CS542200 Parallel Programming
Homework 3: Single source shortest path

Due: December 18, 2016

1.1 GOAL

This assignment helps you getting familiar with **Pthread** and knowing the differences between **synchronous** and **asynchronous** computing using a **fully distributed vertex centric graph computation** model. In this assignment, you need to implement single source shortest path in three versions:

1. Pthread
2. Fully distributed synchronous vertex-centric MPI
3. Fully distributed asynchronous vertex-centric MPI

2 PROBLEM DESCRIPTION

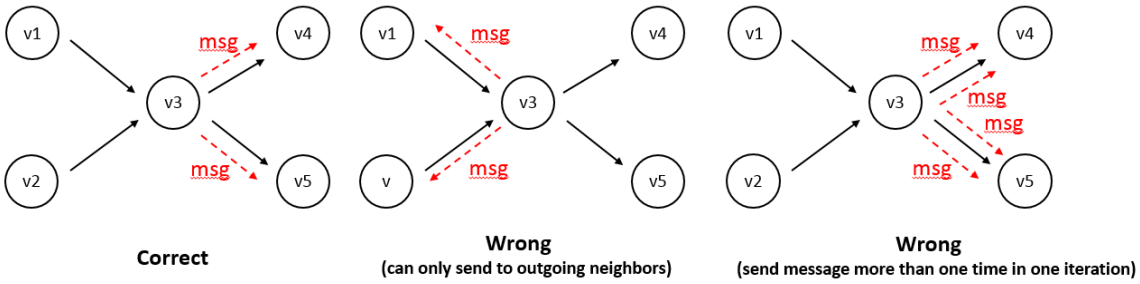
2.1 PTHREAD VERSION

- In this version, you are going to implement single source shortest path using Pthread. You may choose to implement any **algorithm without any limitation or constraints** as long as its **result is correct**, and **follows the input/output file format**.

2.2 FULLY DISTRIBUTED SYNCHRONOUS VERTEX-CENTRIC MPI

- In this version, you are going to implement single source shortest path using **synchronous** computing in **vertex centric** programming model.
- **Vertex centric** programming model means that during the computation, each MPI process controls a vertex and does local computation (ex: calculate the new shortest distance of its own vertex) and local communication (ex: only send messages to its outgoing neighbors).
- **Synchronous** computation means that each MPI process (vertex) can only communicate with each of its outgoing neighbors once per iteration! After each iteration, MPI_barrier() MUST be called in your code.

Example:



- The termination condition of this version is controlled by the network diameter (which is given in the input file) of the input graph. For instance, if the network diameter of the input graph is 3, you MUST run 3 iterations to ensure the final result is correct. You CANNOT use any other detection mechanism to reduce the number of iterations.

2.3 FULLY DISTRIBUTED ASYNCHRONOUS VERTEX-CENTRIC MPI

- In this version, you are going to implement single source shortest path using asynchronous computing in vertex centric programming model.
- The main difference between asynchronous and synchronous version is that MPI_barrier() will be removed from your code, and there CANNOT exist any synchronous point across all processes during the computations. In other words, each MPI process (vertex) can send arbitrary number of messages to its outgoing neighbors at any time.
- You MUST implement the Dual-Pass Ring termination algorithm to detect termination condition.

P.S.: For these two versions of MPI implementation, your computations must be fully distributed and follow the algorithm introduced in the lecture slides. However, you may use a single process to distribute the input and collect the output.

3 INPUT/OUTPUT

3.1 INPUT PARAMETERS:

./executable ${1} ${2} ${3} ${4}

- ${1}: number of threads [integer]

  (P.S For MPI version, this parameter will not be used, but please give it a number for maintaining the same input format)

- ${2}: the input file name [string]

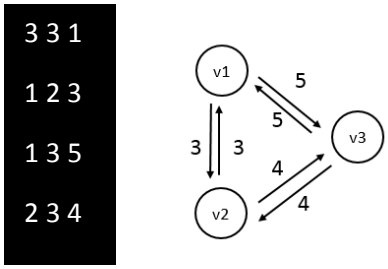
- ${3}: the output file name [string]

- ${4}: the id of source vertex [integer]

3.2 INPUT FILE FORMAT:

The input graph will be a connected weighted directed graph.

Below is an example of input file:



The first line means {number of vertices} {number of edges / 2} {network diameter}

The line start from second line means {vertex id1} {vertex id2} {distance between vertex id1 and vertex id2}

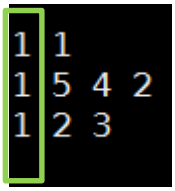
For example: 1 2 3 in the second line means there is an edge from vertex1 to vertex2 and an edge from vertex2 to vertex1 with distance 3.

