GROUP № 3

Mandelbrot Area Challenge

GROUP MEMBERS:
Viacheslav Bochkarev, Jan Bürger, Lorenz Gaertner, Simon Oster, Nitish Kumar K. V., Shahabeddin Dayani

Organization among the Team

ERUM DATA HUB

Organization

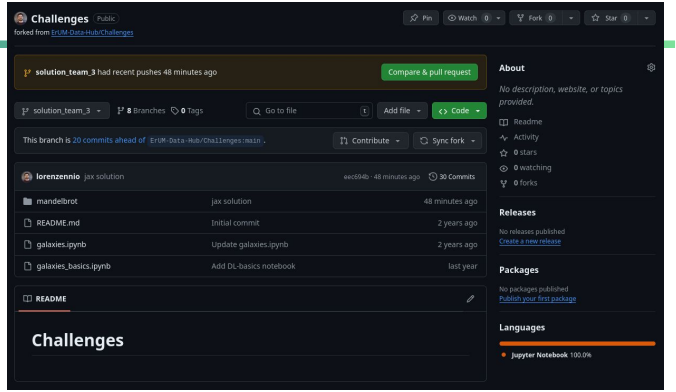
Numba implementation: V. Bochkarev, L. Gärtner

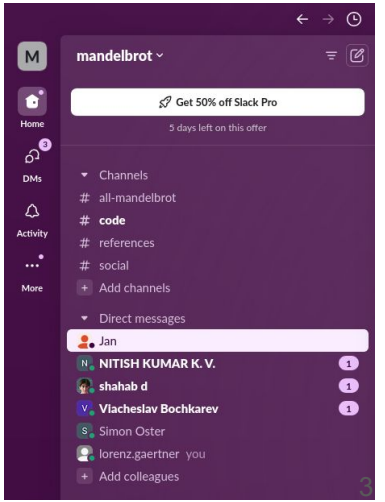
Jax implementation: J. Bürger, L. Gärtner, N. Kumar, S. Oster

Optimization and experiments: J. Bürger, V. Bochkarev, S.Dayani

Refining tiles approach: N. Kumar

State-of-the-art: S. Dayani





Ansatz

numba

- Start with provided code
- Implement @numba.cuda.jit
- Chose sensible thread and block sizes

jax

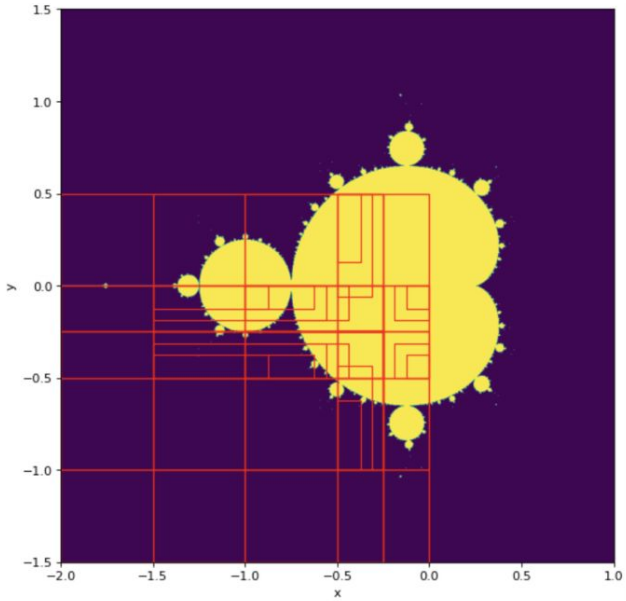
- Start with provided code
- Implement the loops in @jax.jit
- Increase number of batches

Smart tiling

Exploit symmetry and half area

- Our idea:
  - Start with uniform tiles
  - Loop over each tile
  - Get uncertainty of area in the tile
  - If uncertainty > threshold:
    - Subdivide the tile in four equal parts

