

## Serie 9

*Remark: Programs shall be executed on altix! In the C-shell, the needed environment variables can be initialized by sourcing `~caussign/.cshrc`.*

**Exercise 1** Download the program `sdot.c` which does a dot product.

1. Compile `sdot.c` by: `iccbin -tpp2 -parallel -par-threshold0 -par-report3 -openmp . . . .`, which allows to get an automatic parallelization. Observe the compiler messages concerning the parallelization.
2. Parallelize `sdot.c` with OpenMP directives according to first and fourth methods described in the course. The compilation is done with `iccbin -tpp2 -openmp . . . .`
3. Check that the first method gives a wrong result on more than one processor.
4. Compare the results and the CPU times of the code of point 1. and the code for the fourth method on 1 to 16 processors.

**Exercise 2** Download `saxpy.c` which does the operation  $y \leftarrow y + x$ .

1. Compile the code with `iccbin -tpp2 -parallel -par-threshold0 -par-report3 -openmp . . . .` and check which loops are parallelized.
2. Convince yourself that there are no dependencies in both loops and parallelize the code with OpenMP directives.
3. Replace the second loop by a call to the Blas function `saxpy_` of GotoBlas library. In this case, one links by adding `-lgoto` to the compilation command.
4. Compare the CPU times of the three codes on 1 to 16 processors.

**Exercise 3** Download the code `sgemm.c` which does the operation  $C \leftarrow A * B$  for square random matrices of order  $N$ . Notice that the external loop for multiplying runs on the rows of  $C$ .

1. Compile the code for automatic paralelization with `iccbin -tpp2 -parallel -par-threshold0 -par-report3 -openmp -lm`. Execute the program on 1 to 16 processors.
2. Modify `sgemm.c` for parallelizing the external loop for the multiplication with an OpenMP directive by declaring the variables `a,b,c` as shared and the variables `i,j,k` as private . (For these private variables, each processor works on a local copy; for `j`, it is the default. But, if the indices of the internal loop are shared variables, the result will be wrong). Compile and run the program on 1 to 16 processors.
3. Plot the speedup curves  $S(p)$  for both programs.