Parallel Function Programming
Final Project Word-Search-2

Team:

- Sean Zhang (srz2116)
- Keith Lo (kl3695)
- Ardrian Wong (aaw2179)

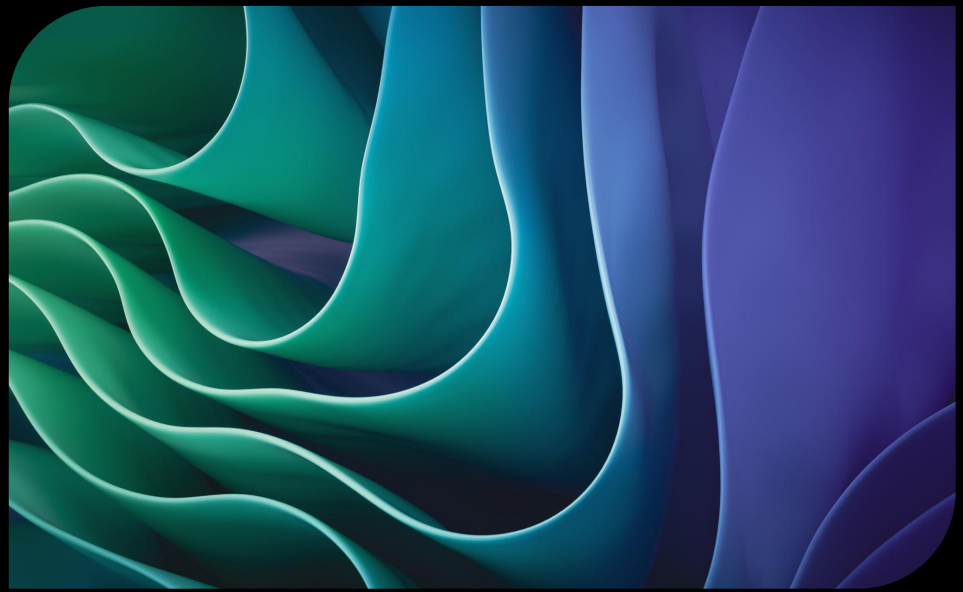
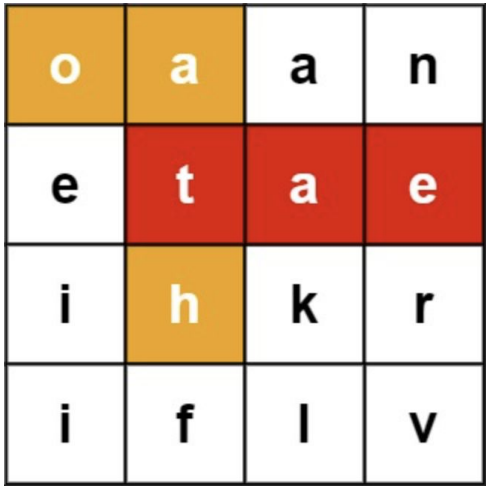


Table of Contents

Problem Statement

Given an m x n board of characters and a list of strings words, return *all words on the board*.



Input: board = [["o","a","a","n"],["e","t","a","e"],["i","h","k","r"],["i","f","l","v"]], words = ["oath","pea","eat","rain"]
Output: ["eat","oath"]

Sequential Algorithm

1. Insert the target words in a Trie for efficient search during DFS.
2. Initiate DFS for each cell (searchFromCell) in the grid (this happens in findWords)
3. Check if the character in the current cell matches the character in the trie.
    a. If true, mark the current cell as visited and continue DFS all directions. Add any words found during DFS to the results
    b. If false, don’t continue DFS from the current cell

Proposed Methods of Parallelism

- ParallelWords: Parallelize the search for each target word
- ParallelDepth: Parallelize recursive DFS calls up to a configurable depth
- ParallelSubgrids: Divide the input grid into N^2 subgrids and parallelize DFS from each of them

Technical Challenges

- Data Generation:
  - Leetcode test cases insufficient for testing
  - No online word search generator that generates snaking target words
- Lazy Evaluation with par:
  - List of results was full of thunks. Resulted in timing in problems timing the algorithm.

