Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

Vectorisation labs: with SIMD and OpenMP pragmas

Objectives and learning goals

In this example we learn how to vectorise and parallelise regions using SIMD and OpenMP pragmas

- To enable the compiler to generate diagnostic information
- Understand the vectorisation performance
- Understand vectorisation reports
- To control memory allocation on Xeon Phi

1. Lab 1

- Compile the program without modifying the original code.
- Use the –no-vec flag to turn off the vectorisation:
    - $icc –no-vec –qopenmp –mmic nBody.c –o nbody.knc
    - $icc –qopenmp –xMIC-AVX512 nBody.c –o nbody.knl
- run the program with
    - for KNC: $micnativeloadex ./nbody.knc
    - On KNL: ./nbody.knl

and record execution time.

- Add the **vector report flags: -qopt-report=5**
    - $mpicc –no-vec –qopt-report =5 –qopenmp –mmic nbody.c –o nbody.exe
- Display the otimisation report file “nbody.optrpt” and try to understand the vectorised regions.
- Remove the –no-vec and –qopt-report flags and repeat the execution step above to record the execution time in the end. Check the performance results.
- Display the source code and switch on the parallelisation lines.
- Compile the program only with:
    - $icc –qopenmp –mmic nBody.c –o nbody.knc (KNC)
    - $icc –qopenmp –xMIC-AVX512 nBody.c –o nbody.knl (KNL)
    - and repeat the execution line above.
- Check the performance results.
- Change the environment variable: OMP_NUM_THREADS to: 20, 80,…up to 256 and run again.

What about the performance.

2. Lab 2

- Display nBody.c code and replace the LRZ WORK FOR YOU comments with SIMD and OpenMP calls.
- Display the Makefile to add the flag **-qopt-report=5 –qopt-report-phase:loop,vec** or just compile the code with the vector report flags
- Display the output reports and try to understand the vectorised regions, and remove unnecessary type conversions.
- Use a faster floating point semantics by adding the flag: **-fp-model fast=2**
- Compile again / run and check the performance results
- Set the following environment:

export MIC_KMP_AFFINITY=explicit,granularity=fine,proclist=[1-236:1]
export KMP_AFFINITY=granularity=fine,compact,1,0

Try now to understand the performance numbers observed for the host and native execution. And between the KNL and Standard Xeon.