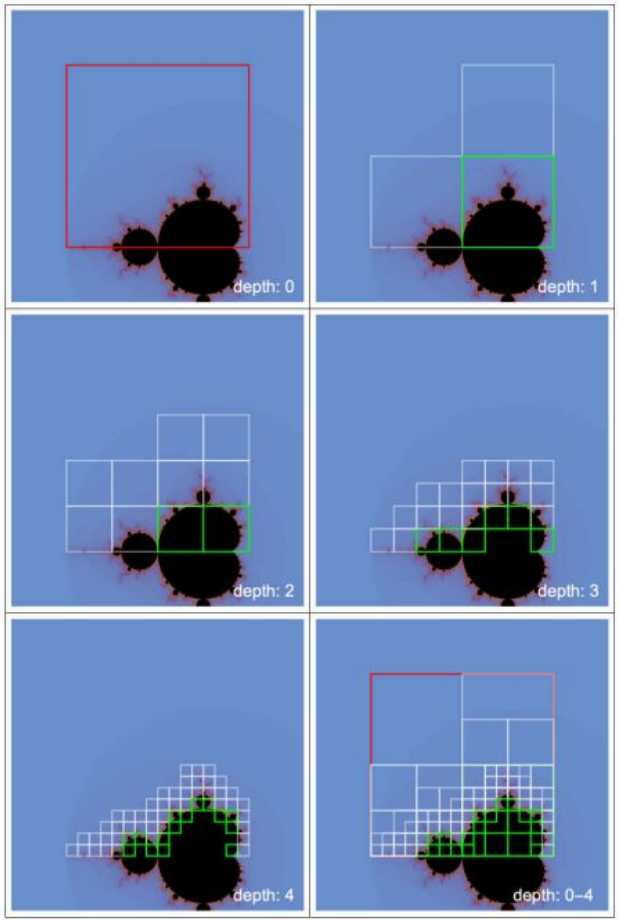
Preliminary version



State-of-art

Tiles are classified into five types:

- Full Member Tiles: Entirely inside the Mandelbrot set.
- Full Outside Tiles: Entirely outside the Mandelbrot set.
- Partly Member Tiles: Center inside but partly outside.
- Partly Outside Tiles: Center outside but partly inside.
- Chaotic Tiles: Indeterminate due to limitations in the distance calculation.



State-of-art

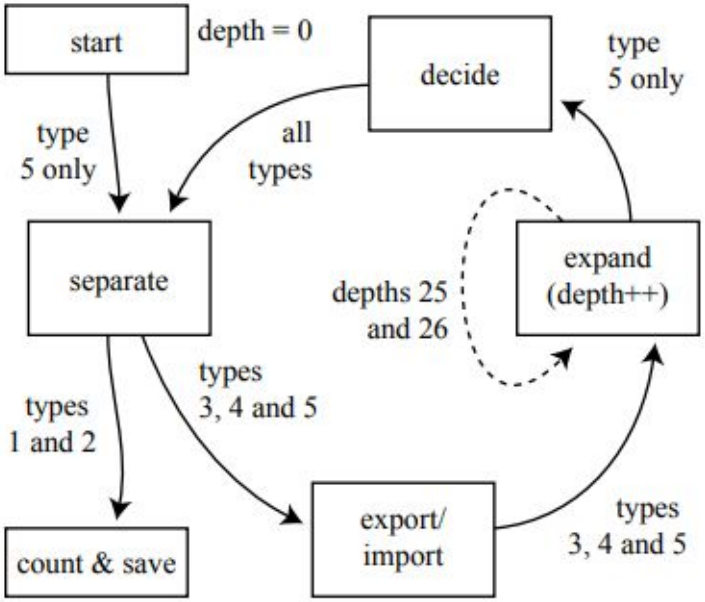
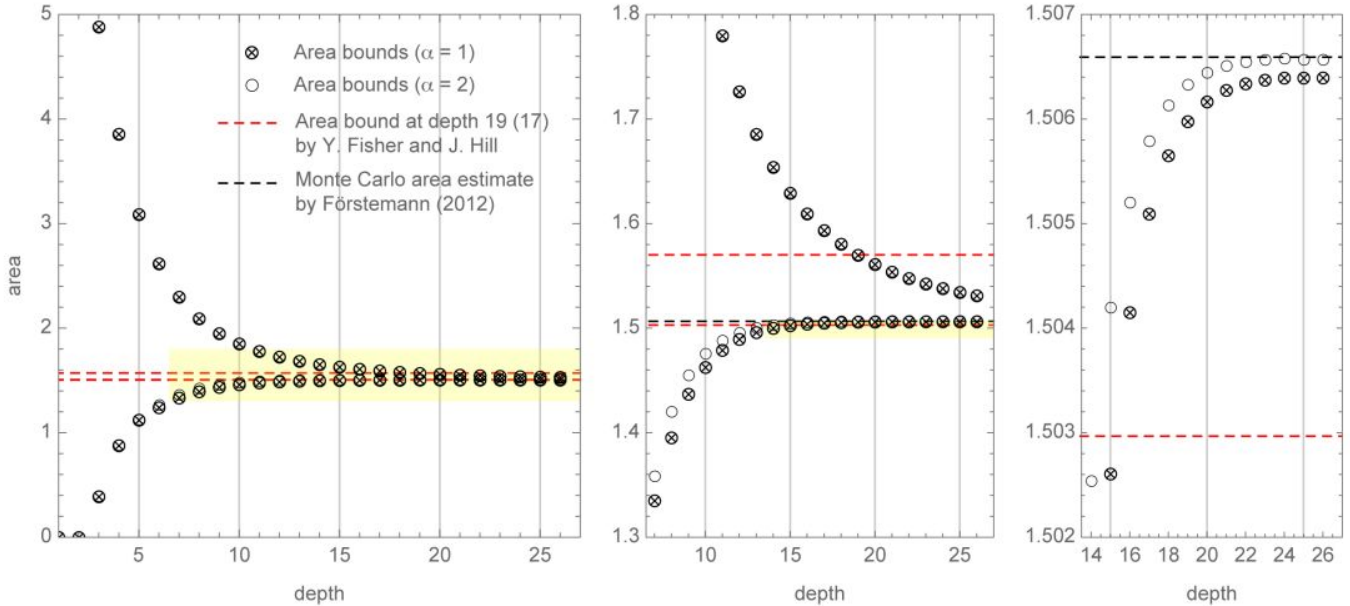