Algorithm Evaluation

We benchmark performance on the following the following three test cases:

- 100x100 grid with 10 target words
- 500x500 grid with 20 target words
- 1000x1000 grid with 30 target words

We first parse the input from disk and then time the execution of the algorithm itself. This approach ensures that we exclude I/O time from our benchmarks.

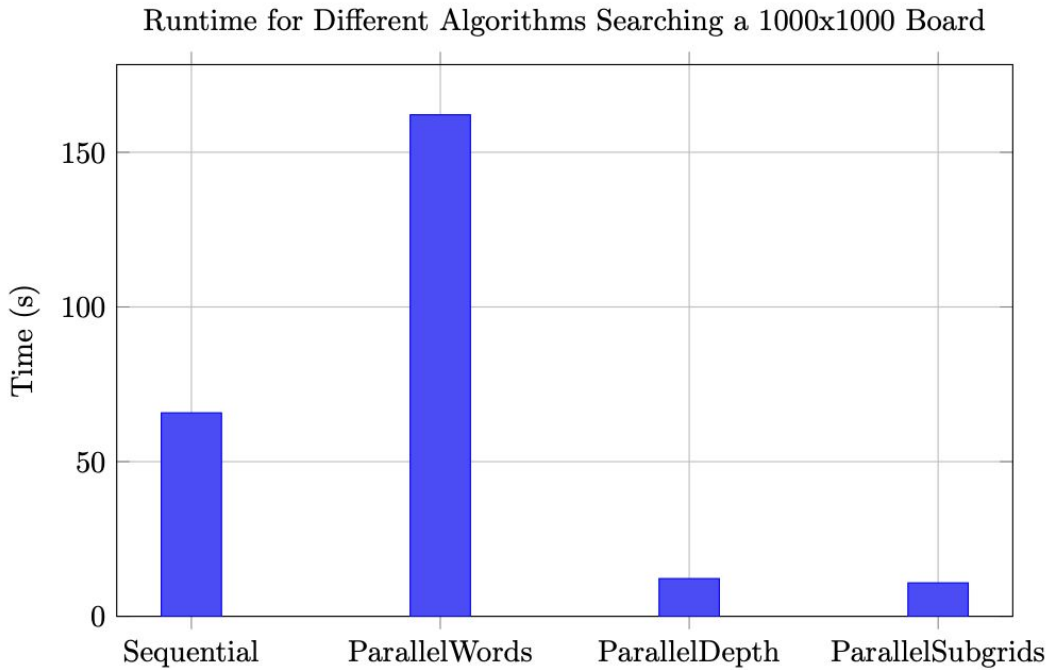
Note: Target word length ranges from 8-15 characters.

Hardware

All testing was conducted on a 2022 Macbook Air:

```
(base) MacBook-Air-691:PFP sean$ sysctl -a machdep.cpu
machdep.cpu.cores_per_package: 8
machdep.cpu.core_count: 8
machdep.cpu.logical_per_package: 8
machdep.cpu.thread_count: 8
machdep.cpu.brand_string: Apple M2
```

Overall Results



All parallel algorithms were run with 8 threads. ParallelDepth has depth 8 and ParallelSubgrids has 196 subgrids.

Sequential Results

	Board Size		
	100x100	**500x500**	**1000x1000**
Time (s)	0.02597	5.491277	65.772943

Table 1: Sequential algorithm runtimes.

ParallelWords Results

Threads	Board Size		
	100x100	500x500	1000x1000
1	0.068571	25.964812	842.595844
2	0.042393	14.382055	460.157849
3	0.032634	11.135458	305.842003
4	0.025253	8.376221	257.679208
5	0.025845	7.921799	205.091986
6	0.020172	7.079879	192.660891
7	0.019657	6.866078	163.534461
8	0.021034	6.735405	162.122001

Table 2: ParallelWords runtimes (in seconds).

