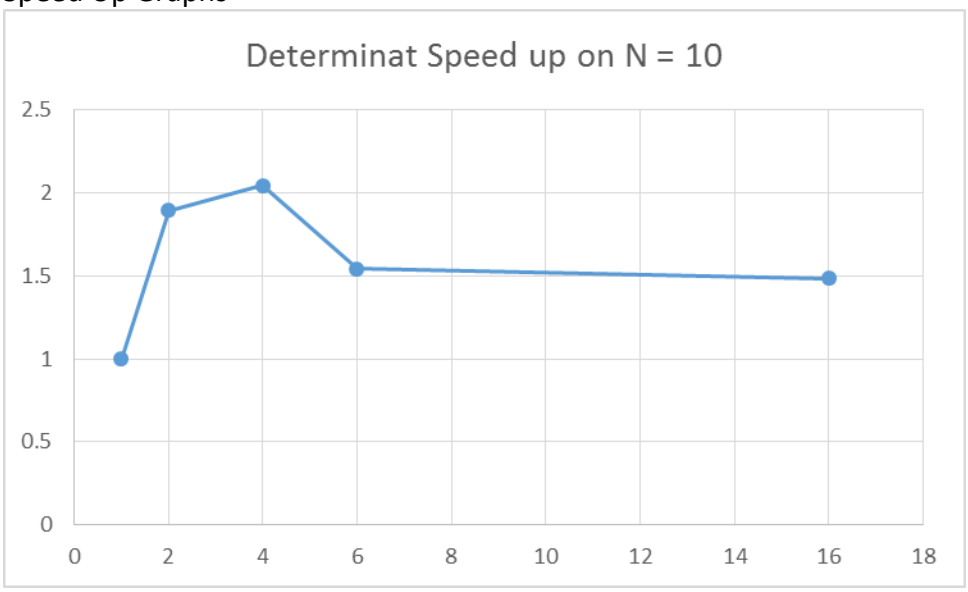
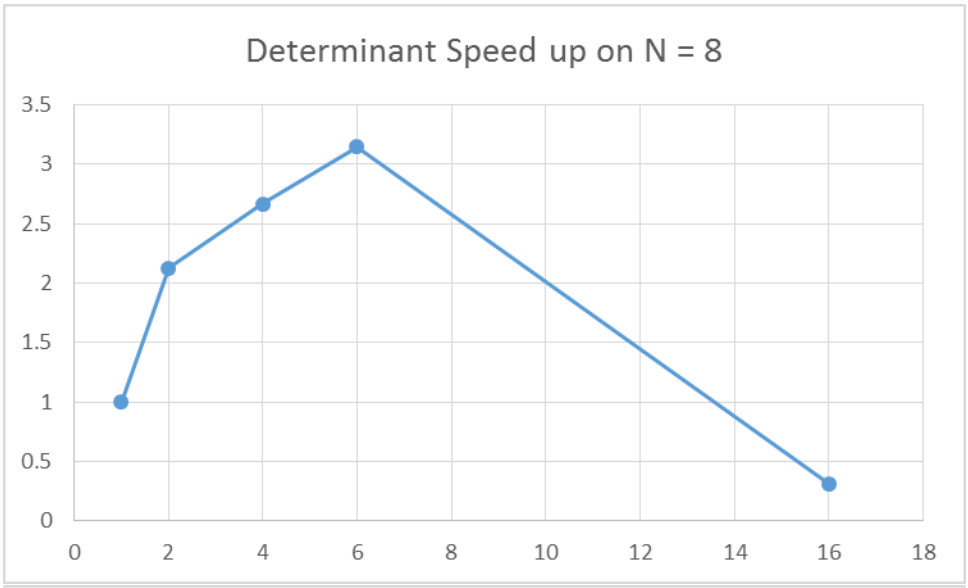
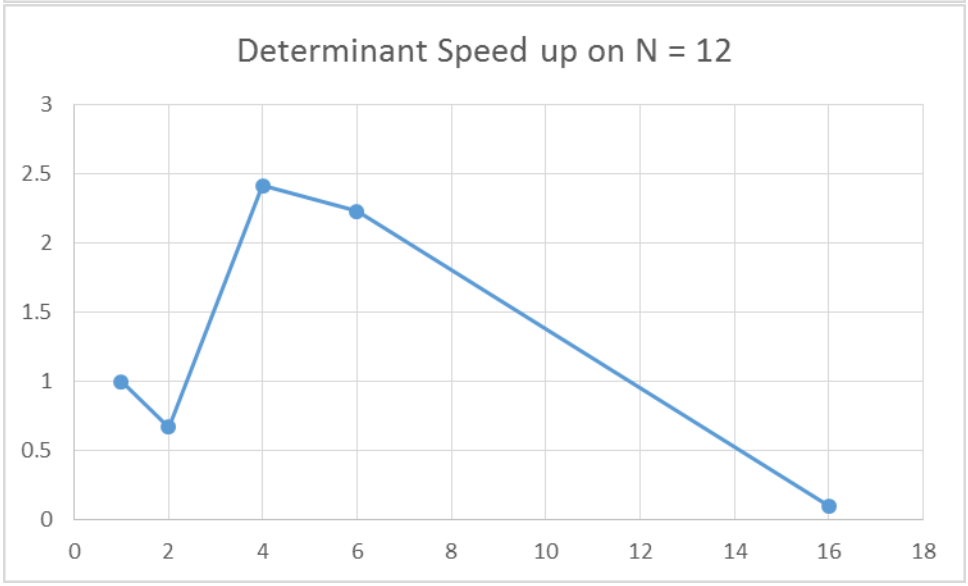
Report on the Parallelization of Determinant Calculation and Shear Sort
Introduction to High Performance Computing CS 1645
Brian Lester bdl20

