MYCHP203 — TOP : Project

Gabriel Dos Santos
gabriel.dossantos@cea.fr
gabriel.dos-santos@uvsq.fr

Hugo Taboada
hugo.taboada@cea.fr

April 17, 2024

Seismic core - 3D stencil optimization

1. Problem overview

This code computes a 16th-order, 49-points, 3-axis stencil, as in Figure 1. The dimensions of the stencil are defined at runtime, via a configuration file (defaulting to `config.txt`).

The stencil operation is a Jacobi iteration. It uses three, 3rd-order tensors (i.e. 3D matrices) storing (among other things) 64-bit/double-precision floating-point values:

- A is the input tensor, initialized with a value of 1.0 for core cells, and a value of 0.0 for ghost cells.
- B is a constant input tensor, that is initialized as follows:

$$B_{x,y,z} = \sin(z \times \cos(x + 0.311) \times \cos(y + 0.817) + 0.613)$$
$$\forall x \in [0, \dim_x[, \forall y \in [0, \dim_y[, \forall z \in [0, \dim_z[$$

- C is the output tensor, with all its coefficients initialized to 0.0.

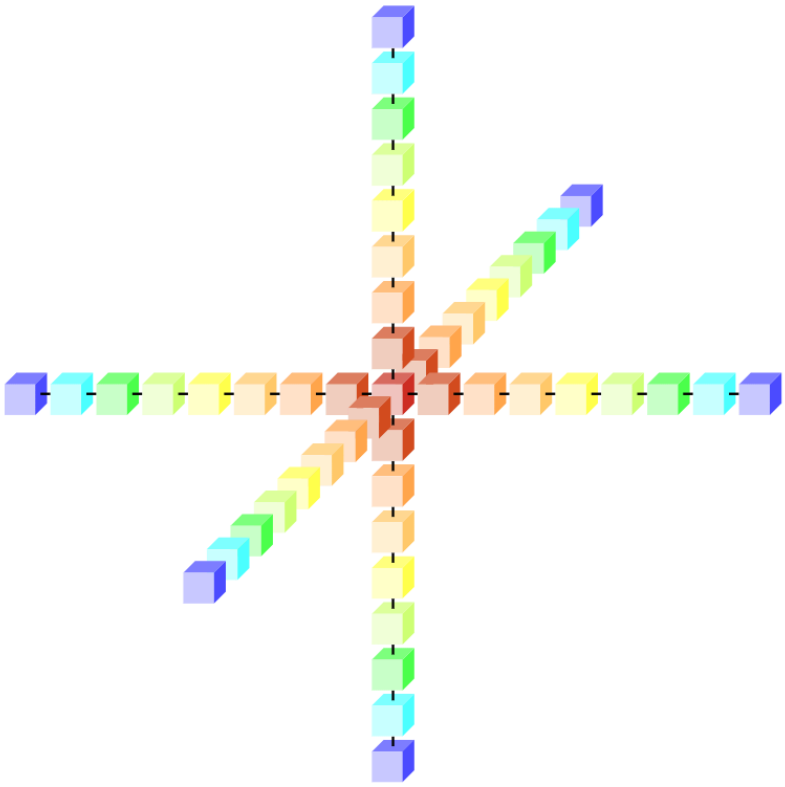

Figure 1: Visualization of the 16th-order, 49-points, 3-axis stencil

The `solve_jacobi()` function iterates over each axis z, y, and x and computes the following product:

$$C_{x,y,z} = A_{x,y,z} \times B_{x,y,z} + C_{x,y,z}$$

Then, for each neighboring cell in the x, y, and z axes up to an order of 8 (i.e., up to eight cells away from the $\{x, y, z\}$ cell), we recompute this product and divide it by an order-dependent power: 17.0^o, where o is the order of the neighbor cell we are calculating, effectively minimizing their value the further away they are from the $\{x, y, z\}$ cell.

2. Getting started

2.1. Pre-requisites

- CMake 3.16+
- C17 conforming compiler
- MPI 3.0+ implementation

2.2. Build

```
cmake -S . -B <BUILD_DIR> [CMAKE_FLAGS ...]
cmake --build <BUILD_DIR> [-j]
```

2.3. Run

```
<BUILD_DIR>/top-stencil [CONFIG_FILE_PATH] [OUTPUT_FILE_PATH]
```

3. Project modalities

This project is to be done in pairs. You shall write a well-documented report explaining your optimization process in detail. You are expected to present the debugging and/or profiling steps you have taken, the modification you have done to the code (and the reasons why), as well as how you asserted the improvements. Your report shall include a link to a Git repository containing the history of code updates.

> **Important**
>
> The deadline for submitting the report and the optimized code is set to:
>
> **Sunday, May 5, 2024 at 23:59 CEST/UTC+2**
> **2024-05-05T21:59:59.000+02:00**
>
> Project submissions past this time will be dismissed, and subsequent commits will be ignored.

You shall send it as an archive (`.tar` or `.zip`) on the following Pcloud link:
https://e.pcloud.com/#page=puplink&code=K1lZd09G1Ep1E9B7Xu1MXhrWehbaq2fX.

> **Warning**
>
> Reports sent by email will be dismissed.

In addition, you will prepare some slides summarizing your report for a project defense that will be held on the Tuesday, May 7, 2024 (date to be confirmed). You will present for 10-15 min, then exchange with the jury for 5 min of questions.

3.1. Goals

Your goal is to make the code as fast as possible. You can do whatever you want to it (rewrite it completely if you want); just make sure that it produces the same result as the baseline version. Use the `scripts/compare.py` script to validate your modifications.

> ⚠ **Warning**
>
> You *cannot* simply "print" the expected values (doing this would directly result in a 0/20).

The baseline code is very slow, even with small dimensions. Once you optimize it, you are expected to increase the dimensions of the problem, ideally up to $1000 \times 1000 \times 1000$ (requires at least 24 GiB of memory).

3.2. Grading

Your grade will be based on the overall approach (use of tools seen in class, optimizations techniques, etc...), not necessarily on the raw performance improvement (although it remains part of the grade). If possible, try to run your code on a cluster (e.g. OB1) in order to profit from the higher core count and/or NUMA architecture. Include as many figures as you want: strong/weak scaling plots, performance improvement graphs across optimizations, etc.

The Git log/history of your project is important: the share of work will be taken into account in the final grade, so make sure *both* members of the project regularly commit modifications.

3.3. Some report recommendations

- It is strongly advised to write in a academia-friendly format: either LaTeX or Typst.
- Your report shall have an introduction, (possibly) multiple sections detailing each steps of the optimization process, and a conclusion.
- Figures shall have a title, a legend, and named axes with clearly identifiable units.
- An overview of the hardware (architecture, number of cores/sockets, memory hierarchy, ...), software toolchain (profiling tools, compilers, MPI implementations, Linux kernel, ...) you have used shall be provided.
- Mention every optimization you have attempted to make, even if it did not yield an improvement in performance.

> **If you have any question, feel free to ask the TA either by email or PM on Discord.**

Have fun! :)