Technische Universität Kaiserslautern
Fachbereich Informatik
RHRK
Priv.–Doz. Dr. Josef Schüle

Introduction in High Performance Computing
Sheet 2

Parallel Execution, Memory and Vectorization

Take the source from sheet 1, task 2. If you didn't use a function for the loop, please do it now and write this function into a separate file.

Change the code to utilize a global variable written in `main` for the constant value 2.3. Then change the code to allow for a simple compiler macro to change between float (single precision) and double precision. Introduce the achieved memory bandwidth in GB/s in your output. Use 3 different values for N for your tests - mathematically

$$N \in \{10,000; 1,000,000; 100,000,000\}$$

Run your tests on at least 2 different hardwares. You may use CPUs with AVX, AVX2 and AVX512. The later ones are reached with

salloc --reservation=schuele --exclusive ./your_executable

Compare the results in single and double precision. Why is single precision faster?

The code is easily vectorized. Inform yourself about the necessary compiler options for vectorization for `gcc` and `icc`. The site `https://elwe.rhrk.uni-kl.de/Tutorials/AVX/` may be helpful. Control the vectorization in the assembly output and compare the vectorization for different hardware.

Now change the code adding a new file or a new function to your code to execute as well the loop

```
for(i=0;i<N;i++) if(mod(i,2)==0) y[i]=2.3*x[i]+y[i]; else y[i]=2.3*x[i]-y[i];
```

You may optimize this of course.

Run the code vectorized as before, compare assembly, performance and memory bandwidth. Explain the differences.

Now copy the two functions and parallelize the loops with help of OpenMP. Select a specific hardware and run your code for 1 - number_of_cores_on_the_hardware. Inform yourself about possibilities to control the placement of threads on specific cores. Use different placement strategies for your tests.

Finally calculate the speedup by parallelization. It is defined

$$S(p) = \frac{time\_without\_parallelization}{time_parallel\_with\_p\_cores}$$

Try to explain your results.