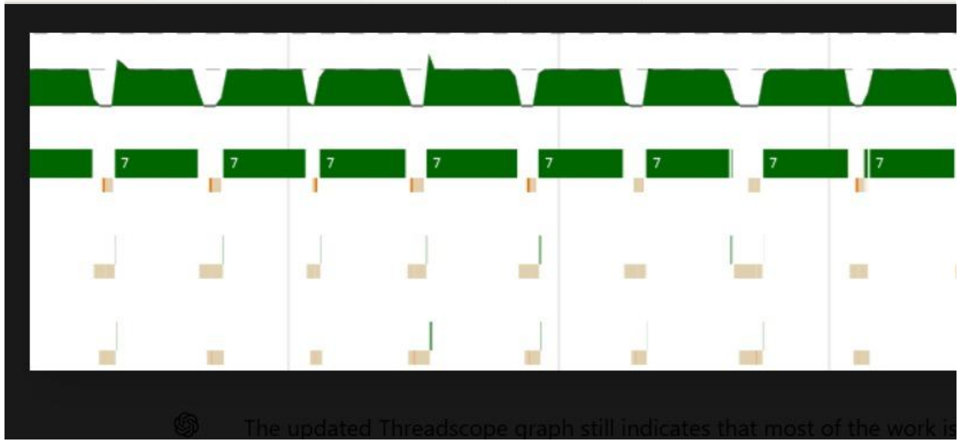
- Disable GUI
- Read position from file
  - Adjust inputs and create sample game states
- Add alpha beta pruning
- Allow setting depth of search tree
- Print resulting game board and move



Initial parallelization

- Used parMap
- Multiple issues
  - Many Sparks fizzling/GC'ing
    - Due to nested/recursive parallelization calls
  - Dependency issues
    - Only 1 thread running at a time
    - Threads waiting on each other for alpha beta values
    - Need to use rdeepseq to force parallel evaluation