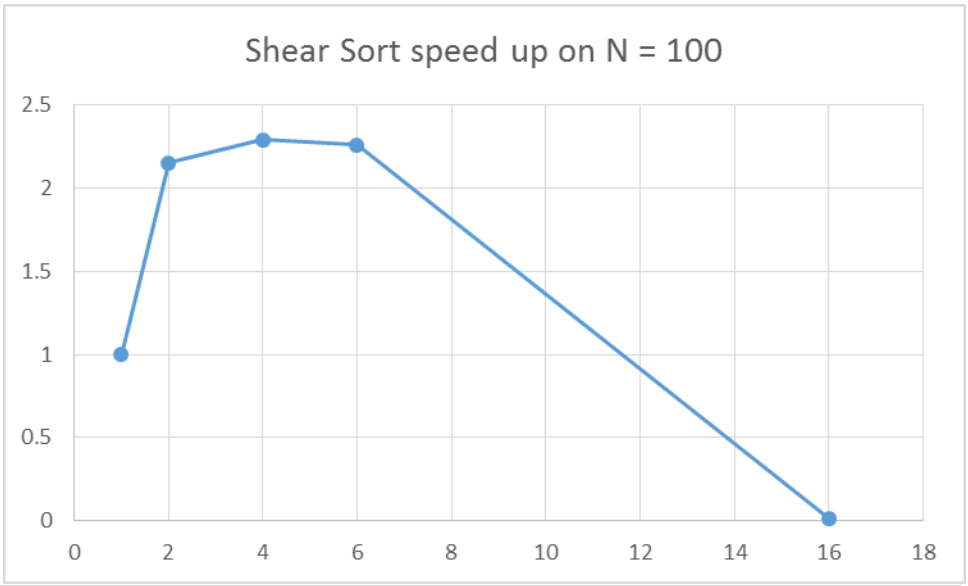
1) Parallelization strategies for each problem.
   Determinant Calculation – Fork and Join was used. A matrix's minors can all be computed in parallel and the determinant of each can be computed. These results (determinant of each minor) are saved into an array then the final determinant is calculated from that array with the original formula. The array to save the minor determinants need to be shared so each thread can write into its own section of the array.
   Shear Sort – Each row can be sorted independently as can each column but the columns can only be sorted after the rows are so fork and join is used for the rows and column but they are separate. First there is a fork and each thread sorts a row then they join and then each thread sorts a column. This forced ordering is why each for loop is parallelized rather than the whole thing being parallelized. Originally the sort functions were also parallelized but with the tests I ran that introduced to much parallelism overhead and slowed it down.

