MPI Project: Single Source Shortest Paths

Otwarto: piątek, 16 maja 2025, 00:00
Wymagane do: poniedziałek, 9 czerwca 2025, 23:59

MPI Project: Single Source Shortest Paths

MPI Project: Single Source Shortest Paths

Table of Contents

Due date: 9 June 2025

1. Introduction

Your goal is to implement in MPI the distributed Δ-stepping algorithm for the Single Source Shortest Paths (SSSP) problem, along with several heuristic optimizations. The Δ-stepping algorithm can be seen as a intermediate algorithm between Dijkstra and Bellman-Ford algorithms: Dijkstra algorithm is very work-efficient, but it can't be efficiently parallelized, while the Bellman-Ford algorithm is easy to parallelize, but has larger computational complexity.

2. Objectives

Both the Δ-stepping algorithm, and the optimizations are described in the paper: https://www.odbms.org/wp-content/uploads/2014/05/sssp-ipdps2014.pdf. You should implement all of the optimizations that are described in this paper, that is:

- Edge classification including inner-outer short heuristic
- Pruning + Push vs. Pull Heuristic
- Hybridization
- Load balancing (inter and intra-node)

Thus, you should focus on the sections II - III in the above article.

Moreover you should measure the performance and weak (Gustafson) scaling of your solution, as well as the impact of implemented optimizations on the performance. The experiments should be performed on the Okeanos, and described in a report. Moreover, you are allowed to implement your own optimizations and heuristics.

3. Solution format, input and output specification

Submit a single .zip archive with a directory named ab123456/ (replace with your login). This directory should contain:

- Sources
- Makefile
- report.pdf file containing your report

After invoking the 'make' command (on Okeanos) it should build a binary named 'sssp'. This binary will be run using slurm, and each rank will be provided with two command line arguments: input and output file path. In the input file, each rank will receive part of vertices and a list of all the edges incident to all of those vertices.

The first line of the input file consits of 3 integer: the number of vertices in the whole graph, and the indexes of the first and last vertex this process is responsible for. The remaining lines contain the descriptions of the edges, consisting of 3 integers each: the indexes of end vertices of the edge and the length of an edge.

You can assume, that:

- There will be no loops, or repeated edges.
- The input part will fit into a process memory.
- The number of vertices will be less than 10^9.
- The edge lengths will be integral, nonnegative and the sum of the edge lengths in the whole graph will be less than 10^{18}.
- The verticess will be equally distributed among the processes (with up to 1 vertex difference), and the vertex indexes will be increasing with respect to the rank number.

Each process should write to the output file the result: for each vertex this process was responsible for there should be one line containing a single integer: the length of the shortest path from the source vertex 0 to this vertex.

NOTE: The source vertex is always the vertex with number 0.

We provide a testing script along with an example test in the attachment to this excercise. All the solutions that fail the example tests on Okeanos will automatically get 0 points.

4. Performance considerations

Your solution will be tested for performance on the Okeanos supercomputer. For its sake, you should fine-tune your final solution for performance. It includes optimization of various parameters of your algorithm (Δ, hybridization threshold, load balancing itp.). You may also disable some of the optimizations you implemented if they haven't improved the performance, but if you do, it should be stated and motivated in your report. Moreover, you should leave the code of such optimizations in your solution - you can only disable it using e.g. constant flag, or command line flag.

You can assume, that the graphs used in the performance tests will be generated similarily as the ones in the experiments described in the abovementioned paper. That is, the edges will be generated from either the RMAT-1 or RMAT-2 model and the edge lengths will be sampled from the uniform distribution between 0 and $w_{\max}$.

5. Evaluation Criteria

- Correctness: 6pt
- Performance: 10pt
- Implementation and assesment of optimizations:
  - Edge classification: 1pt
  - Pruning: 3pt
  - Hybridization: 2pt
  - Load balancing: 3pt (2pt for inter + 1pt for intra-node)
- Other components of the report:
  - Weak scaling of your final solution: 3pt
  - Clarity, esthetic and adherance to the best practices: 2pt

To receive full points for each optimization you need to present, in your report, a performance comparision of a solution with and without this optimization. The comparision should be performed for several numbers of workers, and should be forwarded by a short analysis of presented measurements.

Additionally you can get at most 4 points for your own optimizations, although your score can't exceed 30 points total - the additional points can only recover you points lost elsewere. The rules for those optimizations are the same as for the ones from the paper, but you also should provide a detailed explanation of the optimization, along with the rationale why such optimization may be benefitial.

Of course the implementations of all features described in the report, should be present in the code you submit (even if they are turned off for the performance sake).

For the category "clarity, esthetic and adherance to the best practices" counts among others:

- data presentation (plots / tables): clearily visible numbers, axis descriptions, confidence intervals etc.
- descriptions: it is clear what experiments were performed and what do the conclusions result from.

- choice of experiments: the scale of experiments and the choice of input datasets should be suitable for the objective.

Your solution must score at least 8 points to complete the course. Points for performance and report will ge given only if the solution is mostly correct.

6. Deadline

Please do submit your assignment by the due date. If you're late, your score will be reduced by 1 point for every 12 hours (i.e.: if you're late by 2h, we subtract 1 point from your score; if you're late by 25h, we subtract 3 points).

There is a second due date - a week after the first one. Submitting by this due date is very risky. First, very good solutions submitted by this due date receive at most 10 points. Second, there will be less time for patching. Please do submit your assignment by the normal, first due date.

7. Tips

For the implementation of the communication you might want to use the **one-sided MPI communication**. Here is a good tutorial on this topic: https://cvw.cac.cornell.edu/mpionesided
Author: Tomasz Kanas

Created: 2025-05-16 pią 15:18

checker.tgz 17 maja 2025, 13:49

Dodaj pracę

Status przesłanego zadania

Status przesłanego zadania	Nie przesłano jeszcze zadania
Stan oceniania	Nieocenione
Pozostały czas	Pozostało 7 dni 14 godzin
Ostatnio modyfikowane	-
Komentarz do przesłanego zadania	Komentarze (0)