UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MANIZALES

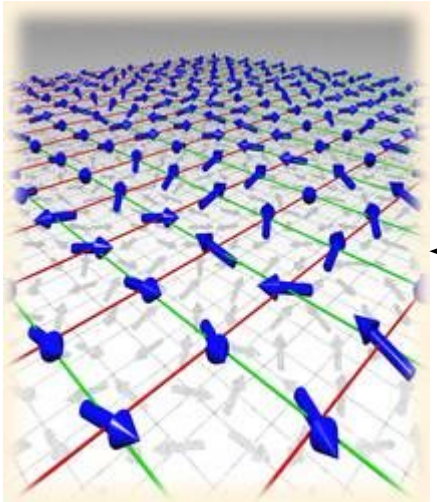
PCM
Computational Applications

Solving the atomistic LLG equation: A parallel approach

Juan David Alzate Cardona
Universidad Nacional de Colombia
Sede Manizales

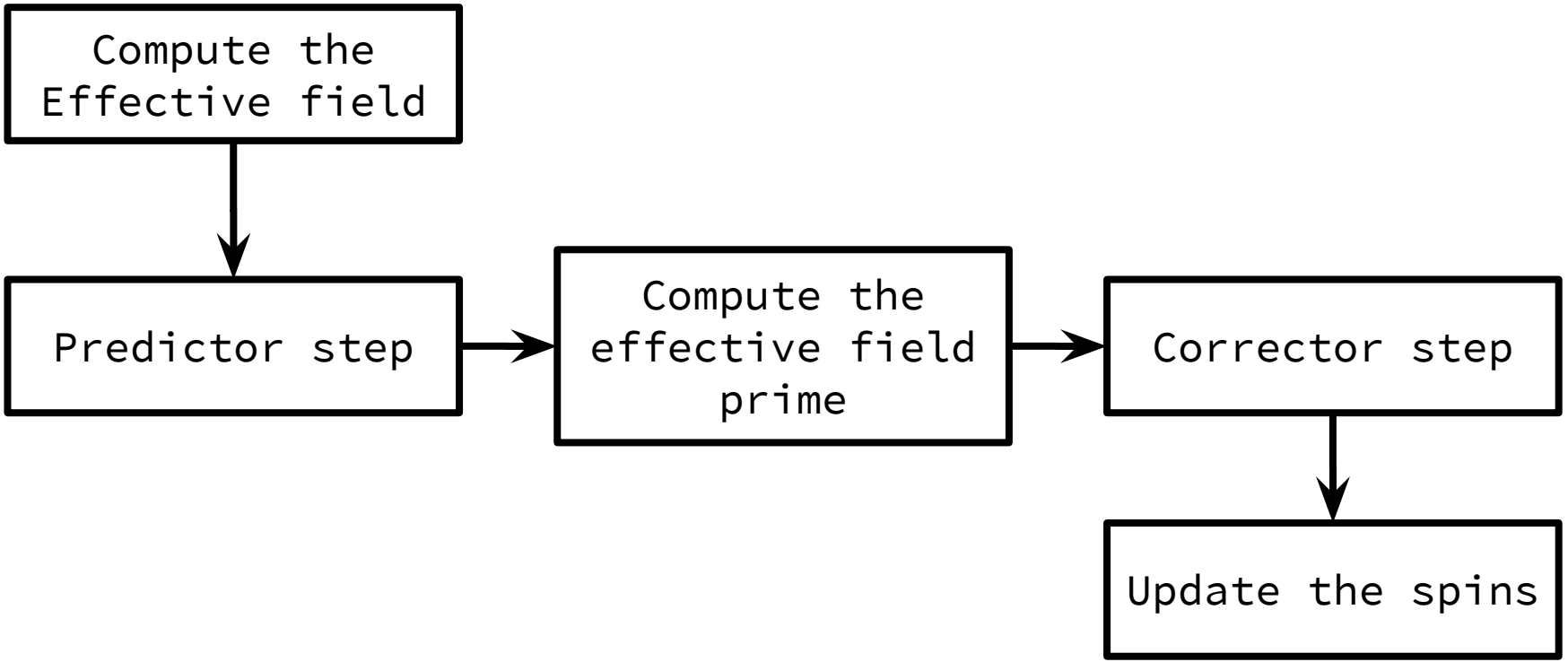
The LLG equation

$$\frac{d\boldsymbol{S}_i}{dt} = -\gamma_L \left(\boldsymbol{S}_i \times \boldsymbol{H}_i^{eff} \right) - \lambda \gamma_L \left[\boldsymbol{S}_i \times \left(\boldsymbol{S}_i \times \boldsymbol{H}_i^{eff} \right) \right]$$

