Parallel Programming Lab Assignment: Diffusion Algorithm

Jan-Hendrik Niemann and Andreas Radke

October 26, 2017

1 Operations and Complexity

We chose the Diffusion Algorithm for our lab assignment. The main part of the algorithm is determined by the loops. We have a triple nested loop in *sum_values* (called twice) for summing all the values of the 3D-matrix, a triple nested loop in *init* (twice) for setting the initial values and the diffusion function with a four-fold nested loop (called once) which is the main part of the computation. Allocating the memory for the array with malloc is in regard to the other functions rather negligible.

Therefore we get for a 3D-matrix with the size of $N_x \times N_y \times N_z$ the number of operations:

$$2N_xN_yN_z + 2N_xN_yN_z + TN_xN_yN_z$$

where $T = \frac{N_xN_yN_z}{1000}$. The first two summands are for bigger input sizes way smaller and hence can be neglected. This leads us to the complexity of $\mathcal{O}(N_x^2N_y^2N_z^2)$.

2 Base Version's Performance and Correctness

3 Effect of Problem Size

4 Compilers

5 Optimized Version without SIMD

6 Optimized Version with SIMD

7 Performance Bottlenecks