Program 1: Parallel Fractal Generation Using Threads

Task:

Parallelize the Mandelbrot generation using std::thread and analyze load balancing

Machine:

CPU: Intel Core i5-11400F, 2.60GHz,

contains 6 cores, 12 threads

- Idea: divide total rows by the number of threads. Each thread computes multiple consecutive rows, and the last thread takes care of the rest
- Problem: since threads have different computational cost, those on the side would soon become idle while the middle ones are still busy
- 3 threads takes more time than 2 threads, because the time spent in the middle is higher than ½ of the image

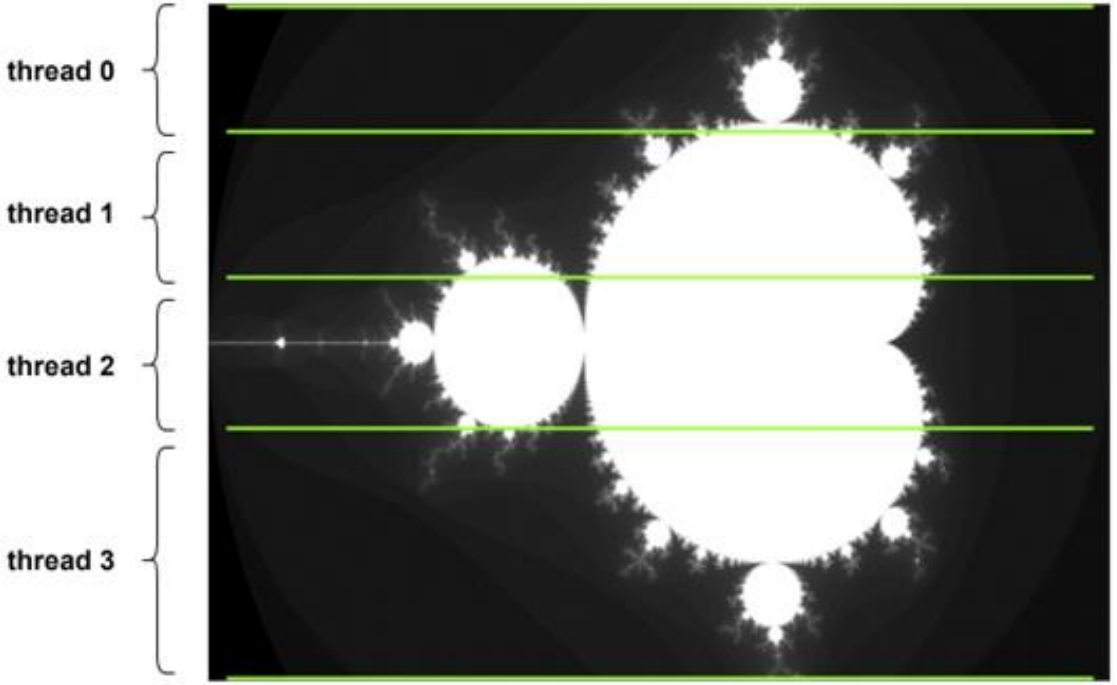

Fig1. Consecutive rows are assigned to each thread

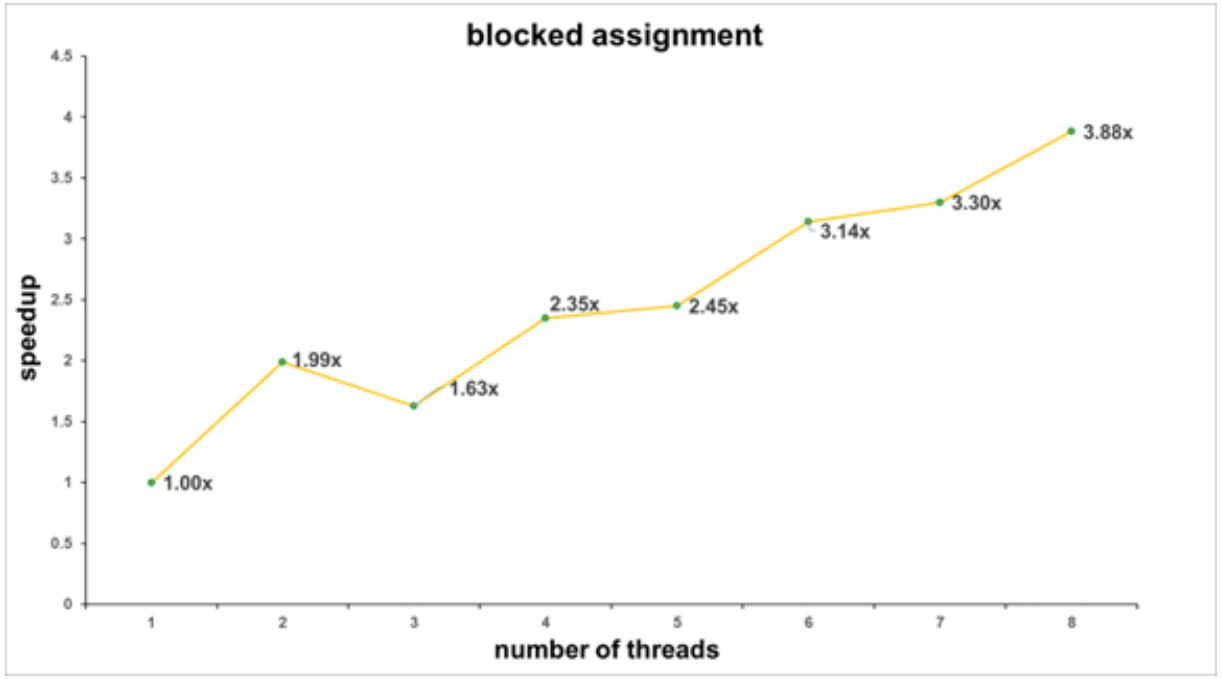

Fig2. Eight threads only produce less than 4x speedup, indicating load unbalance

- Solution: using the “interleaved assignment”, each thread calculates one row at a time, skips several rows, then calculates again
- Computational cost of each thread can thus be roughly the same
- Result: nearly linear speedup
- Further: how about 2 threads for each row?