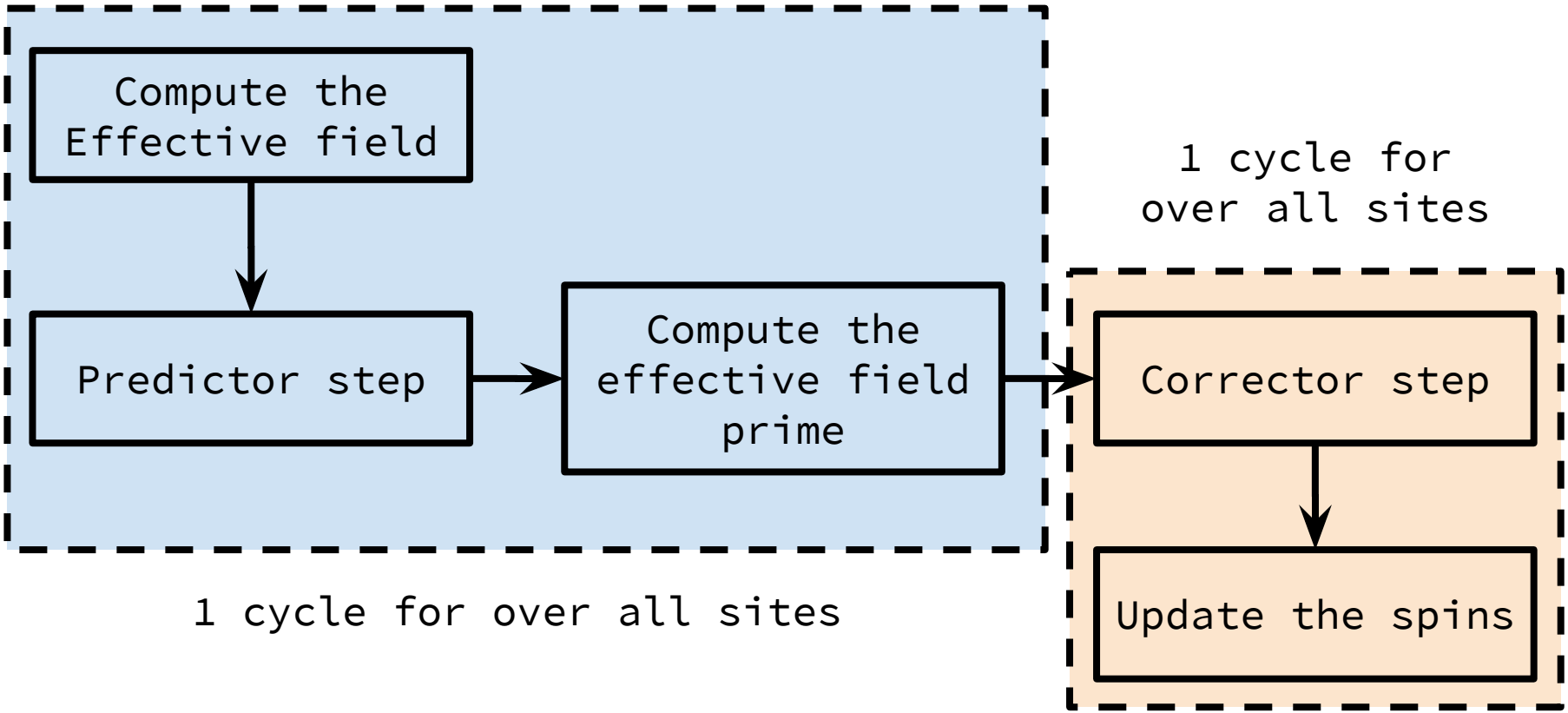


Classical Heisenberg model

Heun scheme


Compute the Effective field
Predictor step
Compute the effective field prime
Corrector step
Update the spins

Heun scheme


Compute the
Effective field
Predictor step
Compute the
effective field
prime
1 cycle for over all sites
1 cycle for
over all sites
Corrector step
Update the spins

Heun scheme (equations)

$$\boldsymbol{H}_i^{eff} = -\frac{1}{\mu_i}\frac{\partial \mathcal{H}}{\partial \boldsymbol{S}_i} + \boldsymbol{\Gamma}_i(t)\sqrt{\frac{2\lambda k_B T}{\gamma \mu_s \Delta t}}$$

$$\Delta \boldsymbol{S}_i = -\gamma_L\left(\boldsymbol{S}_i \times \boldsymbol{H}_i^{eff}\right) - \lambda\gamma_L\left[\boldsymbol{S}_i \times \left(\boldsymbol{S}_i \times \boldsymbol{H}_i^{eff}\right)\right]$$

