Parallel solution for computing Reiter's model snowflake

Univerza v Ljubljani Fakulteta za racunalnistvo in informatiko 2023

izr. prof. dr. Uroš Lotrič
asist. dr. Davor Sluga

Dejan Mandic, Gasper Copi, Andrej Subelj

Reiter's model snowflake

Definition 1 *A cell z is frozen if $s_t(z) \geq 1$ (an F-cell) . If a cell is not frozen itself but at least one of the nearest neighbours is frozen, the cell is a* boundary cell *(a B-cell). A cell that is neither frozen nor boundary is called* nonreceptive *(an NR-cell). The union of frozen and boundary cells are called* receptive cells *(R-cells).*

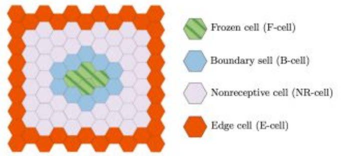
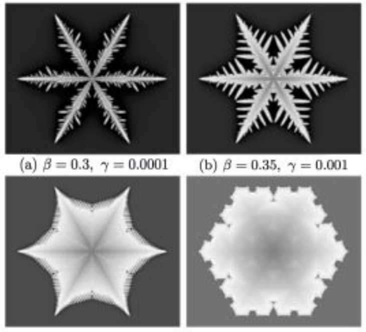


FIG. 2: Classification of cells.



(a) $\beta = 0.3,\ \gamma = 0.0001$ (b) $\beta = 0.35,\ \gamma = 0.001$

(c) $\beta = 0.6,\ \gamma = 0.01$ (d) $\beta = 0.9,\ \gamma = 0.05$

Define the following functions on a cell z:

1. the amount of water that participates in diffusion ut(z); and
2. the amount of water that does not participate vt(z)

st(z) = ut(z) + vt(z)

Constant addition. For any receptive cell z

v + t (z) := v − t (z) + γ

Diffusion. For any cell z,

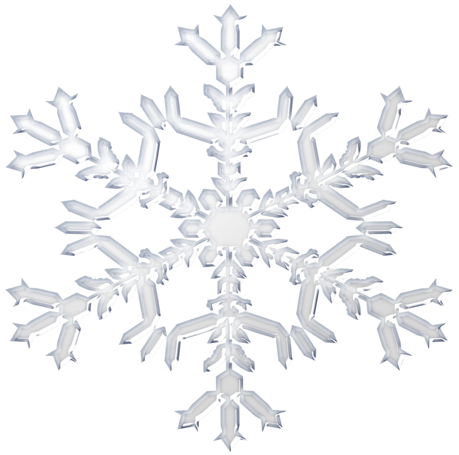
u + t (z) := u − t (z) + α 2 (u − t (z) − u − t (z)),

Receptivity of byfrost cells and other cells:

$$D(z) = \begin{cases} 1 \\ 0 \end{cases}$$

Water quantity for middle cell, and other cells:

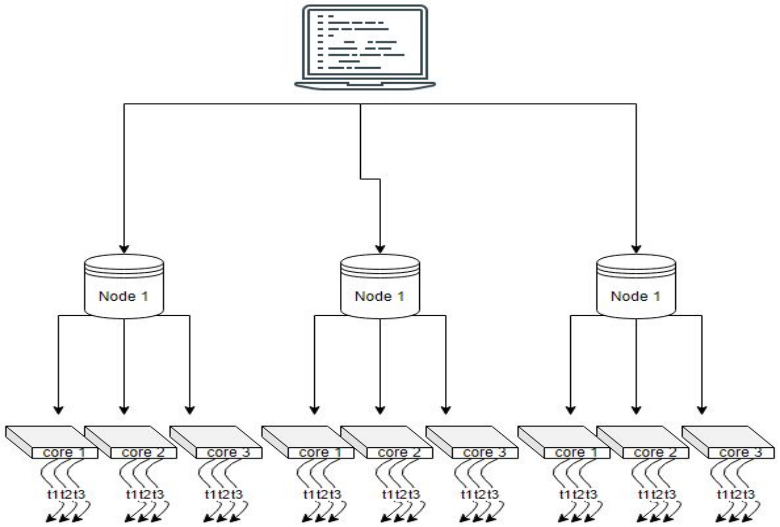
$$s_0(z) = \begin{cases} 1 \\ \beta \end{cases}$$

