

## Solution of Serie 9

*Results about CPU times and speedup plots are in the file `cpu.pdf`.*

**Exercise 1** 1. *Both loops are automatically parallelized. Notice that the second loop is not parallelized if we omit the option `-par-threshold0`.*

- 3. On  $p$  processors, the first method gives the result divided by  $p$ , since the variable `sdot` in the second loop is shared.*
- 4. CPU times show that the dot product is not very well parallelizable; automatic parallelization gives the better results.*

**Exercise 2** *the compiler can parallelize both loops without detecting any dependency.*

- 2. Without dependencies, one can just add the directive `#pragma omp parallel` for before the loop. The results are correct on any number of processors.*
- 3. The function `saxpy_` produces the best CPU times.*
- 4. The CPU times show a light performance enhancing when the number of processors increases. But the efficiency is small, because the operation works only on vectors.*

**Exercise 3** *This is a Blas3 operation, and one can check a good parallelization.*

- 1. The external loop for the matrix product is automatically parallelized.*
- 2. For obtaining a correct result, one must add the directive `#pragma omp parallel for shared(a,b,c) private(i,j,k)` before the external loop.*
- 3. You can plot the CPU times by using the script `parallel.sh`. The plots show an excellent speedup in both cases.*