Dear Prathamesh,

Below your task for the coding challenge.

It is basically related to our activity with Arm.

The task is writing a parallel code by using openmp.

You have to implement a top down BFS by exploiting a linear algebraic approach which relies Sparse Matrix Vector Multiplication.

The code should be able to read snap graphs (Googling for it) and use graph500 for validation (optional but I strongly recommend to use).

Write a small ( 1 page roughly ) report to describe the task, the related work and the performance that you get by increasing the number of threads (strong scaling). Use Traverlas Edge per second as metric for the benchmark. Cite other codes or peace of code if you get from others.

This task should assess your research skills (search for relevant paper), coding skills ( implementation with openmp) and communication skills.

Optional tasks:

1) write a bottom up BFS implementation.

2) Compare with other existing  implementations or a traditional vertex centric approach.

This two tasks are not mandatory.

Please send us a git repo of your code with the instructions for the complication and for reproducibility of your results by 18th of August.

For further information or if something is note clear feel free to ask.

Regards and good luck.

Flavio and Filippo

**Task:** To implement top down BFS exploiting linear algebraic approach based on SpMV using SNAP dataset and benchmarking at Graph500.

Here, basic difference between traditional BFS and BFS using linear algebra is the use of data structure. Trad-BFS uses queue to traverse where as sparse matrix can be stored in compressed sparse row ( CSR). This requires arrays to store value, indices and respective pointers.

**Literature Review:** There are different implementations of it. Most recent is by Paul Burkhardt[1] about the optimizing the BFS for sparse graphs. They have claimed complexity of O(n) for BFS computation. In [2], emphasis is given on the reducing number of matrix operations using associative, cumulative properties for large graphs. SlimSell by Maciej Besta[3] is vectorizable graph representation BFS using SpMV and claimed leveraging the storage by up to 50% using Sell-C -σ matrix storage format. There is an enormous work by Aydın Buluç and Kamesh Madduri on sparse matrix multiplication on distributed systems, multithreading and graph partitioning[4][5]. In a view of analyzing robustness and concrete contribution [6] put forth GPU based BFS SpMSpV to gauge the speed up.

**Things that I could do:** I feel sorry to say that I could not get the desired result. Program for SpMV multiplication and parallizing it using OpenMP is implemented but could not adhere to the BFS algorithm. Few codes from Github helped to have better understanding. SNAP dataset is well structured and codes from it’s repository were successfully implemented. However, program specific dataset files could not be read. I have to dig out more for SNAP. Graph500 has asserted the benchmark specifications.

REFERENCES

1. Paul Burkhardt, “Optimal algebraic Breadth-First Search for sparse graphs”.

<https://arxiv.org/pdf/1906.03113.pdf>

2. Kepner et al., “Mathematical Foundations of the GraphBLAS”.

<https://people.eecs.berkeley.edu/~aydin/GraphBLAS-Math.pdf>

3. Maciej Besta et al. “SlimSell: A Vectorizable Graph Representation for Breadth-First

Search, the 31st IEEE International Parallel & Distributed Processing Symposium

(IPDPS'17).

4. A. Buluc and K. Madduri. “Graph partitioning for scalable distributed graph computations”

Graph Partitioning and Graph Clustering, 588:83, 2013.

5. Aydin Buluç, Kamesh Madduri, Parallel breadth-first search on distributed memory

systems. SC 2011: 65:1-65:12

6. Carl Yang, Yangzihao Wang, and John D. Owens, Fast Sparse Matrix and Sparse Vector

Multiplication Algorithm on the GPU, IEEE International Parallel and Distributed

Processing Symposium Workshop 2015.