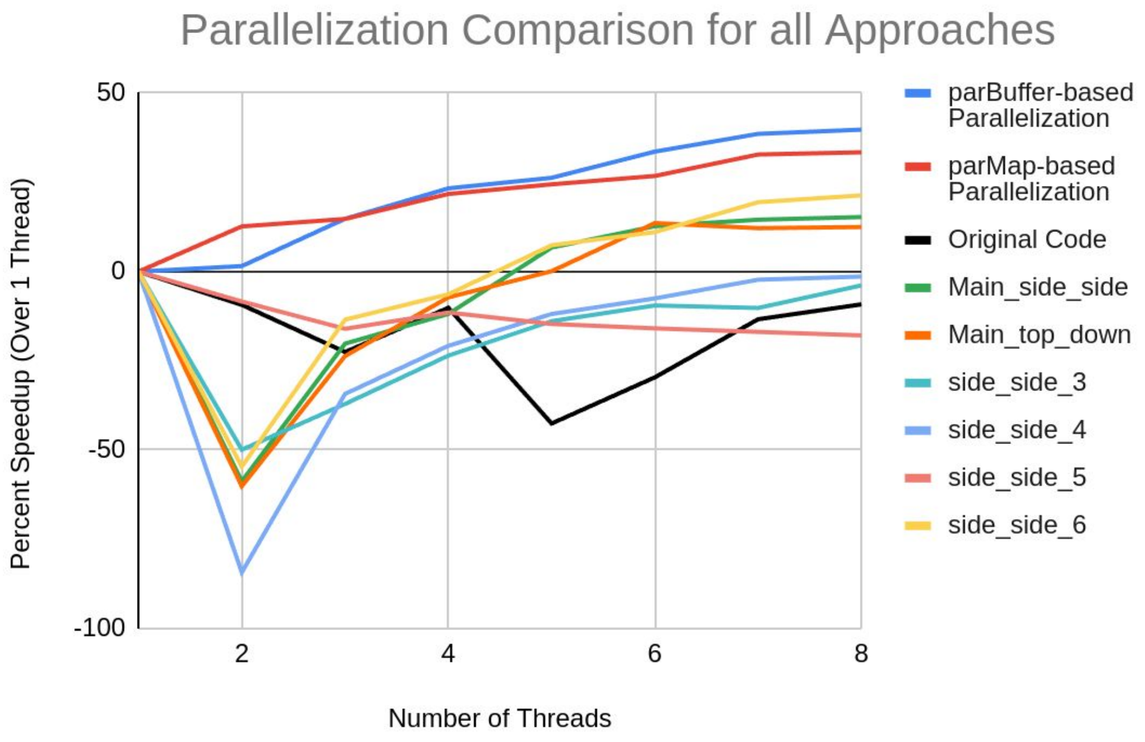



The updated Threadscope graph still indicates that most of the work is

Parallelization theories

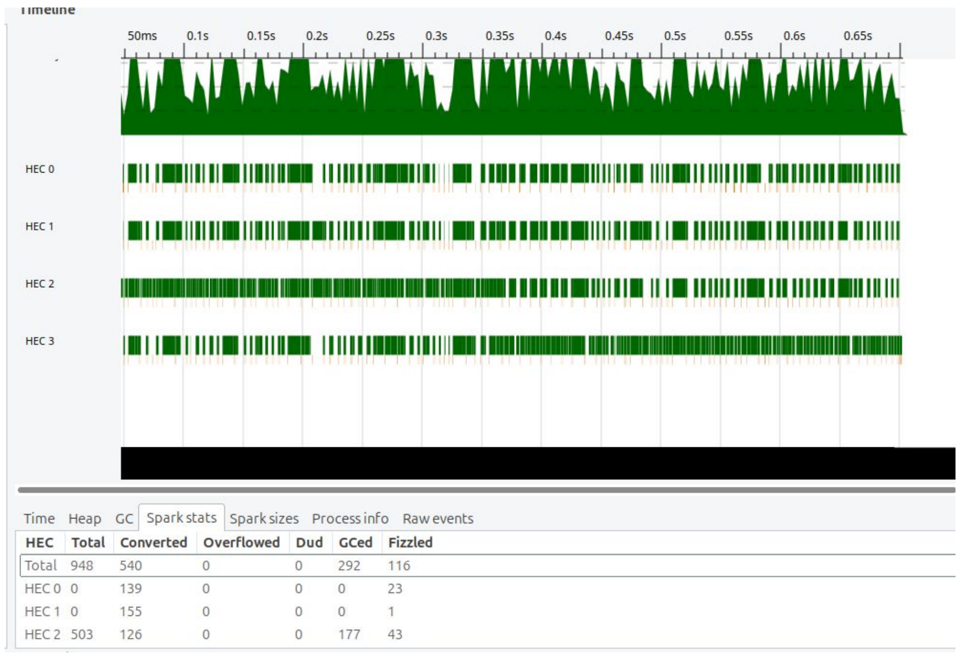
- Top down idea
    - Parallelize top rows as much as possible
        - Everything below top rows are just fully run in parallel
        - Idea - parallelize within a ‘parallelized subtree’
- Side side
    - Parallelize every child of every node at one row
    - After each node, move on to next one
    - Lose less alpha beta knowledge than when silo’d like top down
- Recursive parallelization is wasteful
    - Leads to fizzling/GC’ing as duplicate sparks are made
    - Go with Side Side ideas
    - Add input of parallelDepth for what row to parallelize on in search tree