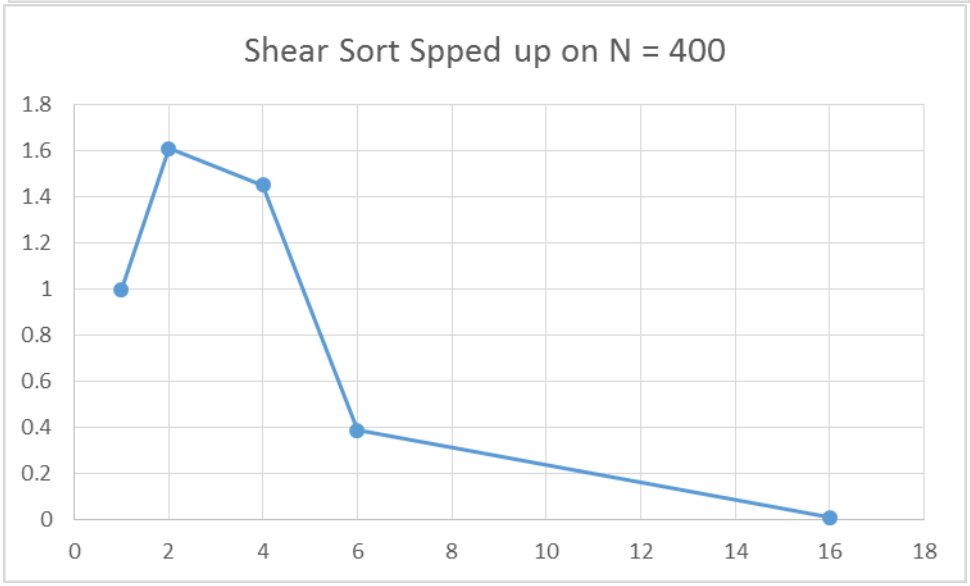
2) Speed Up Graphs

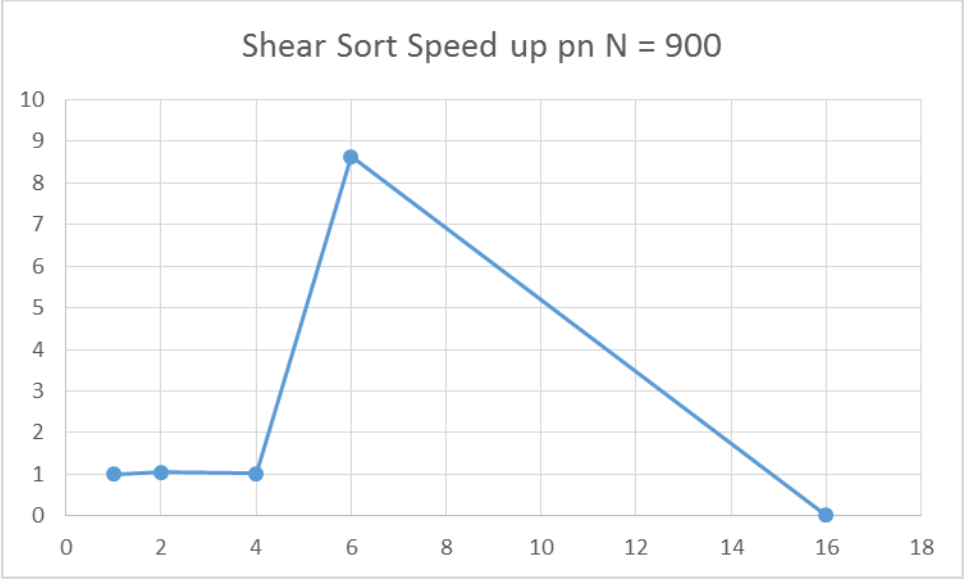



Determinant Speed up on N = 8
3.5
3
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
16
18

