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

Introduction in High Performance Computing
Sheet 3

Parallel Execution and Memory

Take the source for the `stream` benchmark, see the last exercise sheet. There you measured the memory bandwidth for a single core. Measure now (*Hint:* Learn to write a shell script, there you can run a program with different parameters and submit this script to a full node with 16 cores) the bandwidth for different numbers of cores.

- With help of the environment variable `OMP_NUM_THREADS` use $1, \ldots, 16$ cores and note the bandwidth in this setup.
- Read about the environment variable `KMP_AFFINITY` (intel compiler). Find out how to use a set of cores there the threads numbered $0, \ldots, 7$ are started on CPU 0 and the threads $8, \ldots, 15$ on CPU 1. Repeat with this setup the measurement above.
- Repeat the above measurement with all even threads on CPU 0 and all odd threads on CPU 1.

Depict your results graphically and comment the differences.

Program optimization and parallelization

Use the Intel compiler `icc` for this exercise. Download the specified source code (E03_source.c and E03_time.h) and compile and execute the code.

Now compile it again and use in addition the option `-g` that annotates the source code for further inspection. Start the program `vtune` with the command `amplxe-gui` on the head node. Make a hotspot analysis and find the most consuming parts of the program.

vtune and the advisor are licensed products. We do have only 5 licenses to be used at a time. Please close your sessions as soon as possible to allow other students to use this software.

Modify the code to speed up your version. Your improvements should reduce the time for an iteration far below 20 sec. Increase the number of iterations (variable `nIters` so that the overall runtime is in the range of 1 minute.

Below 10 sec for one iteration you may get help from the `advisor` tool: `advixe-gui`. Use this for further improvement and vectorize the code. Don't forget to increase `nIters`.

If your time per iteration is significant below 5 sec you may start the parallelization of the code with help of `OpenMP`. Run the program on $1, 2, 4, 8, 16$ cores with the same number of iterations (iterations for appr. 5 minutes for 1 core).

*H*int: The tool `jLSF` allows the submission of an `interactive` job. You may use this possibility (below 10 sec for one iteration) to start `vtune` or the `advisor` on a node different than a head node.