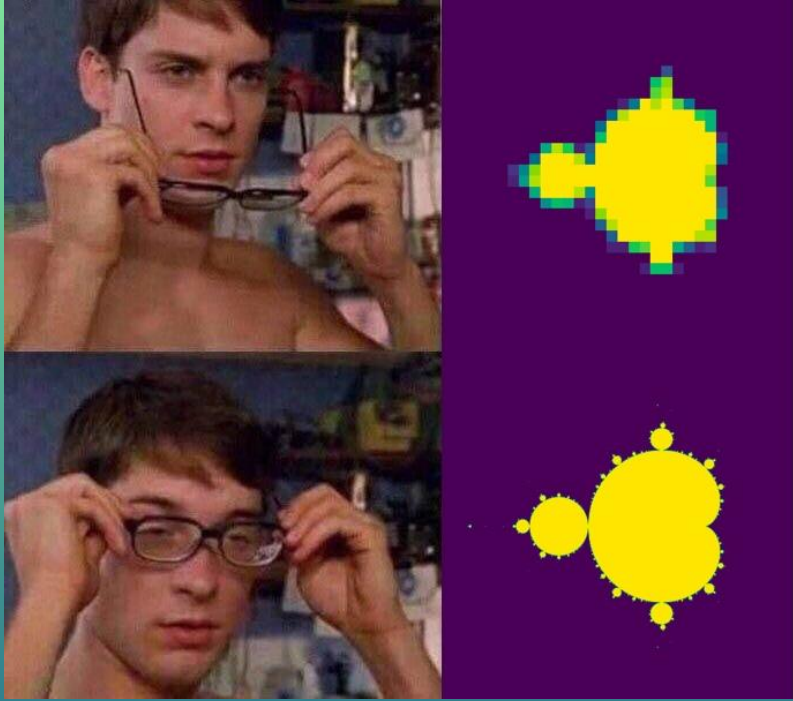
Mandelbrot Area Challenge



GROUP 1 / Red

Jan Lukas Späh, Felix Pfeifle, Ahmad Ihsan, Nicolas Hayen, Máté Farkas

Organization among the Team

ERUM DATA HUB

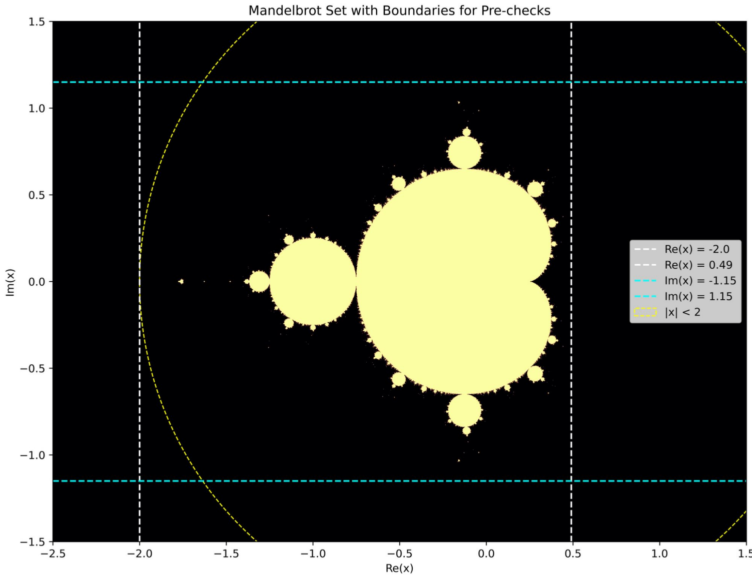
Distribution of work

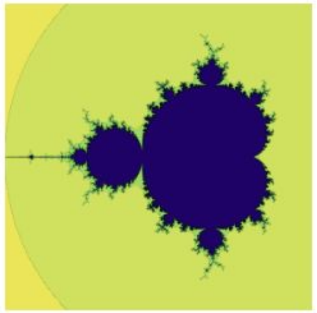
- Jan Lukas and Felix: Numba/Cuda implementation and optimisation
- Ahmad: Port python code into c++
- Nicolas: Mathematical investigations, algorithm improvements
- Máté: Python code port and the C++ code optimization

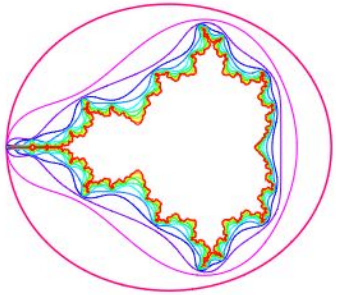
Approaches

Discriminator Optimization

- Refine bounds of sampling box
  - x_min to 0.49
  - y_max / y_min to 1.15
- Terminate early for initial c with abs(c) > 2
- Terminate early if within first cardioid (N=1) or first order element (N=2)
- Improvement by about 20%