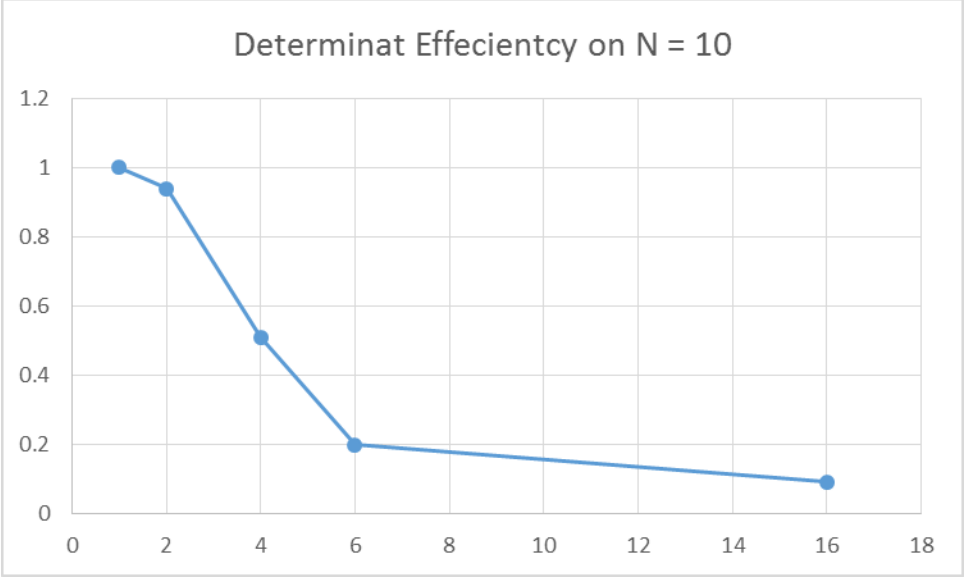

Determinant Speed up on N = 12
3
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
16
18

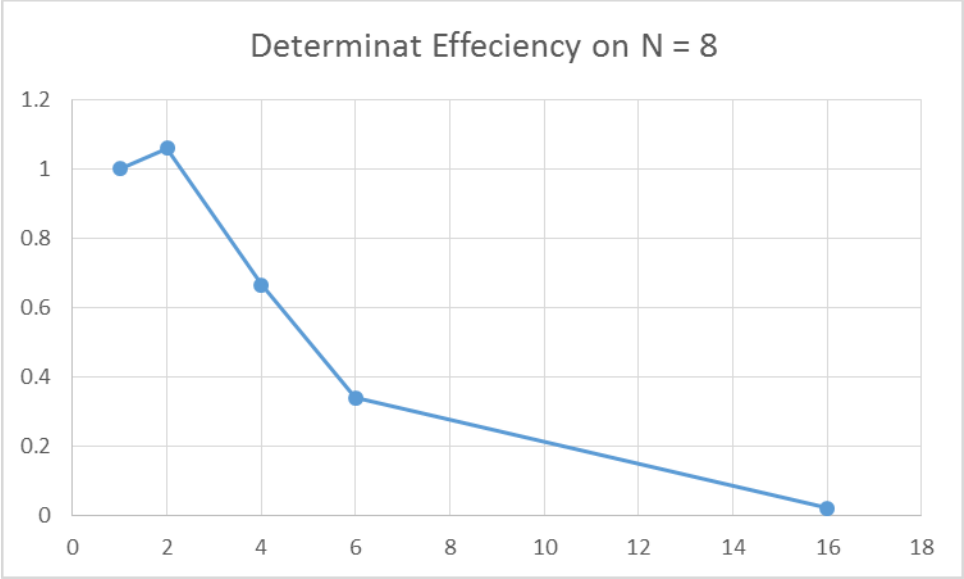

Shear Sort speed up on N = 100
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
16
18