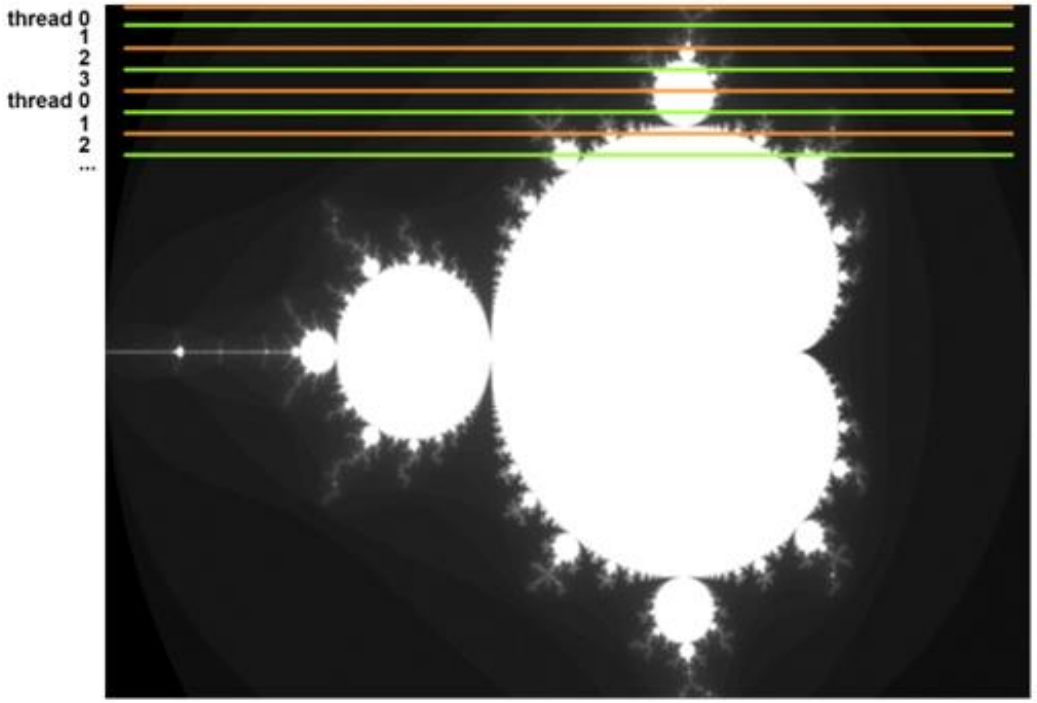

Fig3. Non-consecutive rows are assigned to each thread to implement load-balancing

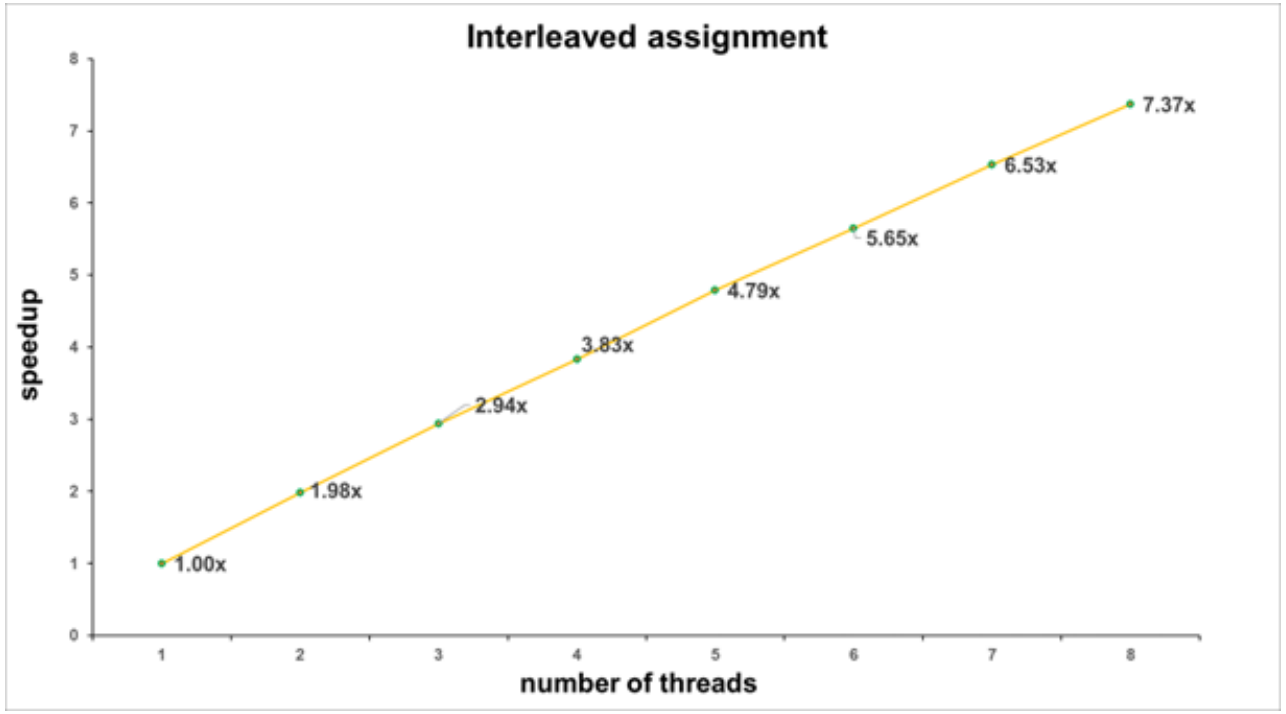

Fig4. Load balancing produces nearly linear speedup