ParallelWords Results

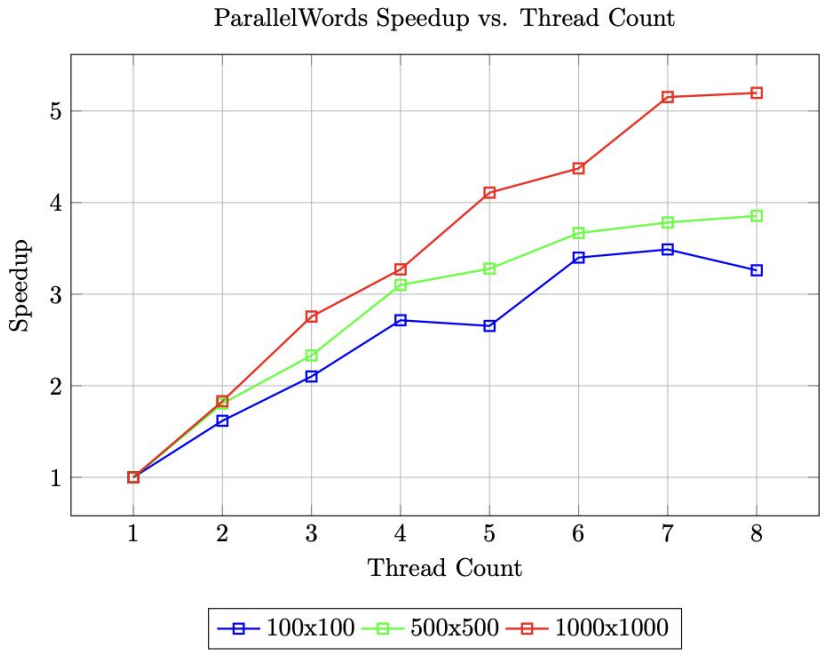
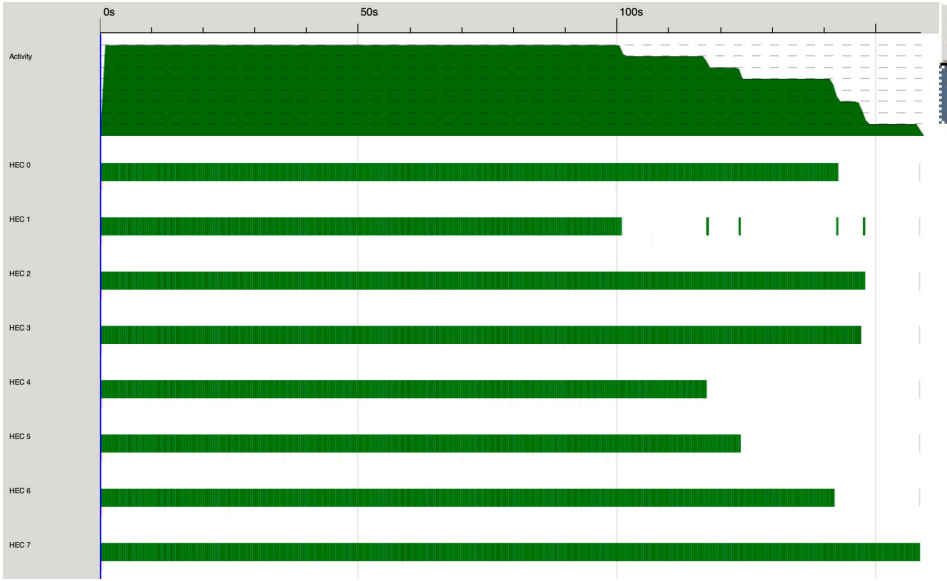


Figure 1: Speedup for varying thread counts across different board sizes.