$$\boldsymbol{S}_i' = \boldsymbol{S}_i + \Delta \boldsymbol{S}_i \Delta t$$

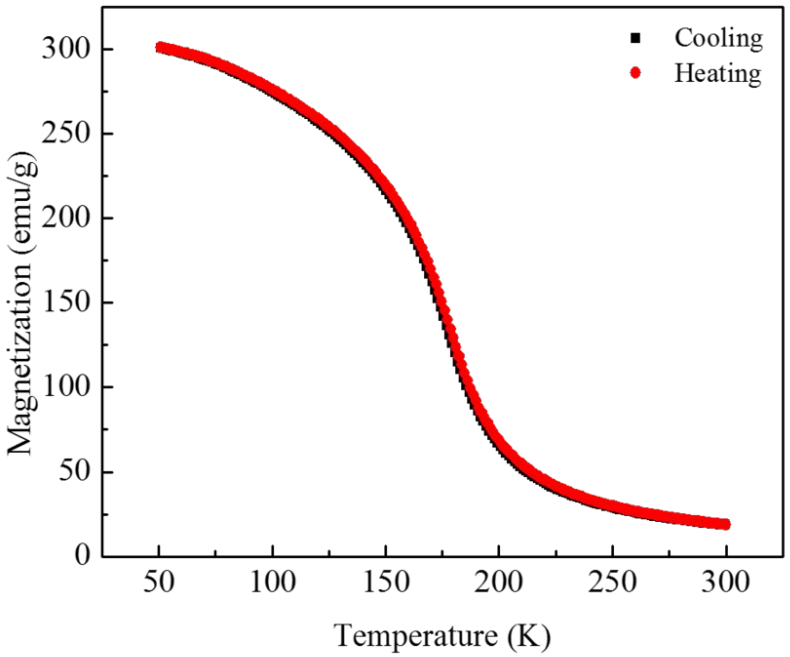
Predictor

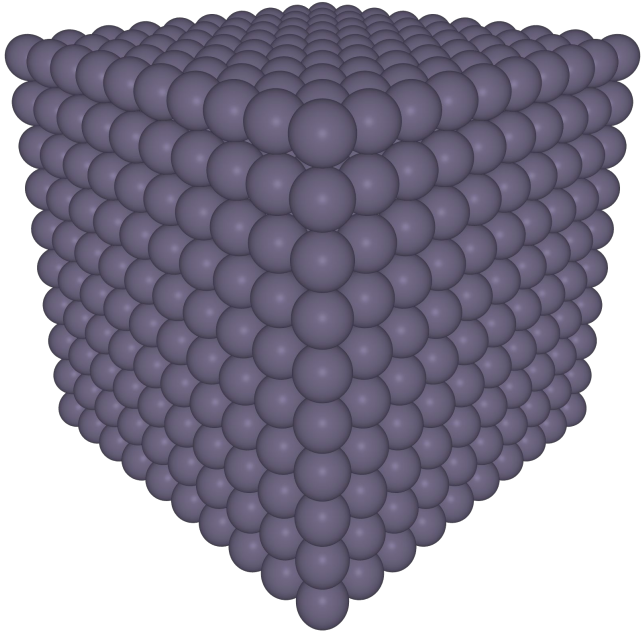
$$\Delta \boldsymbol{S}_i' = -\gamma_L\left(\boldsymbol{S}_i' \times \boldsymbol{H}_i^{eff'}\right) - \lambda\gamma_L\left[\boldsymbol{S}_i' \times \left(\boldsymbol{S}_i' \times \boldsymbol{H}_i^{eff'}\right)\right]$$

$$\boldsymbol{S}_i(t+\Delta t) = \boldsymbol{S}_i(t) + \frac{1}{2}\left[\Delta \boldsymbol{S}_i + \Delta \boldsymbol{S}_i'\right]\Delta t$$

Corrector

My problem

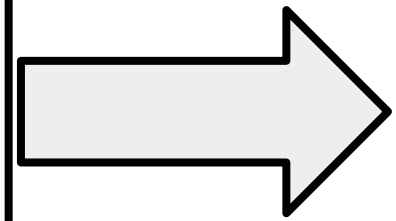




10x10x10 sites

Serial code

- Employing classes
- Pointer of pointers
- Passing by reference
- .
- .
- .
- So many confused things