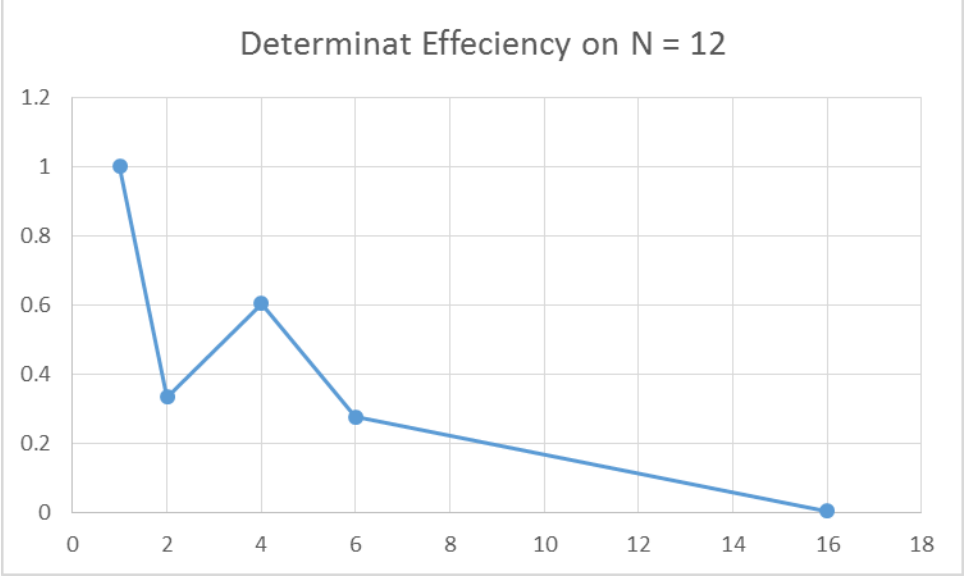

Shear Sort Spped up on N = 400
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18

