NSM GPU Programming Assignment 3 Deadline : December 13, 2023, 23:55

1 Problem Statement

Implement a parallel PageRank in CUDA. PageRank of a node pi in a graph is given by the following formula.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Figure 1: PR formula

Here p1, p2, ..., pN are the nodes, M(pi) is the set of nodes that link to pi, L(pj) is the outdegree of node pj, and N is the total number of nodes in the graph. PR(pj) should be the old value of pagerank of node pj. Number of iterations of your pagerank implementation must be exactly 10 (i.e., the pagerank kernel should get invoked 100 times).

* initialization of PR(pi) = 1/N.
* d = 0.85

2 Input and Output

2.1 Input

* Input to your program would be a single argument mentioning path to the input graph in .txt format as given.
* Input will be large graph datatsets in .txt format

2.2 Output

* Output of PageRank program should be N lines (N is the number of graph nodes) where ith line contains the PageRank value computed for the ith node. Print each PageRank value with nine decimal digits.
* Output must be in .txt file

3 Points to be noted

* We have given all the implementation except the CUDA part like kernel call, cudamemcpy etc. You need to write the corresponding code just below the commented section.
* We have given a helper file **helper.hpp** which will keep track of CSR representation of the graph along with number of edges and number of nodes in a graph.

* We have called CSR things like **offset array** and **neigbourlist array** in the main. You just need to pass it appropriately to kernel and complete the kernel part.
* Do not write any print statements inside the kernel.
* Don't write code having race condition.
* You can use global synchronization also.
* Do not upload anything other than the **full_name.zip** file. The zip file will consist of only helper.cu along with **your_name.cu** file.

4 Submission Guidelines

* Submit your file with your **full_name.zip** which contains the implementation of the above-described functionality
* Don't modify or change helper.hpp file
* Don't use anything other than what is given in the .cu file Follow the naming conventions given for **CSR representation** of the graph.
* Don't change the function prototype, function call, kernel argument and kernel call.
* You only need to write the code just below the commented section
* After submission, download the file and make sure it was the one you intended to submit.
* Kindly adhere strictly to the above guidelines.

5 Learning Suggestions

* Write a CPU-version of code achieving the same functionality. Time the CPU code and GPU code separately for large matrices and compare the performances.
* Usage of global synchronisation