ParallelWords Results



HEC	Total	Converted	Overflowed	Dud	GCed	Fizzled
Total	30	29	0	0	0	1
HEC 0	0	4	0	0	0	0
HEC 1	30	1	0	0	0	0
HEC 2	0	3	0	0	0	1
HEC 3	0	4	0	0	0	0
HEC 4	0	4	0	0	0	0
HEC 5	0	4	0	0	0	0
HEC 6	0	4	0	0	0	0
HEC 7	0	5	0	0	0	0

ParallelWords threadscope graph and spark stats for 1000x1000 board, -N8.

ParallelDepth Results

Threads	Depth							
	1	2	3	4	5	6	7	8
1	11.70577	12.113667	13.801015	12.34687	11.565189	11.279337	11.866786	11.241193
2	11.28168	12.60596	11.886404	12.59457	11.249191	12.183987	11.441741	11.525345
3	11.907646	12.11129	12.81159	11.322576	12.4556	11.543598	12.298782	11.504465
4	11.485883	11.072132	11.509928	11.482786	11.998397	11.600105	11.861788	11.005588
5	11.684131	11.83232	11.658778	11.768344	12.078482	11.875231	12.134167	11.808079
6	11.37276	12.003249	11.371989	12.191675	12.297596	11.051718	11.889646	11.54167
7	11.954613	11.941797	12.601917	12.127493	11.678604	11.495271	11.818165	11.987016
8	11.671538	11.944176	11.662381	11.772531	11.695728	11.620753	13.652866	12.189594

Table 5: ParallelDepth runtimes (in seconds) for a 1000x1000 board.

ParallelDepth Results

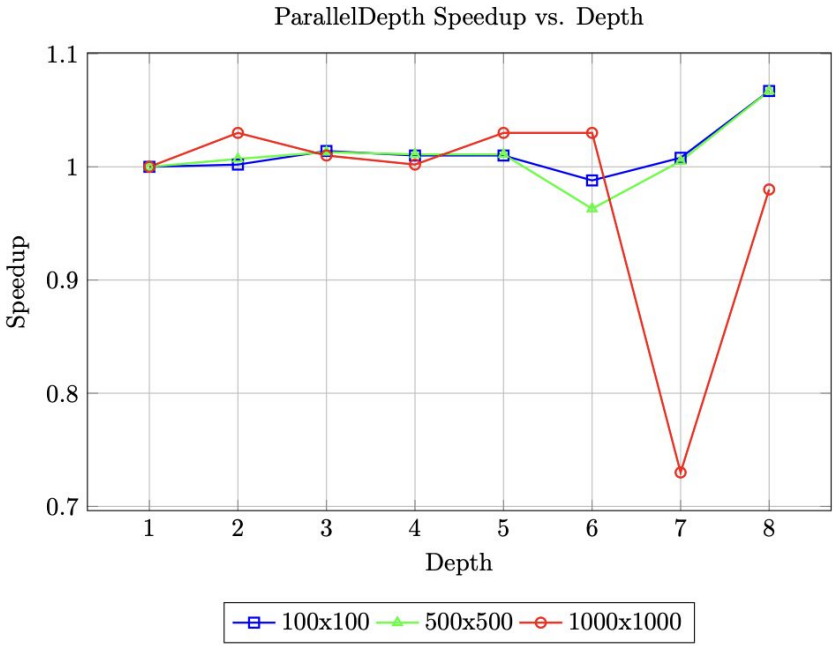


Figure 2: Speedup for varying depth across different board sizes, -N8.