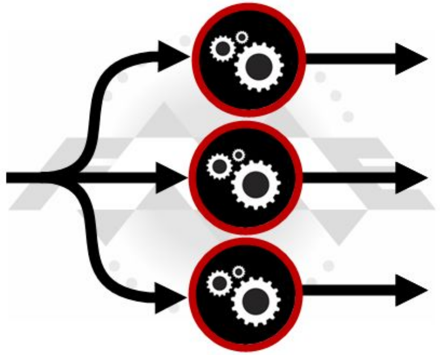


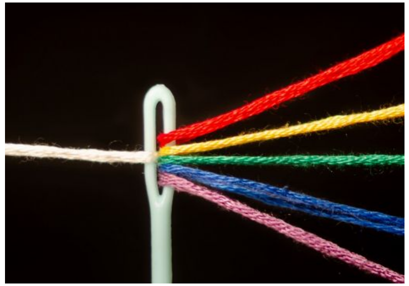
Solutions

Multi-core processing: MPI https://www.open-mpi.org/doc/v4.0/man3/MPI.3.php

Multreading: pthreads https://man7.org/linux/man-pages/man7/pthreads.7.html

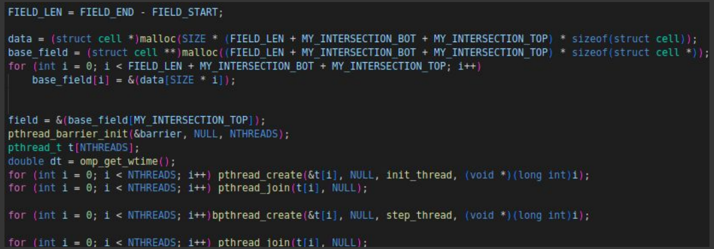






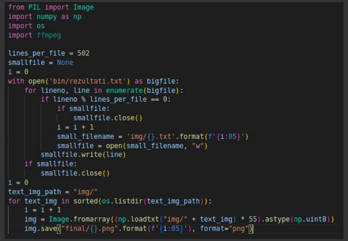
Implementation

Parallelized C code



```c
FIELD_LEN = FIELD_END - FIELD_START;

data = (struct cell *)malloc(SIZE * (FIELD_LEN + MY_INTERSECTION_BOT + MY_INTERSECTION_TOP) * sizeof(struct cell));
base_field = (struct cell **)malloc((FIELD_LEN + MY_INTERSECTION_BOT + MY_INTERSECTION_TOP) * sizeof(struct cell *));
for (int i = 0; i < FIELD_LEN + MY_INTERSECTION_BOT + MY_INTERSECTION_TOP; i++)
    base_field[i] = &(data[SIZE * i]);


field = &(base_field[MY_INTERSECTION_TOP]);
pthread_barrier_init(&barrier, NULL, NTHREADS);
pthread_t t[NTHREADS];
double dt = omp_get_wtime();
for (int i = 0; i < NTHREADS; i++) pthread_create(&t[i], NULL, init_thread, (void *)(long int)i);
for (int i = 0; i < NTHREADS; i++) pthread_join(t[i], NULL);

for (int i = 0; i < NTHREADS; i++)bpthread_create(&t[i], NULL, step_thread, (void *)(long int)i);

for (int i = 0; i < NTHREADS; i++) pthread_join(t[i], NULL);
```

Python script for parsing output into images and a video



```python
from PIL import Image
import numpy as np
import os
import ffmpeg

lines_per_file = 502
smallfile = None
i = 0
with open('bin/rezultati.txt') as bigfile:
    for lineno, line in enumerate(bigfile):
        if lineno % lines_per_file == 0:
            if smallfile:
                smallfile.close()
            i = i + 1
            small_filename = 'img/{}.txt'.format(f'{i:05}')
            smallfile = open(small_filename, "w")
        smallfile.write(line)
    if smallfile:
        smallfile.close()
i = 0
text_img_path = "img/"
for text_img in sorted(os.listdir(text_img_path)):
    i = i + 1
    img = Image.fromarray((np.loadtxt("img/" + text_img) * 55).astype(np.uint8))
    img.save("final/{}.png".format(f'{i:05}'), format="png")
```

