DD2356 Final Project: Heat Equation Solver & Idle Period Propagation

Jacob Wahlgren

KTH

2021-06-08

Setup

- Solve heat equation on the unit square
  - Constant boundary conditions (100)
  - Initial temperature 0
  - Diffusion coefficient 4
- Finite difference method
- Grid of $X \times Y$ points
- N processes

Heat equation

Thermal diffusivity k, temperature θ, time t, two dimensional Laplacian ∇^2.

$$\frac{\partial \theta}{\partial t} = k \nabla^2 \theta$$

Finite difference method

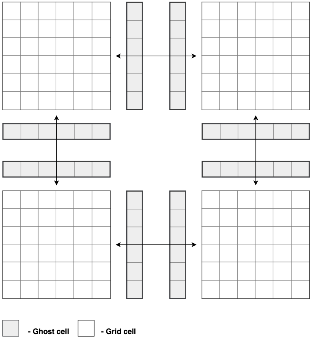
Approximate time derivate using forward difference method.

$$\frac{\partial \theta}{\partial t} \approx \frac{\theta^{n+1} - \theta^{n}}{\Delta t}$$

Discretize space into grid and use timestepping with the following update rule.

$$\theta_{i,j}^{n+1} = \theta_{i,j}^{n} + \kappa \Delta t \left(\frac{\theta_{i+1,j}^{n} - 2\theta^{n} i,j + \theta^{n} i-1,j}{h_x^2} + \frac{\theta_{i,j+1}^{n} - 2\theta^{n} i,j + \theta^{n} i,j-1}{h_y^2} \right)$$

Domain decomposition



- Square tiles
- Balanced workload
- $XY \mod N = 0$

MPI communication

```
MPI_Cart_create(MPI_COMM_WORLD, 2, dims,
                periods, 1, &comm_cart);
for (int t = 0; t < T; t++) {
    // Copy to sendbuf
    MPI_Neighbor_alltoall(sendbuf, n, MPI_DOUBLE,
                          recvbuf, n, MPI_DOUBLE,
                          comm_cart);
    // Copy from recvbuf
    // Compute timestep
}
```

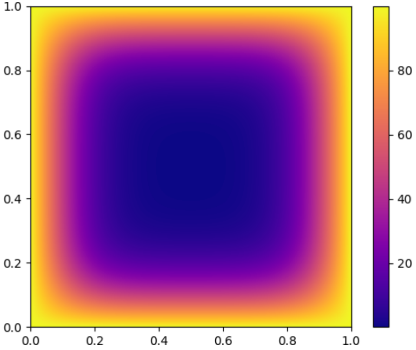
Output

- Rank 0 writes parameters to file `heat-meta.bin`
- Each process writes its tile to file `heat-x-y.bin`
- Transfer files to laptop for visualization (matplotlib)

Tests

Two tests to ensure that the same solution is obtained. 10k timesteps, 512^2 mesh points.

1. With 1 process
2. With 256 processes




```
$ cmp test1-data/vis.png test2-data/vis.png
```

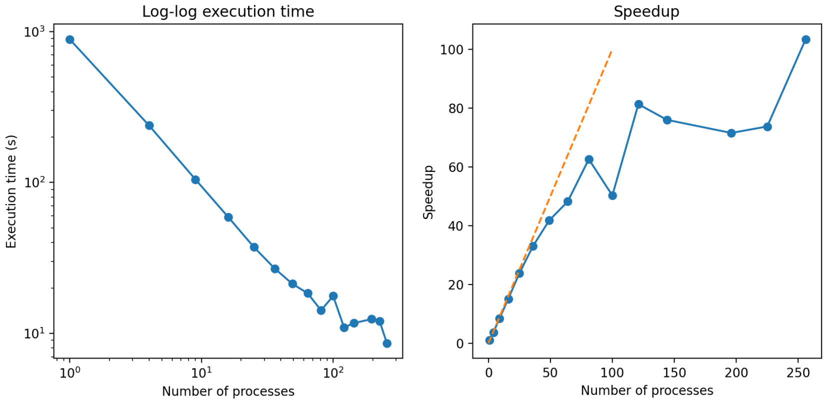
Performance test design

Set $X = Y$ so N is square.
Need to fulfill the condition

$$XY \mod N = 0$$

for all squares $XY \leq 256$.
The smallest number is 720720^2, too big! Excluding 13^2 we get 55440^2 which is ok.

Performance result

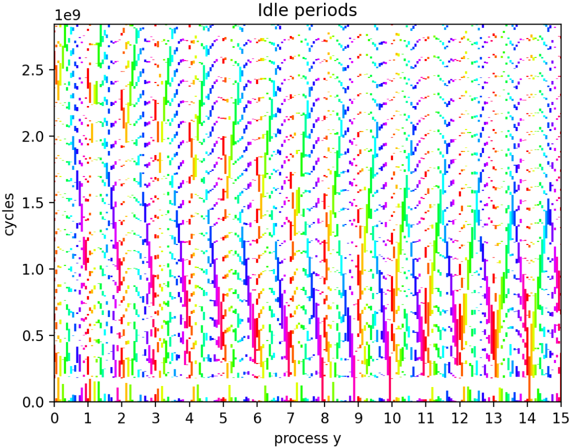


Linear speedup until $N = 49$, bad after $N = 100$

Idle period monitoring

- Replace `MPI_Neighbor_alltoall` with `MPI_Ineighbor_alltoall` and `MPI_Wait`
- Use `RDTSC` register to measure cycles spent waiting
- Write results to file `idle-x-y.bin`
- Transfer to laptop and visualize with matplotlib

Idle period visualization



Color indicates process x. (Also 3D demo)

Thank you!

Questions?

`https://github.com/jacwah/mpi-heat-diffusion`