Our Improvements Over Time…



Improvement Attempts

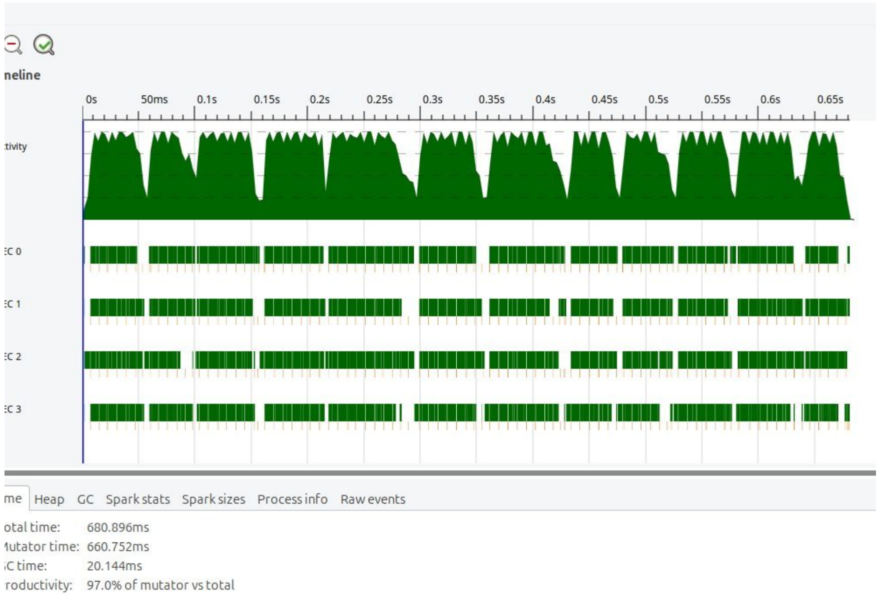
- Introduce explicit chunking based on number of threads
  - If anything, negative effect
- Use chunksOf
  - Theory - divide moves left by number of threads
  - Balanced search tree, so just adds overhead



HEC	Total	Converted	Overflowed	Dud	GCed	Fizzled
Total	948	540	0	0	292	116
HEC 0	0	139	0	0	0	23
HEC 1	0	155	0	0	0	1
HEC 2	503	126	0	0	177	43

Improvement Attempts cont.

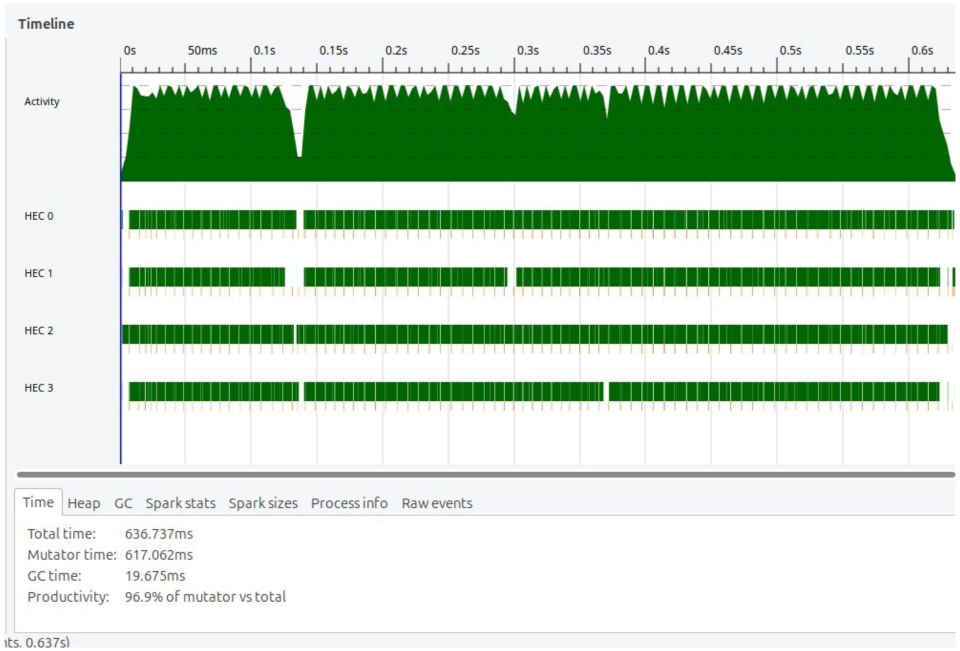
- Attempted to allow threads which finish early to move on to next node
  - Require excess thread communication about current alpha beta values
  - Difficult stopping thread once on path now pruned
  - Relatively low idleness anyways
- Abandoned effort
  - Difficulty getting a working version
  - Overhead from communication almost certain to outweigh benefits



otal time: 680.896ms
Mutator time: 660.752ms
GC time: 20.144ms
roductivity: 97.0% of mutator vs total