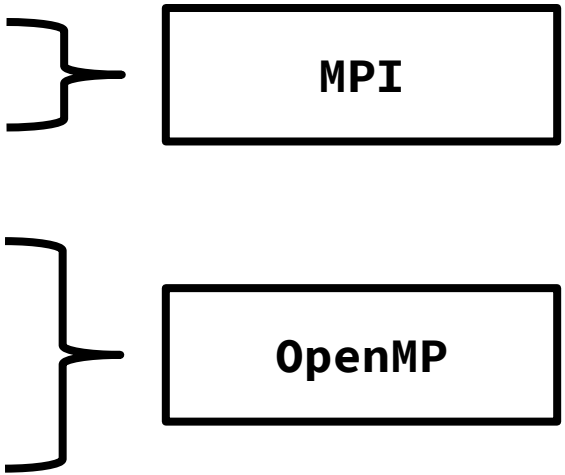
Spaghetti code

2.7 times faster

Dangerous

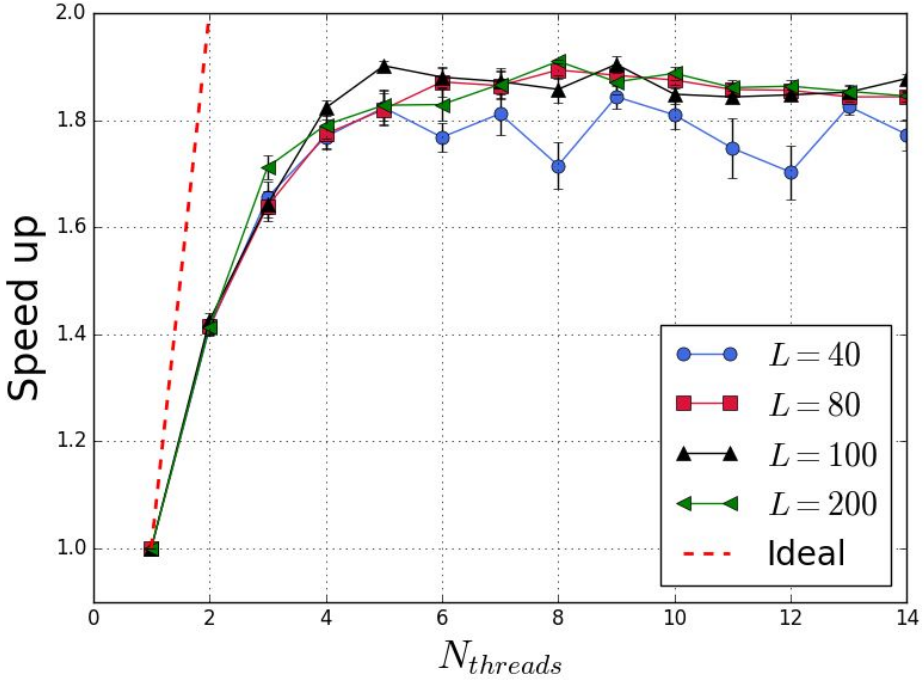
Parallelization

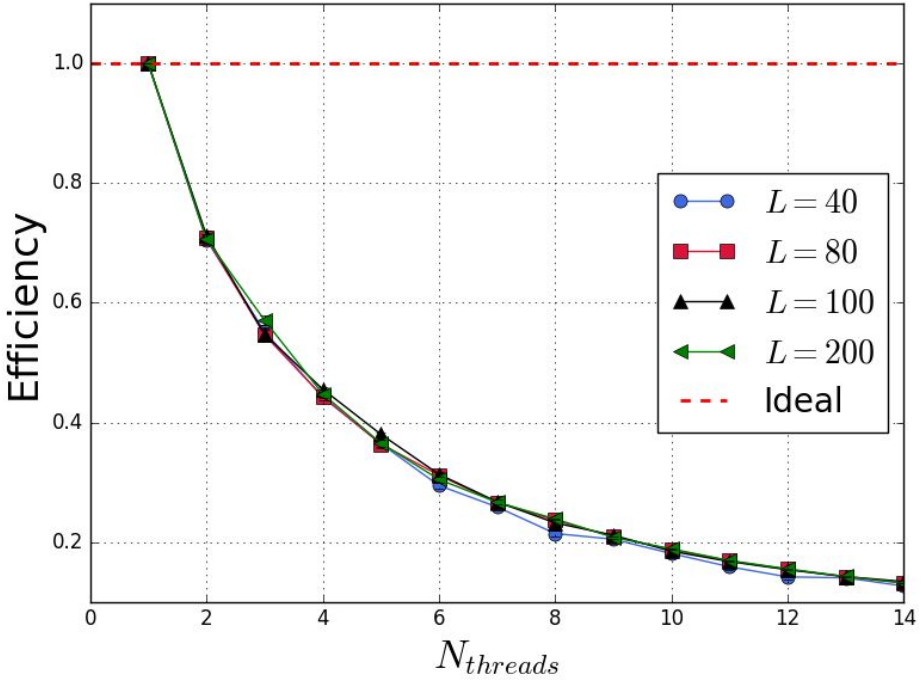
```
For loop in temperatures:

    For loop in time iterations:

        For loop in all sites: (corrector)

        For loop in all sites: (predictor)
```