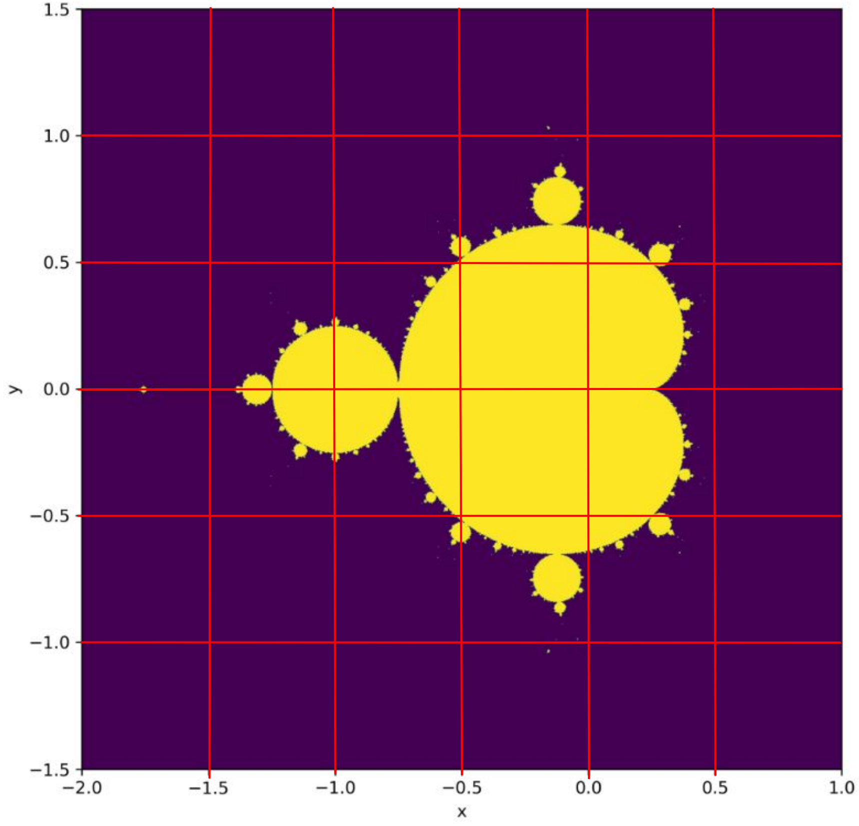


- Tried alternative algorithms to Tortoise and Hare
- Cached intermediate iteration results and compare
- No improvement in testing

Source: O.Knill, *The Area of the Mandelbrot Set*, https://people.math.harvard.edu/~knill/teaching/math21a2019/exhibits/mandelbrot/mandelbrot.pdf

Technical Implementation

- Using the provided code with numba.cuda it can run on a GPU
- The calculation of the sampled tiles was parallelized over the GPU blocks
- With 32x32 threads per block choosing tiles as a multiple of 32 so the area is divided over the number of blocks
- Implementation:
  - Split grid into blocks of 32x32 and tiles
  - 1 tile = 1 thread
- Important: Use float32 and complex64 to reduce memory footprint per thread
- Also tried direct C++ cuda implementation: Reduce overhead from JIT compilation



Results

Numba-CUDA Implementation

- Calculation performed on Nvidia L40
- Result in **1m7s**
  - Area: **1.50659(4)**
  - Width of 95% interval: **7.1e-06**
  - Relative Uncertainty: **4.7e-06**
- Result in **3m49s**
  - Area: **1.50659(57)**
  - Width of 95% interval: **1.4e-06**
  - Relative Uncertainty: **9.1e-07**
- Result in **14m35s**
  - Area: **1.506597(5)**
  - Width of 95% interval: **4.4e-07**
  - Relative Uncertainty: **2.9e-07**

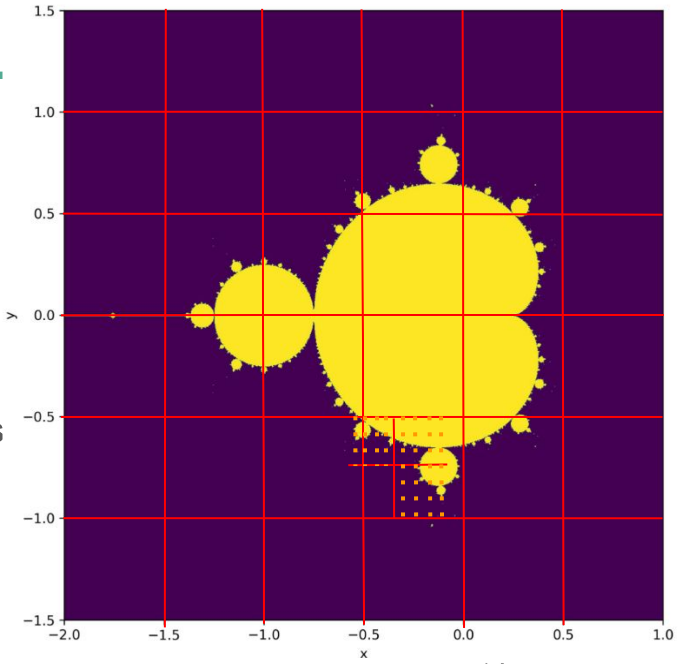
C++ CUDA Implementation

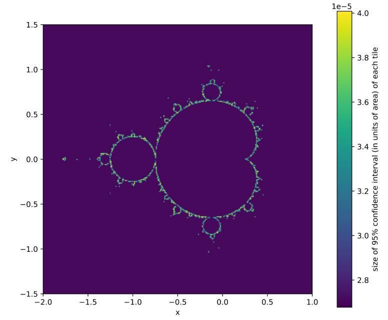
- Configuration:
  - nvidia RTX 4060
  - 10x10 grid
  - 1 thread per grid
  - uncertainty target: 1e-2
  - t = **23.14 s**
  - A = **1.501196**

Outlook

Possible improvements

- Adaptive tiling based on the uncertainty: Iterative approach
    - Divide boundary tiles into subtiles: Leave out homogenous tiles
    - Run kernel again with improved granularity
    - Aggregate results
- Almost free lunch: Use symmetry
- C++:
    - cuda random number generation slow
    - no striding
    - further improvement expected with global variable incrementation




1e−5
size of 95% confidence interval (in units of area) of each tile