Partial Exam Report

Candidate: 15243

April 3, 2019

1 Main data structure and functions

The main data structure of the program is based on dynamically allocated arrays using the library *malloc*-function, which again allows more flexible management of memory, which lets us utilize the data in the arrays in multiple functions. The contiguous memory allocation provided by the *malloc* also helps data decomposition for the different threads when running the parallel regions.

The program has three functions outside the main-function. The first of these is the *read_graph_file*-function, whose purpose is to extract the necessary data from the input file. The function initially allocates arrays to store the *FromNode*-data and the *ToNode*-data, and then reads through the files to initialize the arrays with the corresponding values. The file is then closed, and we enter the first of two nested loops necessary to build the hyperlink matrix. The first nested loop has three purposes:

1. Collect the total number of outbound links from each website
2. Count the total number of outbound links
3. Count the total number of websites that have no outbound links

Since we can not assume that the input files are ordered, we need to loop through all the links for each website. Since this loop is only counting elements, we are able to parallelize it using OpenMP. The function then proceeds to check whether there are any dangling webpages (websites with no outbound links). If yes, the array is allocated to the appropriate size, and the indices of the dangling webpages are collected in the array. After having collected the total number of outbound links (nonzero elements in the hyperlink matrix), the function now allocates the arrays which will be used to store the data in the hyperlink matrix in CRS (Compressed Row Storage) format. The function then enters the second nested loop, which initializes the CRS-arrays with their appropriate values. Finally the arrays with the FromNode, ToNode and number of outbound links-data are freed, as they are no longer needed.

The second function in the program is *PageRank_iterations*, which calculates the PageRank scores for each website. This function starts by declaring the necessary variables that are shared between the different threads. This includes the *one_over_N*-variable, which is calculated once to avoid division in the following loops. The function then enters a parallel region, where the arrays $\mathbf{x}_0$ and $\mathbf{x}_1$ are initialized in a parallelized loop. Assuming the runtime library supports the **first touch**-policy, this assures us that the memory pages of the arrays will be mapped on

to the appropriate locality domain of each thread. The function then enters a *while-loop*, which runs as long as the pre-declared variable *flag* is zero. A single thread then sets the value of W_{sum} to 0, as this needs to be initialized for each iteration. The *pragma omp single*-command has an implicit barrier, which assures us that no thread will continue with the wrong value of W_{sum}. The threads then calculate the new W_{sum}-values using the *reduction*-clause, before continuing to a parallelized *for-loop*, in which the result of *sparse matrix-vector* multiplication ($A\mathbf{x}_{k-1}$) is calculated, and multiplied by the damping factor d. A single thread then calculates the value $\frac{1-d+d\cdot W^{k-1}}{N}$ (which is a shared value), to avoid each thread having to calculate this value for each iteration of the coming loop. The function continues to a parallelized loop which calculates each value in the $\mathbf{x}_k$-array, before entering another loop to find the largest difference between each element in $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$. Then a single thread swaps the pointers (arrays), such that $\mathbf{x}_k$ is now $\mathbf{x}_{k-1}$, and vice-versa. Finally, each thread checks whether their maximum difference values is smaller than the provided threshold convergence value. If yes, the flag-variable is set to 1, which will cause the while-loop to end. An explicit barrier is required at the end of the while-loop, as we don't want threads to start new iterations before all threads have checked their maximum difference, since we only need one thread to change *flag* to 1.

The final function in the program is called *top_n_pages*, and has the purpose of collecting the n webpages with the top PageRank-scores. This is done by going through the PageRank-values n times, finding the biggest value which is also smaller than the previously found biggest value. I.e. a value M_j such that $x_i < M_j < M_{j-1}$ for all $i \in [0, nodes]$ and $j \in [0, n]$. The values and the corresponding webpage-index is then stored in arrays.

2 Timing

The program was run with three varying number of threads: 4, 2, and 1, using an input file with total number of webpages (nodes) 325729 and number of links (edges) 1497134. The resulting timing of the *pagerank_iterations*-function is as follows:

- 4 threads: 245.43ms
- 2 threads: 481.75ms
- 1 thread: 970.22ms

Doubling the amount of threads resulted in a speedup of ≈ 2 for both cases. See (4) for the full outputs and timing of all three functions.