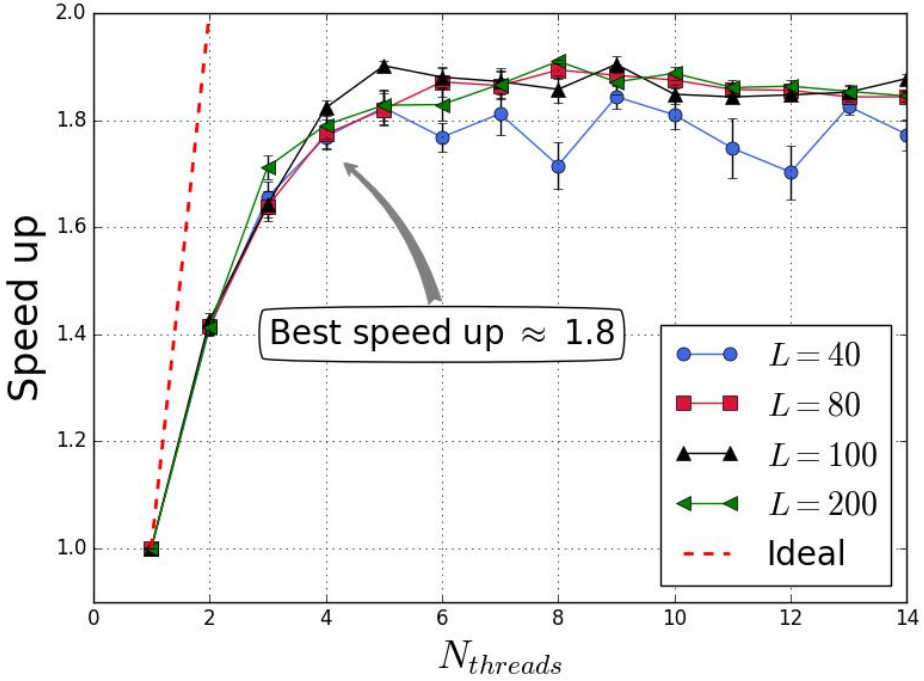
OpenMP results

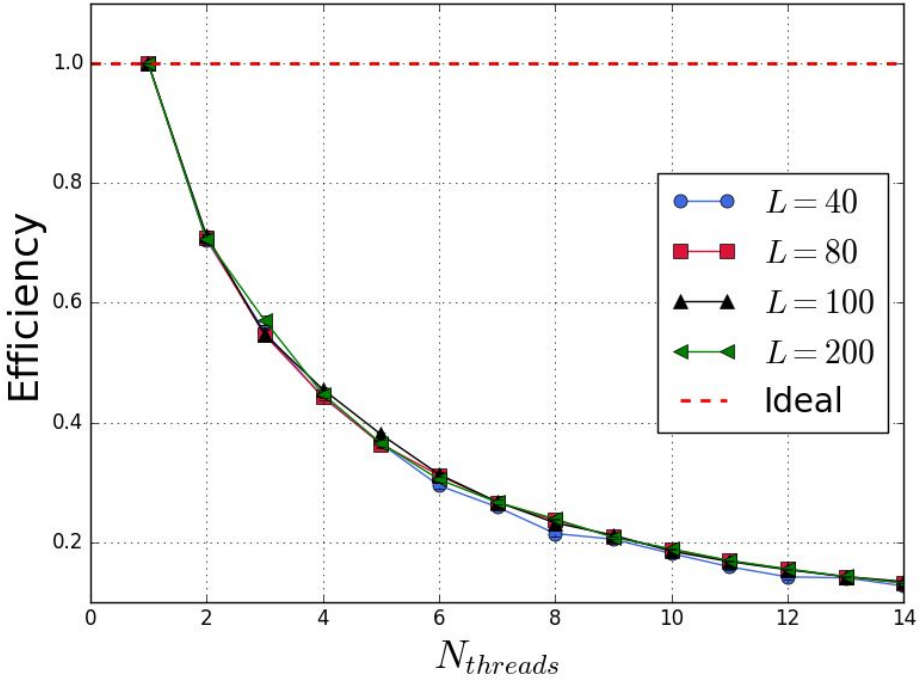
Processes = 1
1 temperature
10.000 time iter.