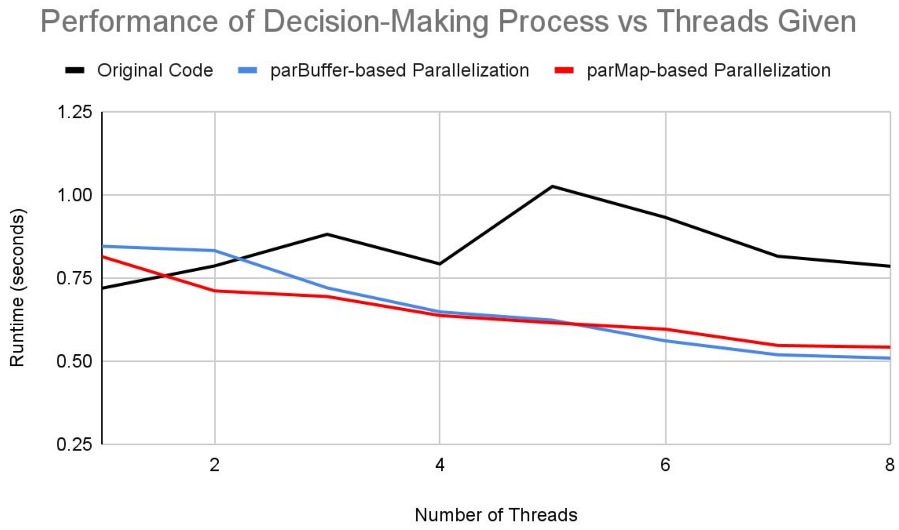
Final improvement attempts

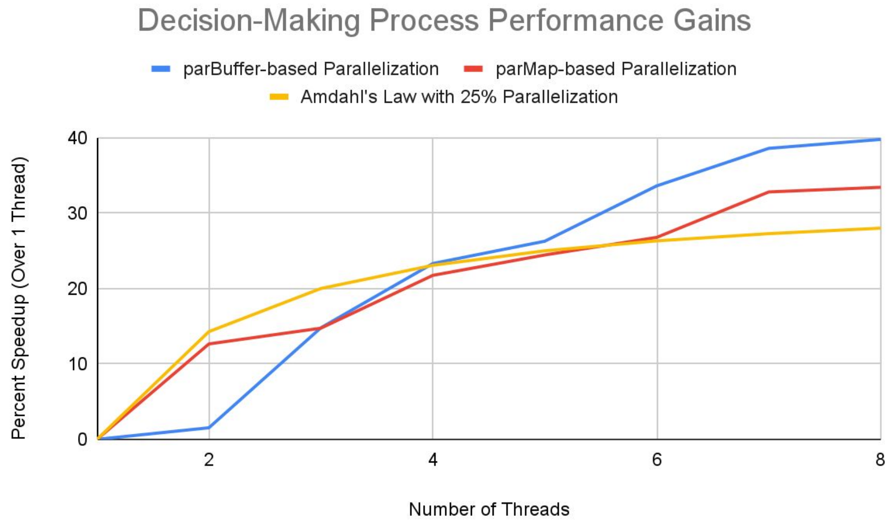
- Start using parBuffer in case threads need work to do
  - Changed distribution of results, but not necessarily positive
  - Competitive as a final version
- Attempted to use parListChunk to try to see if worked better with chunksOf
  - Error - forgot to change function call - only noticed when preparing report
  - Actually was doing basic parMap
  - Appeared to end up being best version
  - A lot of time spent tinkering
    - Apparent “improvements” on this just due to noise
  - Tried intended version - disappointing results



```
Total time:     636.737ms
Mutator time:   617.062ms
GC time:        19.675ms
Productivity:   96.9% of mutator vs total
```

