Csc2002s Assignment 1

Parallel Programming with the Java Fork/Join framework :Terrain Classification

Introduction :

The aim of the project is to find the Basins (local minimum) from scanned input terrain ,using parallel programming .

```
1     if((hi-lo) < SEQUENTIAL_CUTOFF) {
2           int ans = 0 ;
3
4           for(int i=lo;i<hi;i++)
5              if( new Terrain().gridPointsChecker(new
   Terrain(arr[i].gridPoint,arr[i].row,arr[i].column)) )
6                 ans ++;
7
8           return ans;
9        }
10       else {
11          BasinArray left = new BasinArray(arr,lo,(hi+lo)/2);
12          BasinArray right= new BasinArray(arr,(hi+lo)/2,hi);
13
14          left.fork();
15          int rightAns = right.compute();
16          int leftAns  = left.join();
17          return leftAns + rightAns;
18
19       }
```

The theoretical maximum speedup using parallel computing would be 20 times.

Methods :

To approach the solution,I used Java fork-Join framework to parallelize the terrain classification using a divide-and-conquer algorithm as required . Used this algorithm to take break down an Array of Inner points (points that are not on the outer edge of the grid) hence solving the problem in a parallel programming.

Comparing my parallel approach against Serial approach .This allowed me to validate my algorithm. Parallel program needs to be both **correct** and **faster** than the serial versions.

Execution time of Parallel vs Execution time of Serial across different Data sizes and Sequetial Cutoffs . (Time in seconds for 20 Runs)

Sequential_Cutoff :500 ;

Small(Parallel)	Small(Serial)		Medium(Parallel)	Medium(serial)		Large(Parallel)	Large(Serial)
0.026	0.04		0.044	0.099		0.111	0.126
0.008	0.023		0.027	0.017		0.092	0.094
0.005	0.018		0.025	0.019		0.096	0.039
0.006	0.002		0.03	0.012		0.073	0.008
0.003	0.014		0.033	0.028		0.033	0.011
0.005	0.012		0.025	0.004		0.117	0.01
0	0.008		0.021	0.007		0.062	0.011
0	0.012		0.021	0.006		0.047	0.015
0	0.004		0.018	0.006		0.065	0.01
0.001	0.001		0.002	0.008		0.082	0.016
0	0.001		0.002	0.006		0.016	0.014
0	0.001		0.005	0.005		0.106	0.01
0.001	0.001		0.003	0.006		0.015	0.013
0.001	0.001		0.001	0.005		0.01	0.016
0.005	0		0.002	0.004		0.012	0.01
0.001	0.001		0.001	0.003		0.012	0.016
0	0.001		0.001	0.006		0.012	0.013
0	0		0.007	0.004		0.013	0.014
0	0.001		0.003	0.003		0.01	0.011
0	0.001		0.003	0.003		0.007	0.014

Average Time for Parallel: (0.00295+0.01305+0.04719)/3 =0.0211 sec
Average Time for Serial:(0.00676+0.01195+0.04281)/3 =0.0201 sec

Parallel
200X200 ; average time 0.00295 sec
500X500 ; average time 0.01305 sec

1000X1000; average time   0.04719 sec
Serial
200X200 ; average time0.00676 sec
500X500 ; average time 0.01195 sec
1000X1000; average time   0.04281 sec

