ETH zürich

High-Performance Computing Lab for CSE **2020**

Student: FULL NAME Discussed with: FULL NAME

Solution for Project 4

Due date: 13 April 2020, 12:00pm

HPC Lab for CSE 2020 — Submission Instructions
(Please, notice that following instructions are mandatory: submissions that don't comply with, won't be considered)

- Assignments must be submitted to Moodle (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, Matlab). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  Project_number_lastname_firstname
  and the file must be called:
  project_number_lastname_firstname.zip
  project_number_lastname_firstname.pdf
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

Parallel Programming using the Message Passing Interface MPI

1. Ring addition using MPI [10 Points]

The ring addition made me adjust my mental model on how MPI works - ie. explicitly repeatedly passing the received to achieve full summation.

2. Ghost cells exchange between neighboring processes [20 Points]

We're creating a Cartesian grid over the whole (periodic) domain. I introduced a pragma to have C++-like "automatic casting" from bool to int. Sendrecv seems like a fitting choice for a method that exchanges cells.

3. Parallelizing the Mandelbrot set using MPI [20 Points]

I went for a 1D horizontal construction of a grid. In any case, a parallelization will introduce strong load imbalance as we can see from the graph. This is a result of how the image-generating loop is constructed: orbits that are being quit after 1,2,3 iterations are dominant for ranks further away from zero, which lets them finish their tasks much more easily. In the image generated with the program, we can see that an even partitioning of the image does not correspond to an even partitioning of the workload. This would maybe change somewhat using a 2D Cartesian grid partitioning but the problem remains. Ideally, we would adaptively arrange a partitioning of the grid which we would work through using a producer-consumer scheme. An example of an adaptive method would

be the Mariani-Silver algorithm (cf. http://mrob.com/pub/muency/marianisilveralgorithm.html). I would have liked to try that out but time was scarce so I moved on to the next exercise.

I tried to programmatically verify correctness (see file) but somehow 1, 2 pixels always fall flat. I hope it's not the implementation and rather a floating point difference by differently calculating because of the nondeterministic nature of MPI.

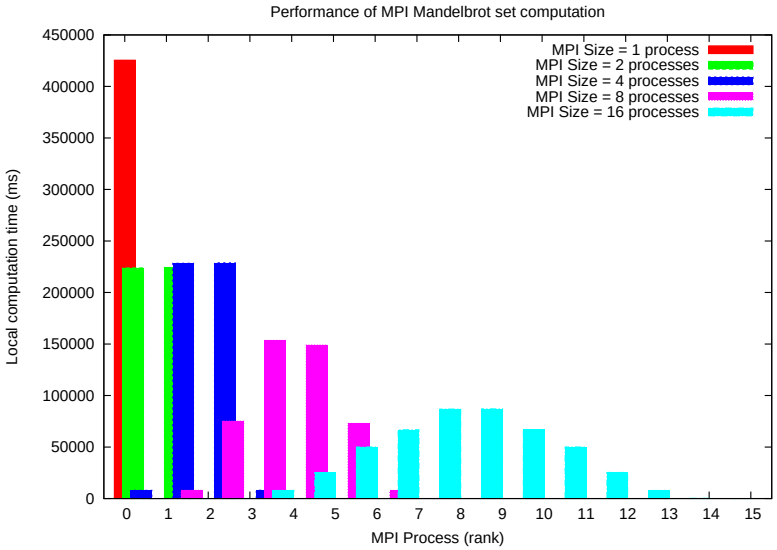

Figure 1: Load imbalance on Euler using the provided plotting script, on XeonE5 2680v3

4. Option A: Parallel matrix-vector multiplication and the power method [50 Points]

I did Option B for a week before realizing that I didn't get through enough theory to put in all the ideas, so here's the rundown for a quick solution to Option A.

5. Option B: Parallel PageRank Algorithm and the Power method [50 Points]

Additional notes and submission details

Submit the source code files (together with your used `Makefile`) in an archive file (tar, zip, etc.), and summarize your results and observations for all exercises by writing an extended Latex report. Use the Latex template provided on the webpage and upload the Latex summary as a PDF to Moodle.

- Your submission should be a gzipped tar archive, formatted like project_number_lastname_firstname.zip or project_number_lastname_firstname.tgz. It should contain
  - all the source codes of your MPI solutions;
  - your write-up with your name project_number_lastname_firstname.pdf.
- Submit your .zip/.tgz through Moodle.