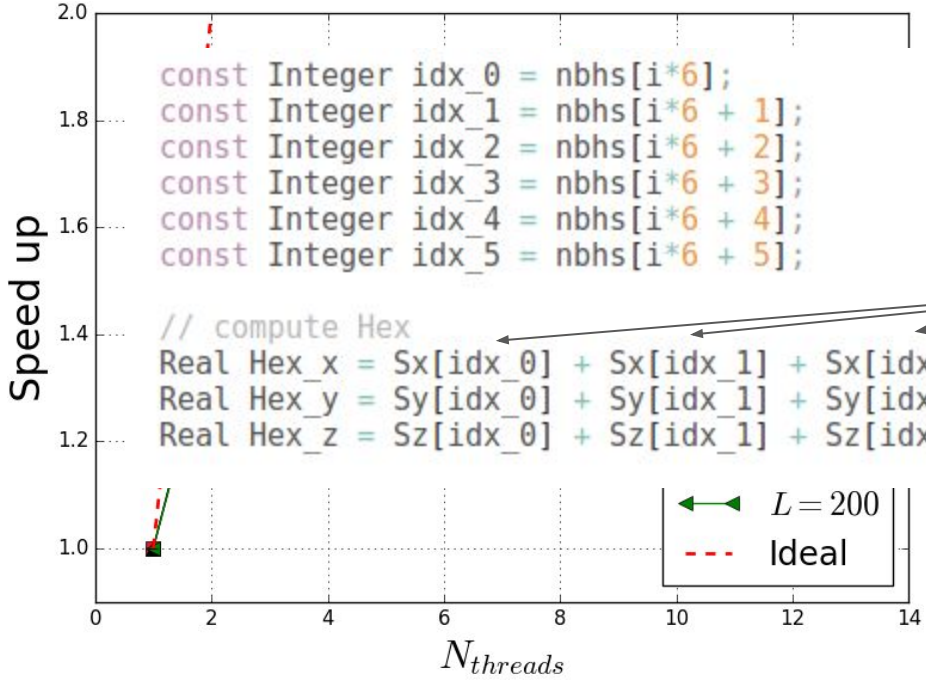


OpenMP results



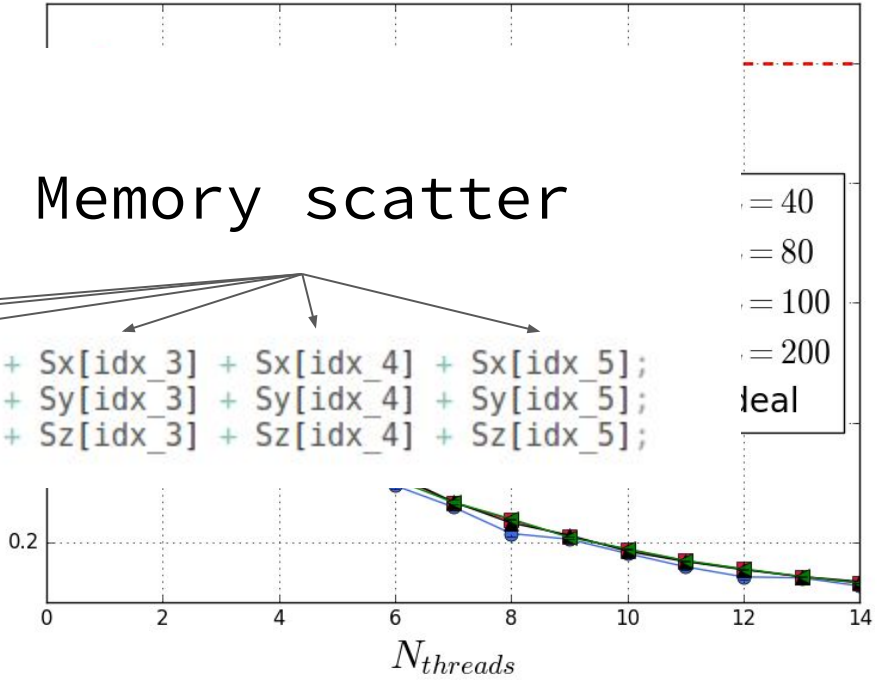


OpenMP results

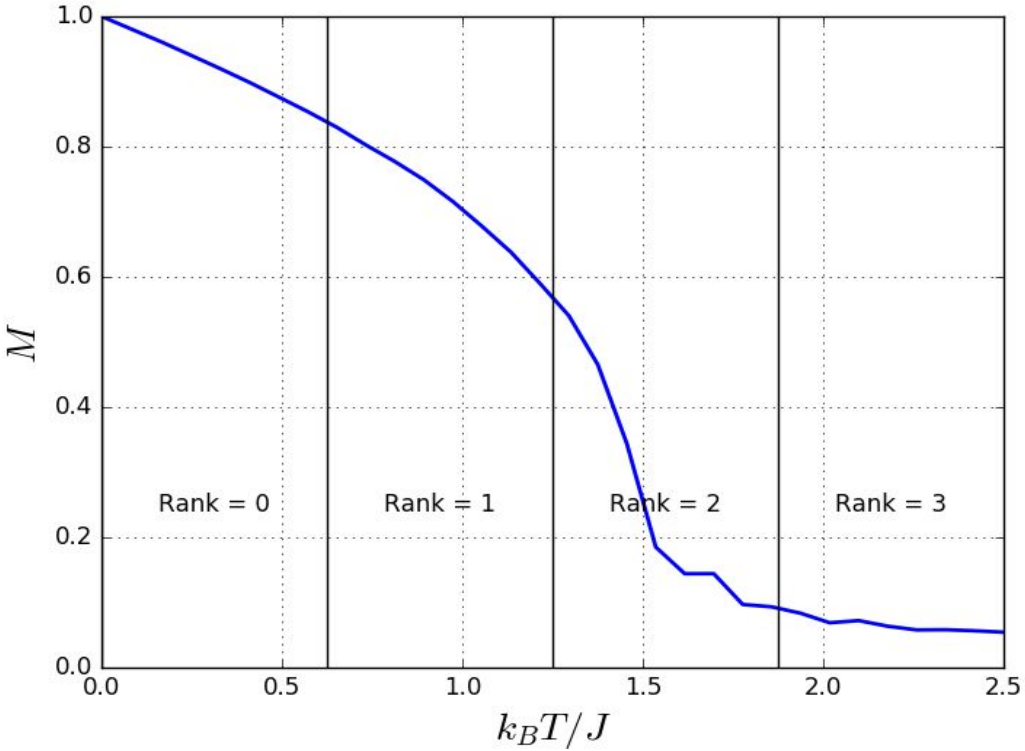



```
const Integer idx_0 = nbhs[i*6];
const Integer idx_1 = nbhs[i*6 + 1];
const Integer idx_2 = nbhs[i*6 + 2];
const Integer idx_3 = nbhs[i*6 + 3];
const Integer idx_4 = nbhs[i*6 + 4];
const Integer idx_5 = nbhs[i*6 + 5];

// compute Hex
Real Hex_x = Sx[idx_0] + Sx[idx_1] + Sx[idx_2] + Sx[idx_3] + Sx[idx_4] + Sx[idx_5];
Real Hex_y = Sy[idx_0] + Sy[idx_1] + Sy[idx_2] + Sy[idx_3] + Sy[idx_4] + Sy[idx_5];
Real Hex_z = Sz[idx_0] + Sz[idx_1] + Sz[idx_2] + Sz[idx_3] + Sz[idx_4] + Sz[idx_5];
```