3 Hardware and Compiler

The program was compiled using GCC, and run on a computer with the relevant hardware:

- Memory: 2x8GB DDR4 2133MHz
- CPU: Intel(R) Core(TM) i5-6600K Skylake
  - Processor Base Frequency: 3.50GHz
  - Cache: 6MB

- No. of cores/threads: 4
- Hyper-threading: No.

4 Examples with Varying Thread Numbers

4 Threads:

```
$ ./a.out web-NotreDame.txt 0.85 1e-10 25

Provided arguments:
Damping factor = 0.850000, epsilon = 0.000010e-5, n = 25

Readfile time: 1482.019096s

Total number of PageRank iterations: 71

PageRank calculation time: 0.245426s = 245.426300ms

Top pages time: 0.019012s = 19.011800ms

Position: | PageRank Score: | Page index:
 ------------------------------
        1 | 0.005626        | 1963
        2 | 0.005404        | 0
        3 | 0.003326        | 124802
        4 | 0.002857        | 12129
        5 | 0.002749        | 191267
        6 | 0.002732        | 32830
        7 | 0.002590        | 3451
        8 | 0.002460        | 83606
        9 | 0.002376        | 1973
        10 | 0.002340        | 142732
        11 | 0.002234        | 24823
        12 | 0.002151        | 143218
        13 | 0.001827        | 31331
        14 | 0.001802        | 149039
        15 | 0.001476        | 140170
        16 | 0.001403        | 12838
        17 | 0.001391        | 81878
        18 | 0.001169        | 226950
        19 | 0.001110        | 73859
        20 | 0.001101        | 292009
        21 | 0.000992        | 63364
        22 | 0.000990        | 24944
        23 | 0.000962        | 88448
        24 | 0.000947        | 88118
        25 | 0.000878        | 10331
```

2 Threads:

```
$ ./a.out web-NotreDame.txt 0.85 1e-10 25

Provided arguments:
Damping factor = 0.850000, epsilon = 0.000010e-5, n = 25

Readfile time: 1815.939312s

Total number of PageRank iterations: 81

PageRank calculation time: 0.481744s = 481.744500ms

Top pages time: 0.019571s = 19.571200ms

Position: | PageRank Score: | Page index:
 ------------------------------
        1 | 0.005627        | 1963
        2 | 0.005405        | 0
        3 | 0.003326        | 124802
        4 | 0.002857        | 12129
        5 | 0.002749        | 191267
        6 | 0.002732        | 32830
        7 | 0.002590        | 3451
        8 | 0.002460        | 83606
        9 | 0.002376        | 1973
        10 | 0.002341        | 142732
        11 | 0.002234        | 24823
        12 | 0.002152        | 143218
        13 | 0.001827        | 31331
        14 | 0.001802        | 149039
        15 | 0.001477        | 140170
        16 | 0.001403        | 12838
        17 | 0.001391        | 81878
        18 | 0.001169        | 226950
        19 | 0.001110        | 73859
        20 | 0.001101        | 292009
        21 | 0.000992        | 63364
        22 | 0.000990        | 24944
        23 | 0.000962        | 88448
        24 | 0.000947        | 88118
        25 | 0.000878        | 10331
```

1 Thread:

```
$ ./a.out web-NotreDame.txt 0.85 1e-10 25

Provided arguments:
Damping factor = 0.850000, epsilon = 0.000010e-5, n = 25
```

```
Readfile time: 2424.786803s

Total number of PageRank iterations: 89

PageRank calculation time: 0.970226s = 970.225700ms

Top pages time: 0.019393s = 19.393300ms

Position: | PageRank Score: | Page index:
 ------------------------------
        1 | 0.005627        | 1963
        2 | 0.005405        | 0
        3 | 0.003326        | 124802
        4 | 0.002857        | 12129
        5 | 0.002749        | 191267
        6 | 0.002732        | 32830
        7 | 0.002590        | 3451
        8 | 0.002460        | 83606
        9 | 0.002376        | 1973
        10 | 0.002341        | 142732
        11 | 0.002234        | 24823
        12 | 0.002152        | 143218
        13 | 0.001827        | 31331
        14 | 0.001802        | 149039
        15 | 0.001477        | 140170
        16 | 0.001403        | 12838
        17 | 0.001391        | 81878
        18 | 0.001169        | 226950
        19 | 0.001110        | 73859
        20 | 0.001101        | 292009
        21 | 0.000992        | 63364
        22 | 0.000990        | 24944
        23 | 0.000962        | 88448
        24 | 0.000947        | 88118
        25 | 0.000878        | 10331
```