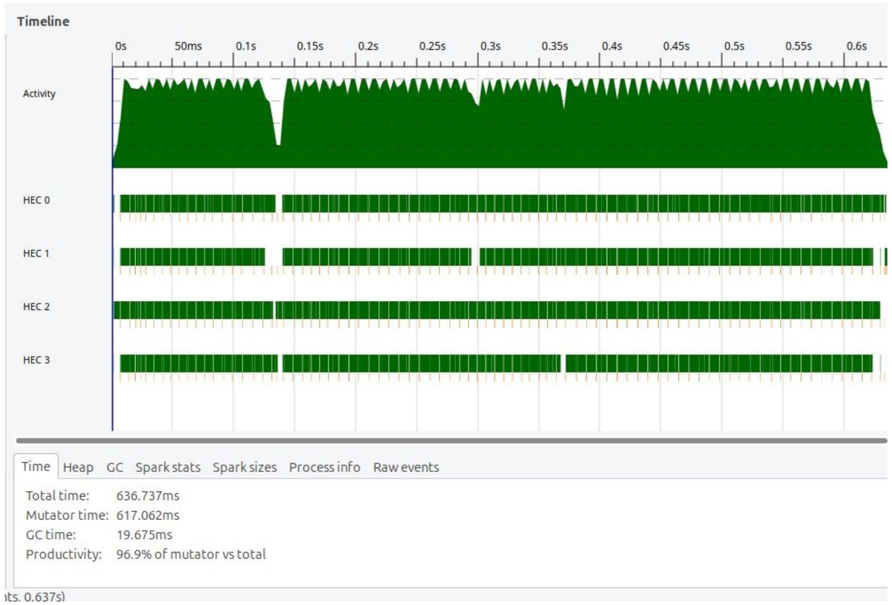
parBuffer-based and parMap-based Parallelization





All data collected while running at depth = 5, parallelDepth =4, starting board = custom_game_2

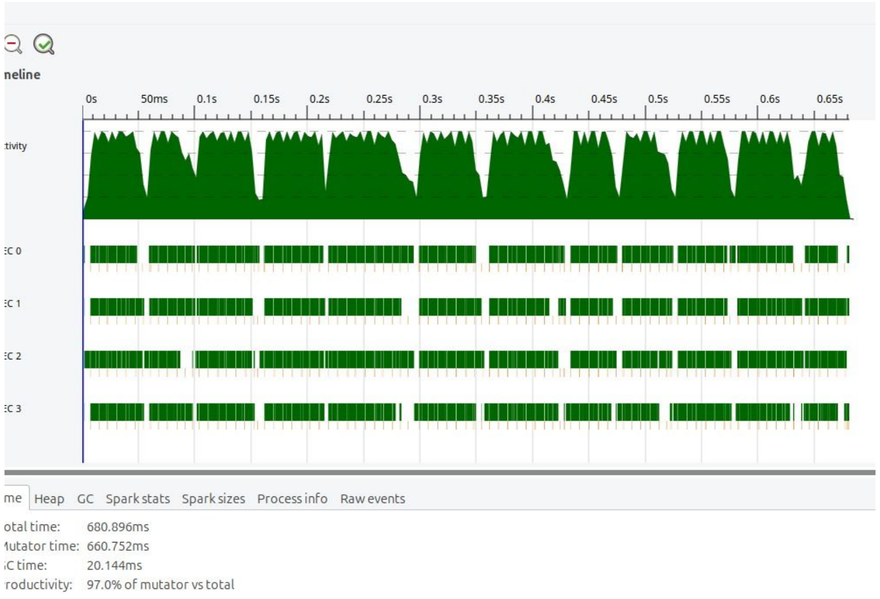
parBuffer-based and parMap-based Parallelization



```
Total time:     636.737ms
Mutator time:   617.062ms
GC time:        19.675ms
Productivity:   96.9% of mutator vs total
```

ıts. 0.637s)

parBuffer-based



```
otal time:      680.896ms
lutator time:   660.752ms
C time:         20.144ms
roductivity:    97.0% of mutator vs total
```

parMap-based

Reflection on process/Conclusion

- Obvious mistake not updating function call
- Turning off alpha beta low effect on relative speedup
  - Suggests losing info from there not main issue holding back results
- Final code implied code a bit over 25% parallelizable
  - Consistent across sizes of search tree
- Most sparks still converted

Theory - main cause of sequentialism was due to adaptation from codebase

Maybe a background function slowing things down?

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

$$\frac{P}{N} - P = \frac{1}{S} - 1$$

$$P = \frac{\frac{1}{S} - 1}{\frac{1}{N} - 1} = \frac{\frac{N}{S} - N}{1 - N} = \frac{N - \frac{N}{S}}{N - 1}$$