Memory scatter

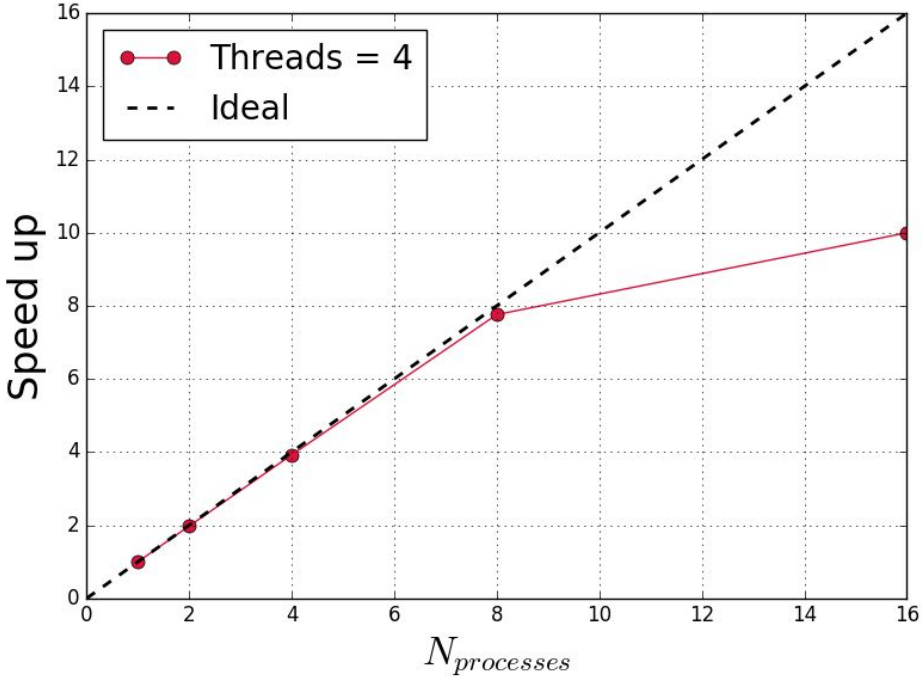


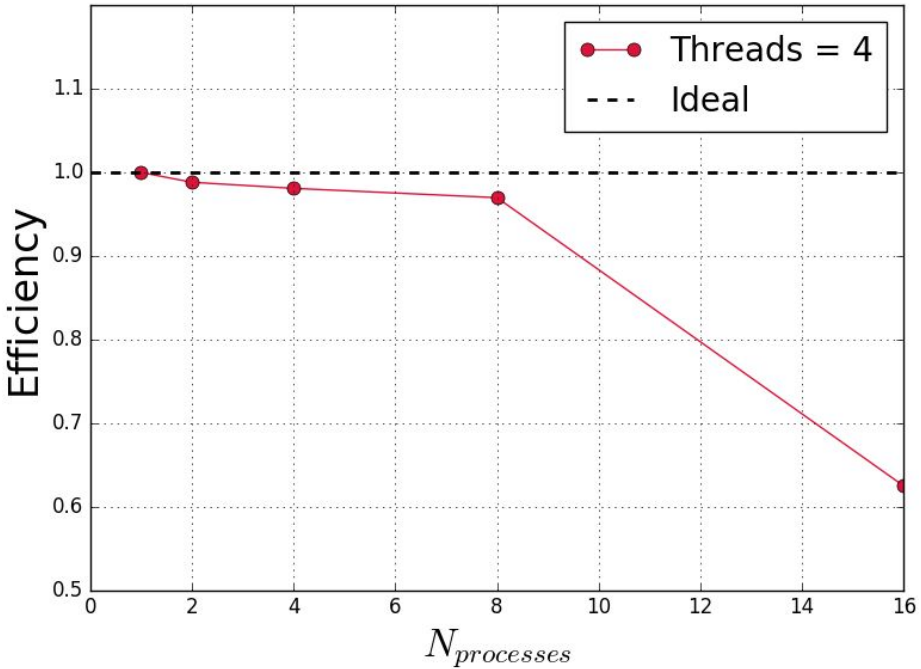
Dividing the temperature loop



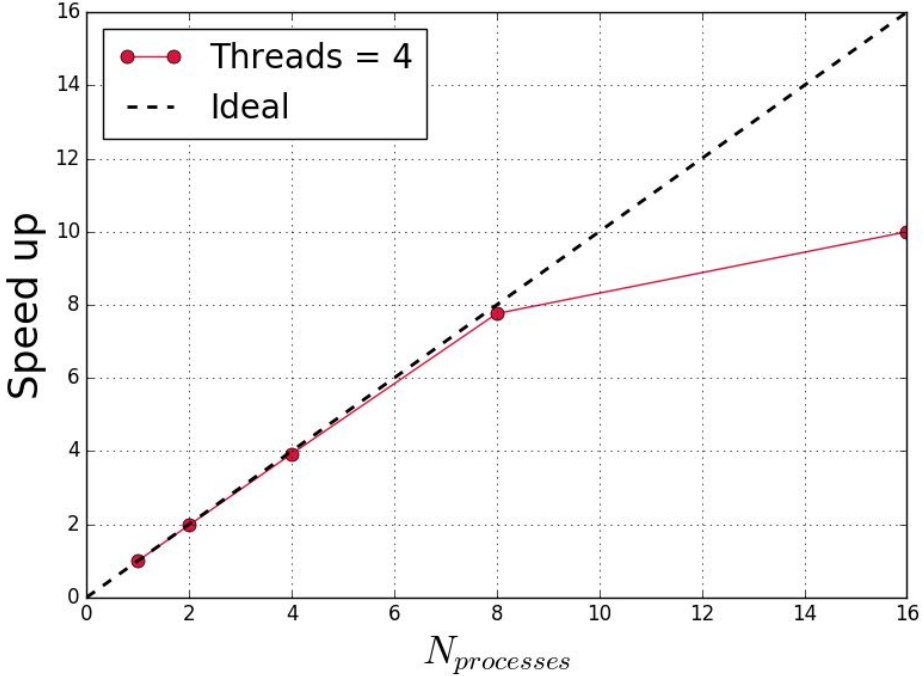
MPI results

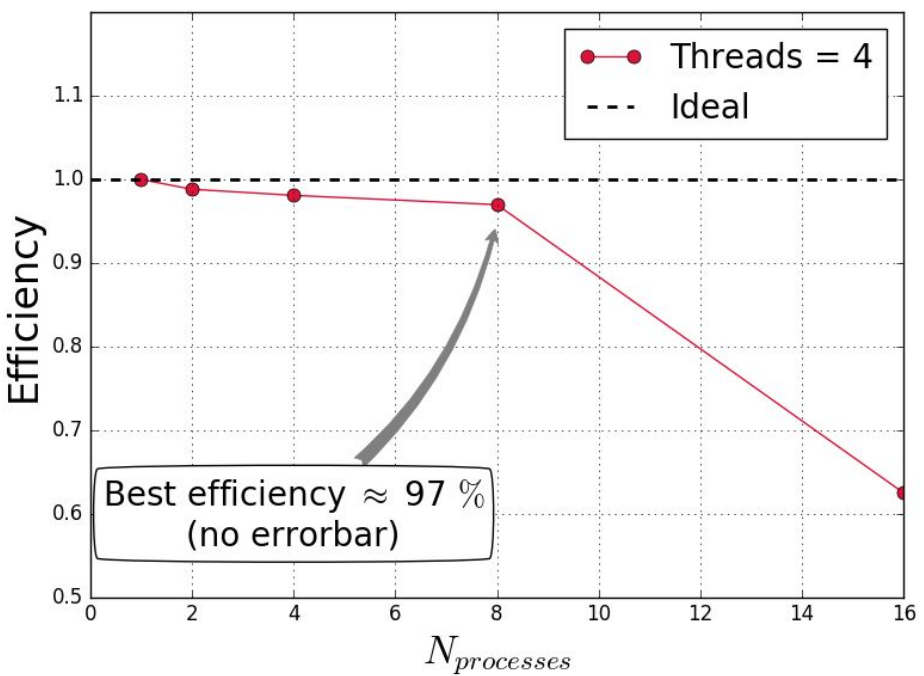
Threads = 4
L = 50
32 temperatures
