Activity: Numerical integration

The purpose of this assignment is for you to

- get started back in writing C++ code
- get familiar with running code on centaurus
- write a simple program that will be reused in future assignments

As usual all time measurements are to be performed on the cluster.

1 Numerical Integration (70 pts)

The purpose of this activity is to compute the following expression:

$$\frac{b-a}{n}\sum_{i=0}^{n-1} f\left(a+(i+.5)*\frac{b-a}{n}, intensity\right)$$

The provided package contains multiple functions f in `libfunctions.a`. The functions are named `f1`, `f2`, `f3`, `f4`, and take two parameters: the first one is a floating point number x where the function is computed, and the second one is *intensity* an operation intensity.

The code you should write should take 5 command line parameters:

- `argv[1]` is `functionid`, an integer to know which function to use as f. If `functionid` is 1, use `f1`; If `functionid` is 2, use `f2`; etc.
- `argv[2]` is `a`, a double
- `argv[3]` is `b`, a double
- `argv[4]` is `n`, a 64-bit integer
- `argv[5]` is `intensity`, an integer

The code should compute the expression and output the value of the on `stdout` (and nothing else). The code should also measure the time it took to compute the expression and write that time (expressed in seconds with decimal values) to `stderr` (and nothing else).

Question: Write the described code. You can use the provided archive as a template. It contain a template code and makefile to help you write the code. You should only need to complete `main.cpp`. You should be able to test if your code is correct using `make test`.

2 Benchmarking on centaurus (20 pts)

Question: Report the time it takes on the cluster to evaluate the expression using `f1` and different values of `n` (from 10^1 to 10^8) and `intensity` (from 1 to 10^4). To help you in that task, you should be able to run `make bench` which should run the benchmark in a SLURM job. Once that job is completed, you can draw charts using `make plot` which reports time in a pdf file `plot/time_plots.pdf`.

Make sure you keep this code around as it is your base for comparisons in future assignments.

FAQ

I can't find the code of function f1, f2, f3, f4 ?!

Indeed, you don't have access to them. You do not need access to the code of functions `f1`, `f2`, `f3`, and `f4`. These functions are in `libfunctions.a`.

When I compile, it does not find function f1.

Do not compile by hand, use `make`.

When I run make bench, it tells me "sbatch command not found".

You need to run `make bench` on centaurus.

When I run make bench, it tells me I need to pass the tests.

You need to have successfully run `make test`.

How long will the job take to run?

The job may run on a centaurus compute node for about an hour. (Depending on how the code is written). Note that you don't need to stay logged on centaurus. Once the job is batched, centaurus will eventually run it.

How should I measure time?

Use `std::chrono::system_clock`. Check cpprefence.

I get "error: unrecognized command line option '-std=c++17"' when compiling on centaurus

Add `module load gcc` at the end of your `~/.bashrc` on centaurus.

Why do we can about computing this anyway?

The expression compute the integral of function f. But it REALLY does not matter why or how that works. The activity is JUST to evaluate the expression. In case you want to know more. here is some description:

Numerical integration is often used when one wants to compute $\int_a^b f(x)dx$ but one does not know how to find a primitive of f. You can use the definition of integration to obtain a simple approximation by computing $\frac{b-a}{n}\sum_{i=0}^{n-1} f\left(a+(i+.5)*\frac{b-a}{n}\right)$. n is often called the number of point in the approximation. (This is the numerical integration using the rectangle rule. You can learn more at `https://en.wikipedia.org/wiki/Numerical_integration`.)

The provided package contains multiple functions to integrate in `libfunctions.a`. The functions are named `f1`, `f2`, `f3`, `f4`, and take two parameters: the first one is a floating point number x where the function is computed, and the second one is *intensity* an operation intensity. The second parameter is used to make the function take more time.

The code you should write should take 5 command line parameters:

- `functionid`, an integer to know which function to integrate. If `functionid` is 1, integrate `f1`; If `functionid` is 2, integrate `f2`; etc.
- `a`, the lower bound of the integral
- `b`, the upper bound of the integral
- `n`, an integer which is the number of points to compute the approximation of the integral
- `intensity`, an integer which is the second parameter to give the the function to integrate. You will see that in the scaffold `f1` takes two parameter, an x, and an *intensity*. Use that parameter as *intensity*.