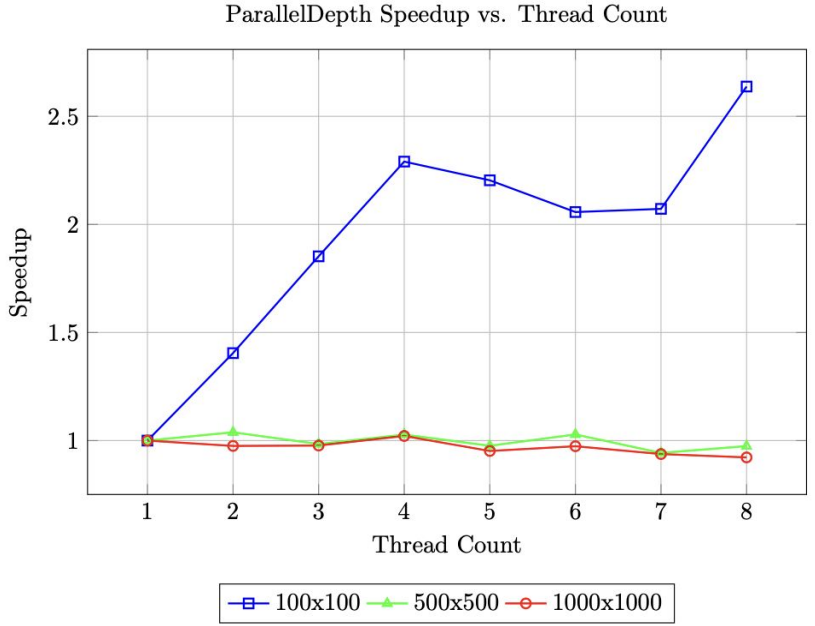
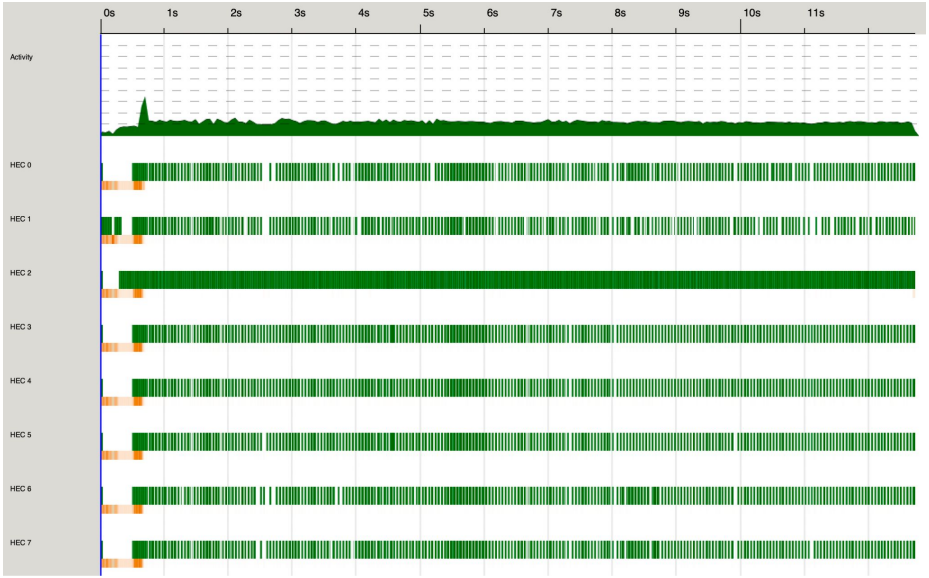


Figure 3: Speedup for varying thread count across different board sizes, depth 8.

ParallelDepth Results



HEC	Total	Converted	Overflowed	Dud	GCed	Fizzled
Total	4547676	24752	1053762	0	698930	2303288
HEC 0	6272	3558	0	0	3	133695
HEC 1	7736	3878	0	0	4	136728
HEC 2	4499552	15	1053762	0	698683	118
HEC 3	7296	3901	0	0	34	578039
HEC 4	7232	3641	0	0	44	506654
HEC 5	8184	4056	0	0	55	605699
HEC 6	5104	2490	0	0	51	171990
HEC 7	6300	3213	0	0	56	170365

ParallelDepth threadscope graph and spark stats for 1000x1000 board, depth 8, -N8.