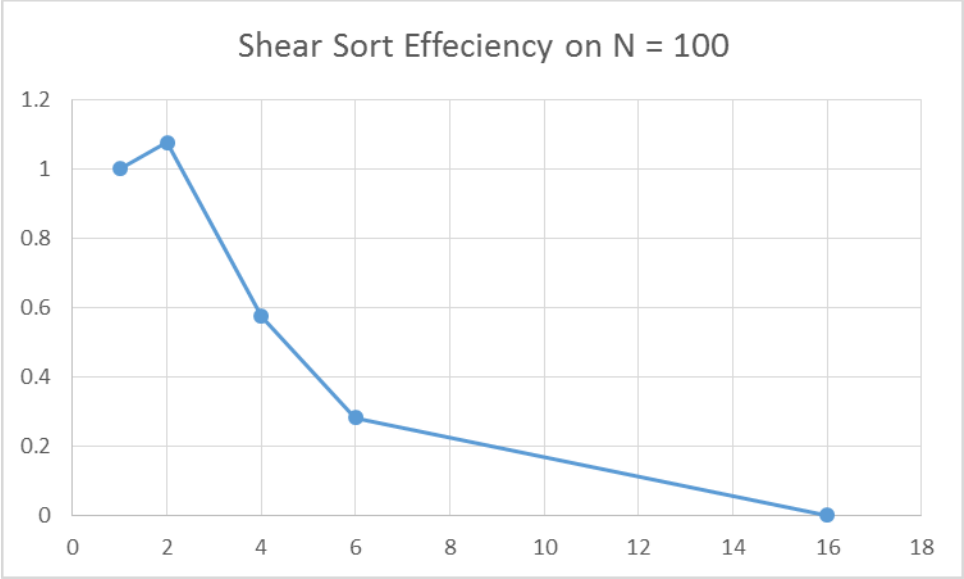


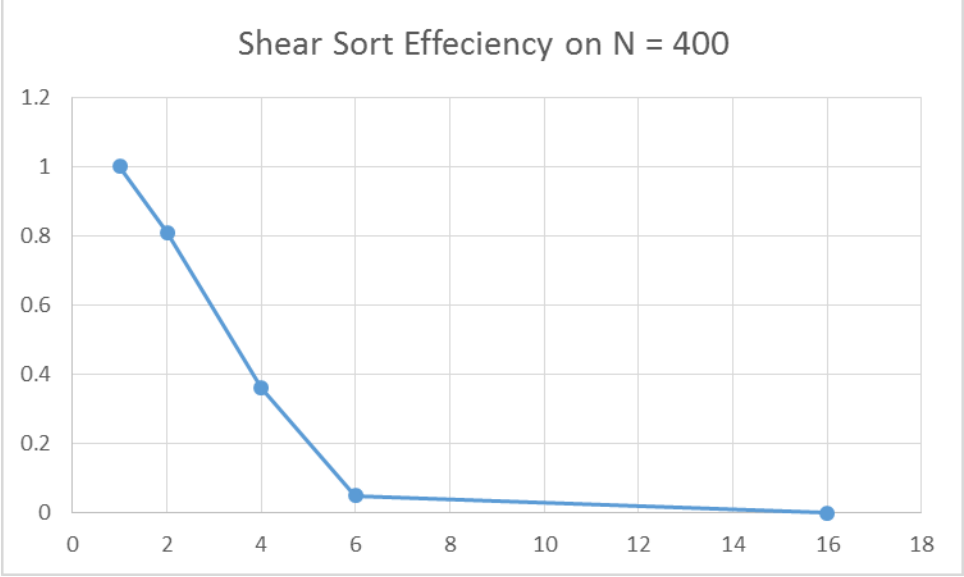
3) Efficiency Analysis Graphs:

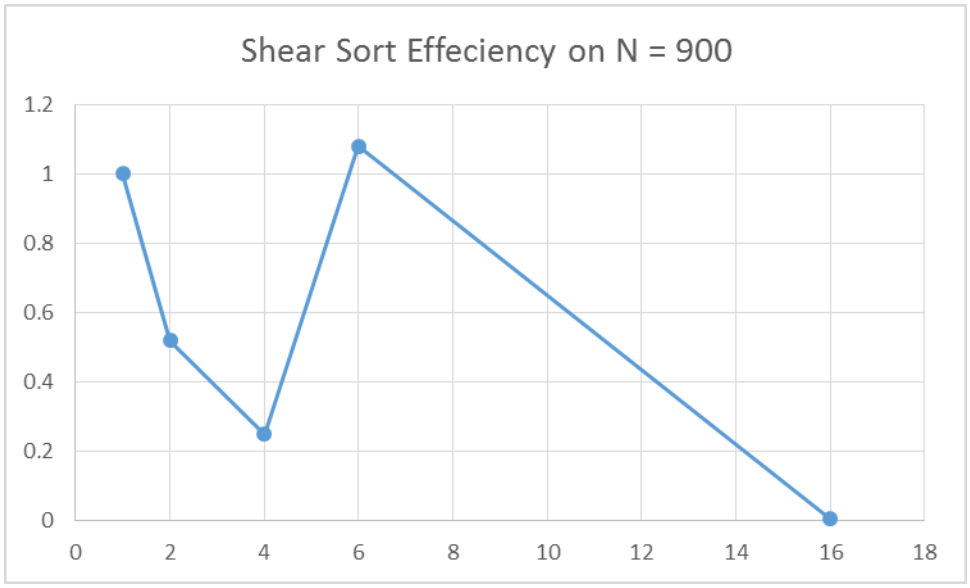



Determinat Effeciency on N = 8
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18


Determinat Effeciency on N = 12
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18


Shear Sort Effeciency on N = 100
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18


Shear Sort Effeciency on N = 400
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
14
16
18



4) Determinant Calculation is bottlenecked by then inability to parallelize the recursive part of the algorithm. If you could have parallelize this the speed up would increase as the miner's miners determinates are found in parallel
Shear Sort is bottleneck by to much overhead. This could maybe with bigger sample sizes you could parallelize the sorting functions and get a faster program.