100.000 time iter.



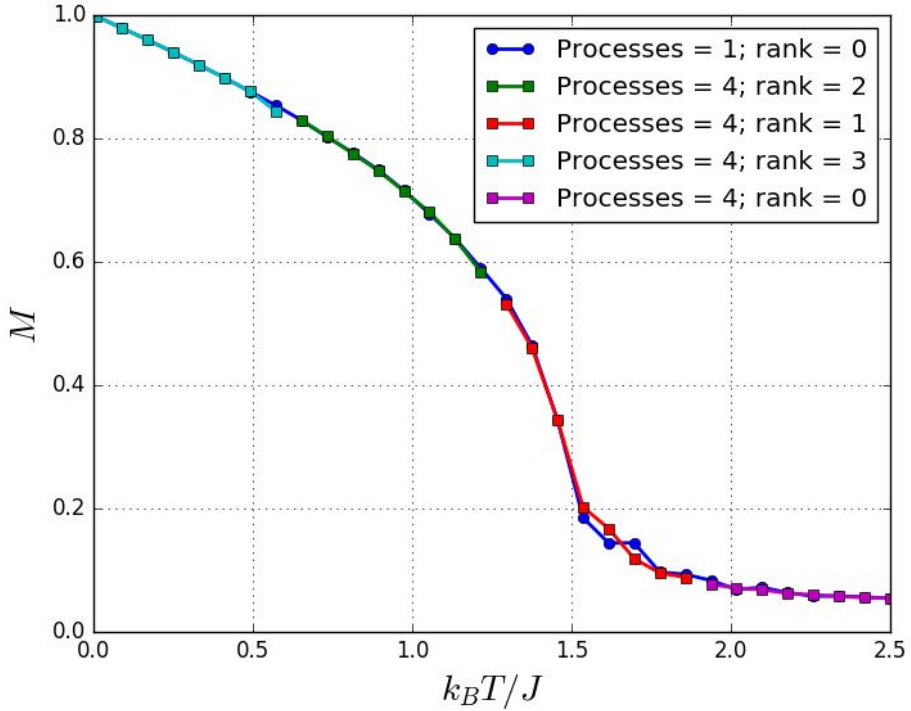


MPI results (consuming so much resources)





Magnetization vs Temperature



Conclusions

- The size is important in parallelization.
- Scatter patterns in memory make slow the programs.