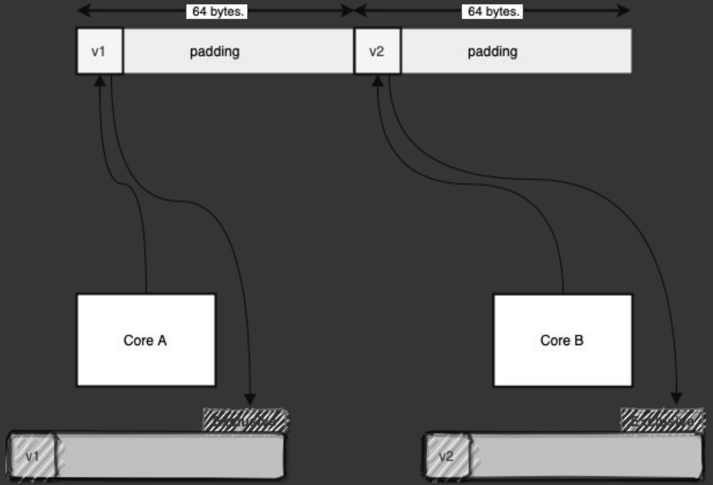
Issues - Solutions

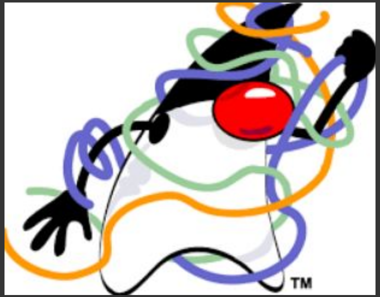
False sharing

- Neighbor intersection sharing
- Cache line padding
- Thread private data

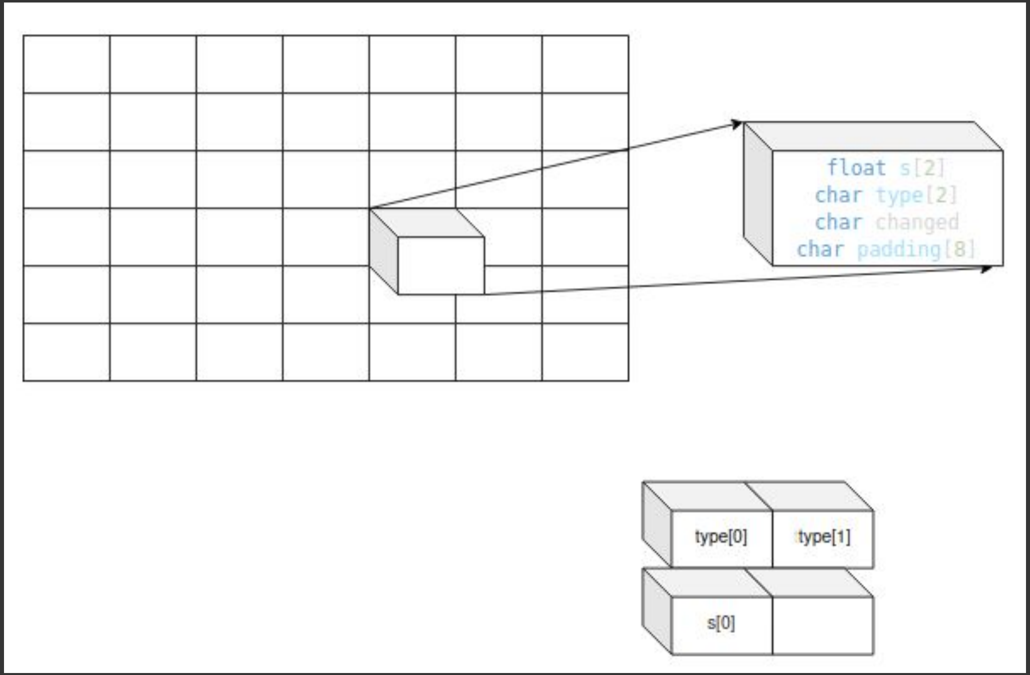
MPI

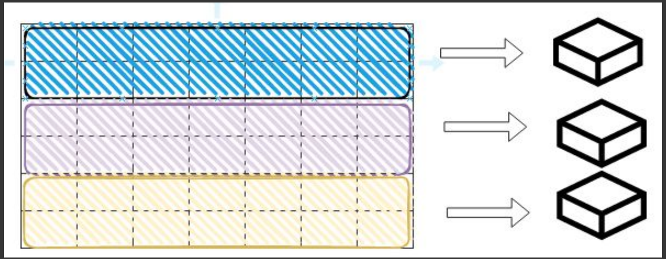
- Asynchrone(Isend, Irecv) termination





Issues - Solutions

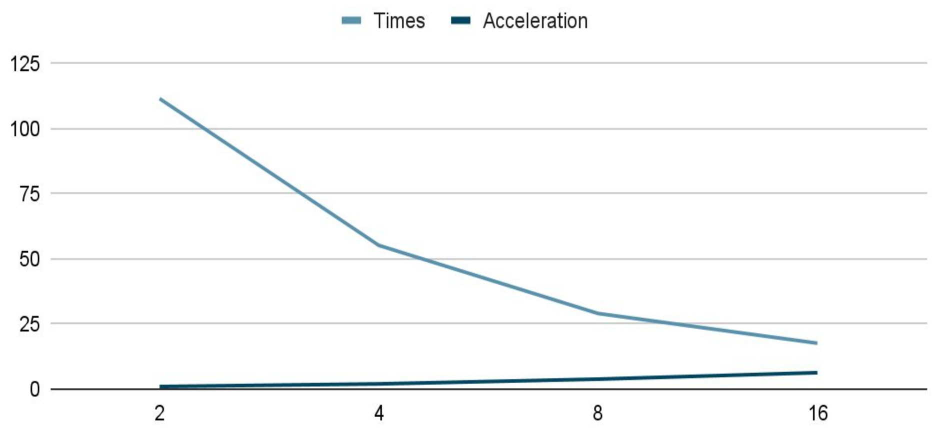




Analysis

1000x1000 2 nodes

Points scored



```
data    N       threads         inter   time
THRE
400             2       1               2               |0.663
400             2       2               2               |2.742
400             2       4               2               |1.540
400             2       8               2               |0.807
400             2       16              2               |0.663


threads
1000    2       2               2               |111.595
1000    2       4               2               |55.2255
1000    2       8               2               |29.060
1000    2       16              2               |17.643327

1000    3       16              2               |8.2143 // 3 nodes
1000    4       16              2               |5.477 // 4 nodes


400             2       1               2               |0.663
400             2       2               2               |2.742
400             2       4               2               |1.540
400             2       8               2               |0.807
400             2       16              2               |0.663

intersection
1000    2       2               2               |111.396
1000    2       2               8               |114.210
```