The distance will be an integer.

In this example, it means an input graph shown in the above figure.

3.3 OUTPUT FORMAT

Below is an example of output file with source vertex 1:



You have to output the shortest path from source vertex to every destination vertex.

Notice that:

- The order of output is sorted by destination vertex.
- Print a blank after each vertex (includes the destination vertex), and print a "\n" after the blank of the destination vertex.
- If there are more than one shortest path for a pair, ex: distance of 1->2->3 is the same as 1->4->3, output ONLY one shortest path per destination.

4 GRADING

This homework will be graded by the correctness, report, demonstration, and performance as described below:

4.1 CORRECTNESS (45%)

During the demo time, TA will use different graph to test your program, and check whether your **output** can pass our judge script.

The detail score distribution of each part is:

- Pthread version (15%)
- Synchronous vertex-centric MPI version (15%)
- Asynchronous vertex-centric MPI version (15%)

4.2 REPORT (30%)

1. Design

   Explain the **detail** of your implementations of three versions in diagrams, figures, sentences, you also need to answer all the questions in the following aspects:

   (a) What algorithm you choose to implement Pthread version? Why?

   (b) What are the pros and cons of synchronous and asynchronous version?

   (c) Other efforts you've made in your program

2. Performance analysis

   - Environment:
     - ✧ You can use your own server to run the experiments, but you need to list the hardware of your own server in the report. (Ex: CPU, memory, network…)
     - ✧ However, make sure your code can run on our platform since we will use our platform to test your correctness and performance!
   - Plots for all three versions of implementation:

   Please explain your graph. Do not just put the graphs without any explanations!

   (a) Strong scalability chart

   i. Strong scalability – scalability to number of cores (Problem size is fixed)

   ✧ The size of the problem is determined by the number of vertices

   ✧ Select ANY 4 input test cases for this plotting

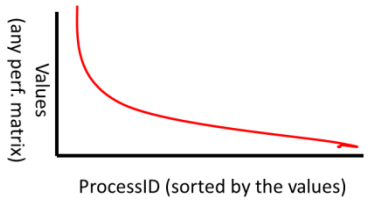
- ✧ For MPI, scaling means to increase the number of CORES not processes. (The number of processes must be the same as the number of vertices.)

(b) Performance profiling:

i. Time distribution for Pthread: computation, synchronization (lock wait time), I/O, others you prefer to show.

ii. Time distribution for MPI: communication, computation, synchronization (barrier wait time), I/O

iii. Load distribution on processes (vertices) for MPI: execution time, number of messages, etc.



(c) Any other experiments you think meaningful to compare and analyze the differences between implementation versions.

3. Experience and conclusion

(a) What have you learned and observed from this assignment?

(b) What difficulty did you encounter when implementing this assignment?

(c) If you have any feedback, please write it here.

4.3 DEMO (15%)

- Test the correctness of your codes, make sure that your code can handle different conditions.
- Go through your codes. Make sure your implementation follows the requirements.
- We will ask some questions about your codes and report.

4.4 PERFORMANCE (10%)

- Pthread version (5%)
- MPI version (5%) (We will use the faster one from the sync and async versions for scoring.)

5 REMINDER

1. We provide sample test case under /home/pp2016/shared/hw3 for your reference.
2. We provide a judge script under /home/pp2016/shared/hw3 for your reference.
3. Please package your codes and report in a file named **HW3_{student-ID}.zip** and name your codes/report as follows:

   (a) SSSP_Pthread.c

   (b) SSSP_MPI_sync.c

   (c) SSSP_MPI_async.c

   (d) Report: HW3_${student_id}_report.pdf

   (e) Please pack your compile script or Makefile with your source code and report.

   Make sure your compile script can execute correctly and your code has no compile error before you upload your homework, and copy to **homework/HW3 directory under your home directory in our server** before **12/18(Sun) 23:59**

   (You also need to upload to iLMS, but we will use the **file timestamp** under your homework/HW3 to see if you submit your homework late.)
4. Since we have limited resources for you guys to use, please start your work **ASAP**. Do not leave it until the last day!
5. Late submission penalty policy please refer to the course syllabus.
6. **Do NOT try to abuse the computing nodes by ssh to them directly**. If we ever find you doing that, you will get 0 point for the homework!
7. Asking questions through iLMS or email are welcome!