ParallelSubgrids Results

Threads	Subgrids									
	1	**4**	**16**	**36**	**64**	**100**	**144**	**196**	**256**	**1000000**
1	65.801883	65.945074	66.799887	66.388034	67.485956	67.299456	67.051436	67.883795	67.056246	310.392841
2	64.714528	37.785980	36.057934	35.417008	35.813678	35.300202	35.437499	35.487171	35.627685	257.808374
3	65.622570	34.679536	27.001564	25.326437	25.149757	24.586220	24.508003	24.854055	25.193567	240.548454
4	67.589009	21.303343	20.879299	19.191283	18.958903	18.670745	18.48981	18.772108	18.509092	237.812547
5	66.191874	21.598264	18.539671	16.277188	16.005820	16.318601	16.025915	15.863053	15.930072	231.578546
6	66.546687	21.820648	15.904608	14.798572	14.075356	14.538098	14.106605	13.942253	13.983451	242.619650
7	66.201424	22.373584	15.098906	13.849142	12.851644	13.162786	12.713559	12.491494	12.521412	245.427277
8	67.099902	22.044958	12.315757	12.435074	11.946892	11.677746	11.468487	10.809685	11.501718	244.793032

Table 8: ParallelSubgrids runtimes (in seconds) for a 1000x1000 board.

ParallelSubgrids Results

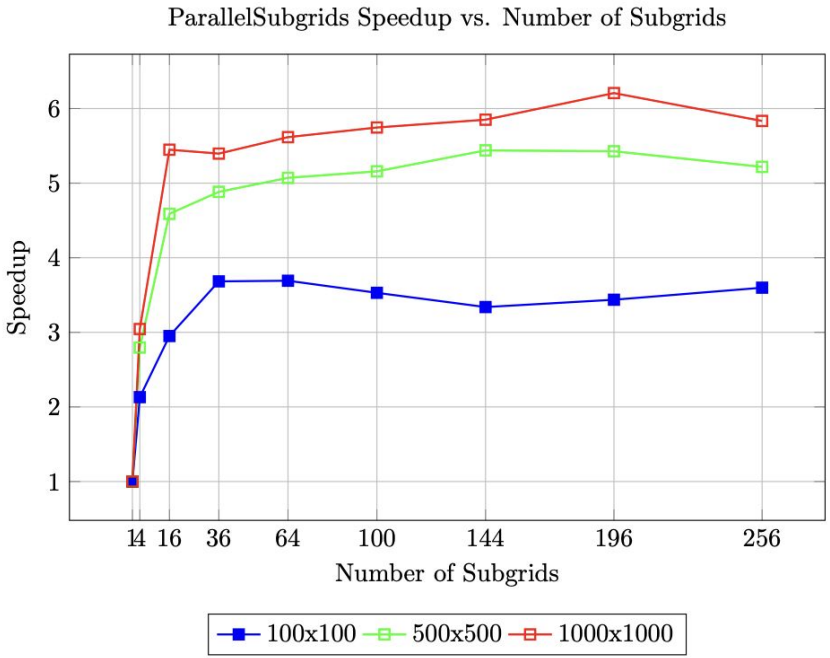


Figure 4: Speedup for varying numbers of subgrids for different board sizes, -N8.