Sequential_Cutoff :1000;

small(Parallel)	small(Serial)		Medium(Parallel)	medium(Serial)		Large(Parallel)	Large(Serial)
0.024	0.058		0.04	0.067		0.12	0.112
0.008	0.049		0.027	0.03		0.061	0.078
0.006	0.026		0.02	0.014		0.085	0.035
0.0019	0.015		0.003	0.016		0.113	0.011
0.02	0.011		0.004	0.005		0.098	0.008
0.013	0.006		0.002	0.002		0.083	0.012
0.007	0.008		0.006	0.002		0.084	0.008
0.014	0.006		0.004	0.003		0.058	0.015
0.006	0.001		0.002	0.004		0.038	0.012
0.015	0.001		0.002	0.003		0.044	0.009
0	0		0.002	0.004		0.007	0.014
0	0		0.002	0.003		0.009	0.015
0	0		0.007	0.002		0.005	0.011
0.001	0		0.004	0.003		0.005	0.013
0	0		0.002	0.004		0.009	0.014
0	0.001		0.002	0.002		0.006	0.012
0.002	0.001		0.001	0.002		0.008	0.014
0	0.001		0.007	0.002		0.004	0.015
0.002	0.001		0.003	0.002		0.005	0.019
0.001	0.001		0.001	0.005		0.008	0.018

Average Time for Parallel:(0.00604+0.0071+0.0425)/3 =0.0185 sec
Average Time for Serial :(0.0093+0.0088+0.0223)/3 = 0.0135 sec

Parallel
200X200 ; average time 0.00604 sec
500X500 ; average time 0.0071 sec

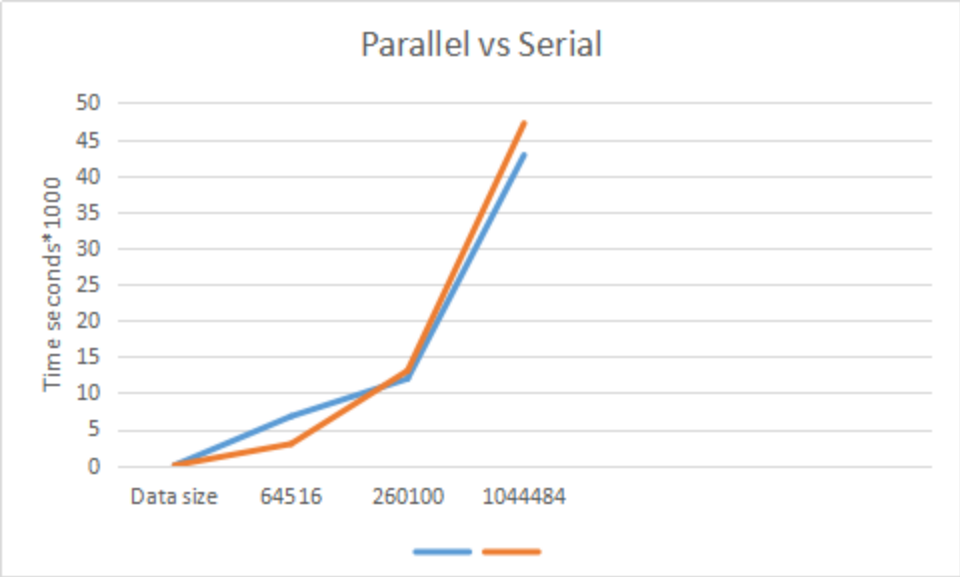
1000X1000; average time   0.0425 sec
Serial
200X200 ; average time 0.0093 sec
500X500 ; average time 0.0088 sec
1000X1000; average time 0.0223 sec

Speedup = (serial time + parallel time)/number of core :
(0.0201+0.0135)+(0.0211+0.0185)/2 = 0.0366

My machine has 2 cores , On average runs about 240 processes and 2500 Threads .

problems/difficulties :
-The larger the input size the longer the execution time of the program.(hence waiting for the outputs)
- Slowing down other running programs

Results And Discussion :



Parallelization increase the speed of excution .
The data size of greater 1000 does perform well for parallelization

Maximum speedup obtained 0.0366 seconds;

The Sequential-cutoff of 1000 is optimal for this problem.

Conclusions :

Parallel program is best ideal for problems with larger datasets .
My results tell me that using parallel program is best for larger data , and depending also on the Sequential cutoff. The speedup depends on percentage of parallel algorithm and sequential algorithm in my program .