FIG. 5. The algorithm of the quad tree tessellation. The quad tree algorithm is outlined in section II B 4.

$d(\alpha)$	lower bound	upper bound	difference	CPU time
0(1)	0.000 00	12.500 00	12.500 00	$<$ 0.01 h
1(1)	0.000 00	9.375 00	9.375 00	$<$ 0.01 h
2(1)	0.000 00	7.812 50	7.812 50	$<$ 0.01 h
3(1)	0.390 63	4.882 81	4.492 19	$<$ 0.01 h
4(1)	0.878 91	3.857 42	2.978 52	$<$ 0.01 h
5(1)	1.123 05	3.088 38	1.965 33	$<$ 0.01 h
6(1)	1.242 07	2.618 41	1.376 34	$<$ 0.01 h
7(1)	1.335 14	2.299 50	0.964 35	$<$ 0.01 h
8(1)	1.395 42	2.092 74	0.697 32	$<$ 0.01 h
9(1)	1.437 04	1.950 36	0.513 31	$<$ 0.01 h
10(1)	1.462 67	1.851 96	0.389 29	$<$ 0.01 h
11(1)	1.479 03	1.779 83	0.300 80	0.01 h
12(1)	1.489 55	1.726 36	0.236 81	0.02 h
13(1)	1.496 08	1.685 55	0.189 48	0.03 h
14(1)	1.500 10	1.654 05	0.153 95	0.04 h
15(1)	1.502 61	1.629 25	0.126 64	0.05 h
16(1)	1.504 15	1.609 49	0.105 34	0.09 h
17(1)	1.505 09	1.593 57	0.088 47	0.21 h
18(1)	1.505 65	1.580 61	0.074 96	0.55 h
19(1)	1.505 98	1.569 98	0.064 00	1.53 h
20(1)	1.506 17	1.561 17	0.055 01	4.47 h
21(1)	1.506 28	1.553 84	0.047 57	13.38 h
22(1)	1.506 34	1.547 70	0.041 36	41.83 h
23(1)	1.506 37	1.542 51	0.036 14	135.65 h
24(1)	1.506 39	1.538 12	0.031 72	$\sim$ 275 h
25'(1)	1.506 40	1.534 40	0.028 00	$\sim$ 170 h
26'(1)	1.506 40	1.531 21	0.024 81	$\sim$ 550 h

State-of-art


Area bounds (α = 1)
Area bounds (α = 2)
Area bound at depth 19 (17) by Y. Fisher and J. Hill
Monte Carlo area estimate by Förstemann (2012)
area
depth
depth
depth

Results

Main results

```
numba.cuda:   1.5066035179718393 +/- 5.710544887432084e-09 // 1023s
jax:          1.5084146321999998 +/- 2.885035565690831e-09 //  348s
```

🐞 Anyone see the problem?

Outlook

TODOs

- Debug Jax version → some bug concerning the area calculation
- Implement smart tiling
- Optimize the grid and number of samples in batch
- Think of overall new method 😎