CS542200 Parallel Programming
Homework 4-2: Blocked All-Pairs Shortest Path (Multi-cards)

Due: December 30, 2019 11:59 a.m.

1 Goal

This assignment helps you get familiar with multi-cards CUDA programming by implementing a blocked all-pairs shortest path algorithm. Besides, to measure the performance and scalability of your program, experiments are required. Finally, we encourage you to optimize your program by exploring different optimizing strategies for optimization points.

2 Problem Description

Please refer to the homework 4-1.

3 Input / Output Format

Please refer to the homework 4-1

Notice that:

- ✓ $2 \leq V \leq 60000$ for single GPU version
- ✓ $0 \leq E \leq V \times (V - 1)$
- ✓ $0 \leq$ *source vertex id, destination vertex id* $< V$
- ✓ $0 \leq W \leq 100$
- ✓ No need to consider the condition of signed integer overflow

4 Working Items

You are required to implement blocked Floyd-Warshall algorithm under the given restrictions.

1. CUDA version blocked APSP algorithm with multiple cards (two cards only)
   - ➢ Same restriction with homework 4-1

- Only need to handle two cards in the same node
- Achieve better performance than single-card Floyd-Warshall implementation.
  Please analysis the data dependency, the total communication complexity should be V^2

2. Makefile
   Please refer to the example in **/home/pp19/share/hw4** on **hades01**
3. Report
   - Implementation
     - ✓ How do you divide your data?
     - ✓ How do you implement the communication?

     Briefly describe your implementation in diagrams, figures and sentences.
   - Experiment & Analysis

     We encourage you to show your results by figures, charts, and description.
     - ✓ System Spec

       Please attach CPU, GPU, RAM and disk information of the system.
     - ✓ Weak Scalability & Time Distribution

       Observe weak scalability of the mulit-GPU implementations.

       Moreover, analyze the time spent in

       1) computing

       2) communication

       3) memory copy (H2D, D2H)

       4) I/O of your program w.r.t. input size.

       You should explain how you measure these time in your programs and compare the time distribution under different configurations.

       ※ We encourage you to generate your own test cases!

※ You should not consider the number of vertex as problem size

- ✓ Others

  Additional charts with explanation and studies. The more, the better.

- ➢ Experience / Conclusion

  Note: The numbers in the following charts may not come from real data.
  Do NOT compare them with your own results!

5 Grading

1. Correctness (35%)
2. Performance (30%)
3. Report (20%)

   DO NOT put your charts without explanations. In such cases, we will not count these charts while grading your report.
4. Demo (15%)
5. Total Points $= \left((1) + (2) + (3) + (4)\right)$
6. We grade your scores by hw4b-judge (or hw4-2-judge) and hidden testcases, make sure your programs can get correct results on all possible testcases. We will grade the performance score by the largest testcase you can pass within 30 seconds. (Larger testcases, higher score)

6 Reminder

1. Please submit your code and report to ~/homework/hw4-2 on **both hades01** and **ilms** (DO NOT pack them):
   - ✓ hw4-2.cu
   - ✓ {student-ID}_report.pdf
   - ✓ Makefile

Note:

- ✓ Your Makefile must be able to build the corresponding targets of the implementations: hw4-2.
- ✓ Your submission time for your source code will be based on the time on hades01 and your submission time for your report will be based on iLMS. DO NOT touch the source code after the deadline.

2. Since we have limited resources for you to use, please start your work ASAP. Do not leave it until the last day!
3. Asking questions through iLMS is welcomed!