Static Loop Scheduler

As usual all time measurements are to perform on the cluster.

1 Preliminary

Question: Before anything, we will need to get timing for sequential execution. Navigate to the `sequential/` directory. Compile the sequential code with `make sequential` and get sequential time by running `make bench` on `Centaurus`.
Question: Rewrite the numerical integration problem using the looping construct. There is a sequential implementation of the looping construct in `sequential/` directory. The construct is in `seq_loop.hpp` while the test codes for the looping constructs are in `loopsample.cpp`. Write that code in `static/static_main.cpp`. Write the code so that it outputs the integral value on `stdout` and the time it takes to make the computation on `stderr`.

You can test the correctness of your sequential implementation using the looping construct with `make test` from the `static/` directory.

The program should take the following command line parameters:

- `functionid`, same as in numerical integration assignment
- `a`, same as in numerical integration assignment
- `b`, same as in numerical integration assignment
- `n`, same as in numerical integration assignment
- `intensity`, same as in numerical integration assignment
- `nbthreads`, an integer indicating the number of threads that should perform the numerical integration

Ignore `nbthreads` for your sequential adaptation.

2 Writing a static loop scheduler

Loop schedulers are used essentially whenever the code reads like your stereotypical *for* loop. In the previous assignment, transform would be a stereotypical for loop, but reduce could also be written this way. Here we will use the numerical integration example.

A static loop scheduler is the simplest way to achieve some parallelism. We will use it to make the numerical integration problem parallel.

Essentially, with T threads, each thread will do exactly $\frac{1}{T}$ of the loop iterations. So that if the loop has 100 iterations and there are four threads, the first threads will execute loop iterations from 0 to 24, the second will execute iterations from 25 to 49, ...

Be careful of what happens when the number of threads is not a diviser of the number of iterations.
Question: Rewrite the looping construct in `static/static_loop.cpp` to execute in parallel using a static scheduler. You can still use `make test` to test your code.

You will need to implement the parfor function to create threads, execute the before function, distribute the loop iterations to the different threads using a static distribution and execute them, and finally execute the after function.

You may also need to add functions to the looping construct abstraction to enable controlling the number of threads and add a call to that function from your main code. (Pass the value from the `nbthreads` command line parameter.)

Note: the `static/static_loop.cpp` contains comments to help you write the construct.

Question: Report time and speedup across a range of precision and intensity. Your code MUST pass the test before you can use `make bench` to start the SLURM jobs. Once they are complete, plot the charts with `make plot`. Note: you must run these commands from within the `static/` directory.

Question: Does the speedup you achieve look right?