CmpE 478 Homework 2

Dilara Gökay & Ece Ata

28 May 2019

1 Part A - Implementing PageRank using MPI

Message Passing Interface (MPI) is a specification for the developers and users of message passing libraries. MPI is designed for distributed memory architectures. Hence, we needed to partition the given web graph using Metis partitioner. Metis works for undirected graphs. Therefore, we made the graph undirected, partitioned this graph, then applied PageRank algorithm to these partitions. We sent each partition to different slaves. We used BoostMPI. Please note that our implementation might not work for very large graphs. Please change the file name in main.cpp according to the location of the graph.txt in your local machine.

```
1  $ python make_undirected.py
2  $ mpic++ main.cpp -o main -L/usr/local/lib -I/usr/local/include/ -
       lboost_mpi -lboost_serialization -lmetis -std=c++11
3  $ mpirun --oversubscribe -np 4 main
```

2 Part B - Implementing PageRank using Thrust

Thrust is a C++ template library for CUDA based on the Standard Template Library (STL). Thrust allows you to implement high performance parallel applications with minimal programming effort through a high-level interface that is fully interoperable with CUDA C.

```
1  $ nvcc -o a.out .\pagerank_thurst.cu -ccbin C:\Program Files (x86)
       \Microsoft Visual Studio 14.0\VC\bin
2  $ .\a.out
```

3 CSR Matrix Storage Format

First part of the project was to convert the given graph in `graph.txt` file into compressed sparse row (CSR) format. We know that the each line of `graph.txt` represents an edge and the first token in a line is the from-vertex and the second token in a line is the to-vertex.

3.1 Assigning index to to-vertices

We decided to assign indices to all to-vertices, then from-vertices so that we can fill the first $|to\text{-}vertices|$ rows of the matrix and the rest of the rows are filled with all zeros. Please note that we did not use any "matrix" to store the graph data but this decision made implementing CSR format easy as well. In CSR format, this implementation decision corresponds to inserting the last value of `row_begin` to the end of `row_begin` for $|all\ from\text{-}vertices \setminus all\ to\text{-}vertices|$ times.

3.2 Constructing CSR vectors

- For each to-vertex (row in the matrix), we inserted the corresponding $|from\text{-}vertices|$ (number of in-edges) to `row_begin`.
- We assigned index to the remaining $|from\text{-}vertices \setminus to\text{-}vertices|$ vertices. For each to-vertex, we inserted the index value of its corresponding from-vertices to `col_indices`.
- We know that all edges in the graph are unique. Therefore, for each from-vertex (column), the sum of values add up to 1.

4 Google Ranking Process

We made matrix multiplication in parallel. The region where we calculate r^t is distributed among processors.

5 Timings

Result is generated in output.csv for Thrust.

Operation	Timing (s)
I/O	65.253
PageRank	28.486

6 Top 5 hosts

For Thrust,

Rank of Hosts	Host Name
1	4mekp13kca78a3hfsrb0k813n9
2	4mekp13kca78a3hfsrb0k813n9
3	3165mii1s1g0invqs94q303v0v
4	46o3c5beh6kiojkvr1tvsk4ptt
5	2494c7mt12frm3c3go86abe13h

7 Result

The algorithm works faster when Thrust is used. This is due to high level of parallelism with the use of GPUs.

8 System Details

Part A:

- Computer: MacBook Pro (Retina, 13-inch, Early 2015)
- OS: macOS Mojave Version 10.14.4
- Processor: 2.7 GHz Intel Core i5
- GPU: -

Part B:

- Computer: MONSTER ABRA A5 V1
- OS: Microsoft Windows 10 Home Version 10.0.17134
- Processor: Intel Core i7-4710MQ
- GPU: nVIDIA GeForce GTX860M