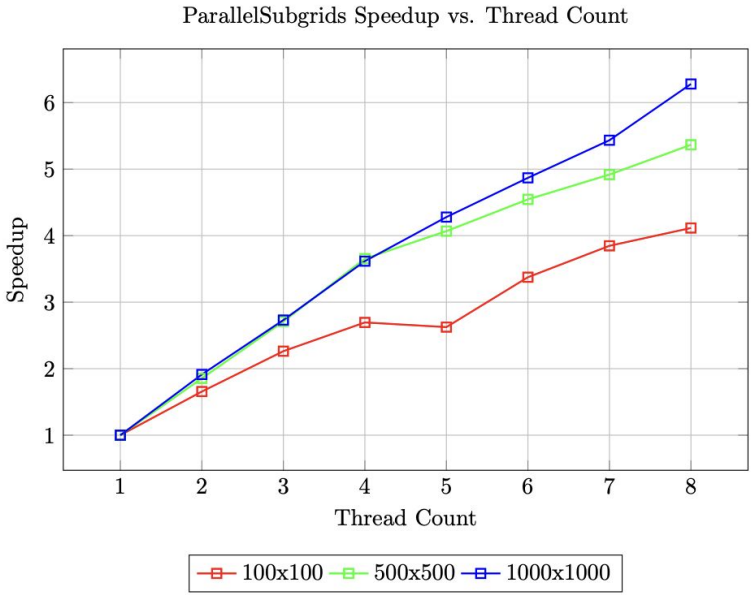
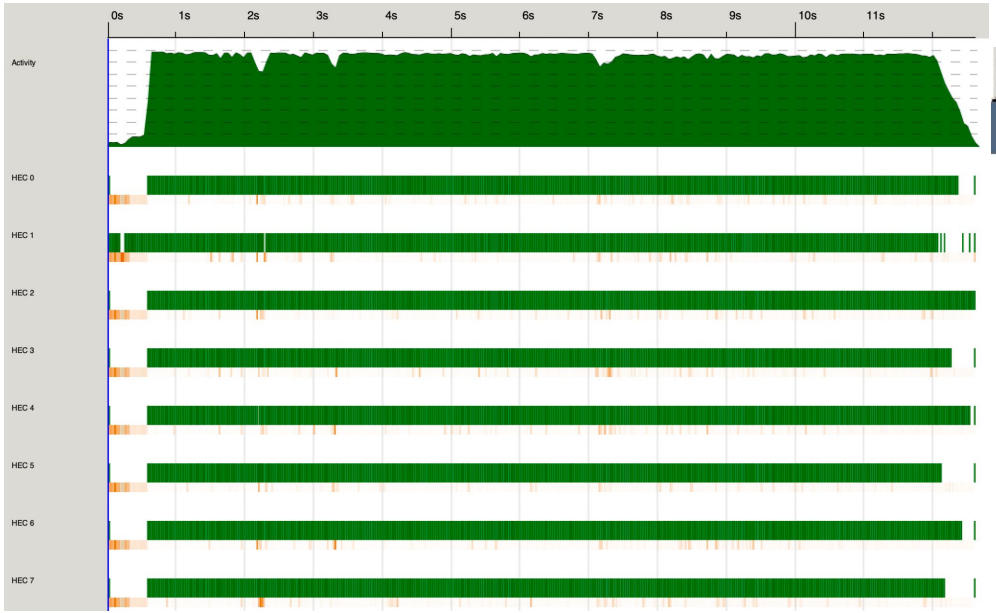


Figure 5: Speedup for varying thread counts for different board sizes, each split into 196 subgrids.

ParallelSubgrids Results



HEC	Total	Converted	Overflowed	Dud	GCed	Fizzled
Total	196	195	0	0	0	1
HEC 0	0	25	0	0	0	0
HEC 1	196	22	0	0	0	0
HEC 2	0	25	0	0	0	1
HEC 3	0	24	0	0	0	0
HEC 4	0	25	0	0	0	0
HEC 5	0	25	0	0	0	0
HEC 6	0	25	0	0	0	0
HEC 7	0	24	0	0	0	0

ParallelSubgrids threadscope graph and spark stats for 1000x1000 board, 196 subgrids, -N8.

Conclusion

- The Word-Search Sequential algorithm was a good candidate for parallelization.
- ParallelWords is a poor method parallelism
- ParallelDepth and ParallelSubgrids show significant performance increases

Future Work

- Test performance on machine with high hardware thread count
- Tune test cases to get more granular performance results of our algorithms given our current hardware setup
- Investigate if there are other algorithms that could be used for more efficient parallelism