Universität Heidelberg
Institut für Technische Informatik

Robert Strzodka
Application Specific Computing

CPU Algorithm Design

Exercise 6

- **Submit electronically as one tar(or tar.gz) file inlcuding C++ source files, CMake build system files and an report(PDF) to Moodle until Thursday, 05.06.2025 23:59**
- **Include names on the top of the sheets.**
- **A maximum of three students is allowed to work jointly on the exercises.**

1 OpenMp scheduler (15 points)

In `omp_saxpy.hpp` we have two function called `transform_mandelbrot` and `transform_mandelbrot_fixed` where they calculate the mandelbrot set parallelly. `maxIter` is the number of iteration before we consider the point not diverging. The `omp for` without any derctives uses a default scheduler.

1. Test you code with different scheduler for the two functions and compare the result.
2. Explain the difference between the result of these two functions in terms of the performence impact of difference scheduler.

Remark

The performance metric for these function is a rough estimation of GFLOPS/s, not GB/s.

2 Calculation of pi(15 points)

The `reduce_pi_grid` fucntion uses area calculation on grid points to approximate the value of pi. It generate 2d grid points in $[0,1] * [0,1]$ and test how many points are inside of the $\frac{1}{4}$ circle. and the fraction of points is the area of this $\frac{1}{4}$ circle, and we can compute pi accordingly. The `reduce_pi_rand` uses random generated points instead of grid points. The random number generator is very expensive in terms of FLOPS.

1. Accelarate the `reduce_pi_grid` and `reduce_pi_rand` function by taking advantage of simd. `reduce_pi_grid` after proper optimization should gives 700+ GFLOPS/s (here the metric also returns GFLOPS/s).

3 Saxpy and Triad(15 points)

Complete the `omp_traid.hpp` where each function perform a Triad instead of Saxpy. Compare the performence of Triad and saxpy.

4 Bounding box calculation (25 points)

Complete the `reduce_boundingbox` function in `omp_reduce.hpp`. Given a set of 2d points, this function should calculate the bounding rectangle of the points. The result should contain two points, (x_1, y_1) and (x_2, y_2), these two points represent the lower left corner and upper right corner of the resulting bounding box. Try to compute this with one pass. i.e. `omp for reduction` should only showup once in your code.

Remark

The saxpy result for the benchmarker run have some problems. Don't worry if you see weird result. If you run it with 'main' files it should give the correct result. But make sure your bounding box function and pi function returns the correct result.