Lab 3 - TFLOP Performance

Advanced Architectures

University of Minho

The Lab 3 focus on the development of efficient code for the Intel Knights Corner and Knights Landing computing devices by covering the programming principles that have a relevant impact on performance, such as vectorisation, parallelisation, and scalability. Use a cluster node with an Intel Knights Corner (by specifying the property `phi` in the job submission, e.g., `qsub -qmei -I -lnodes=1:ppn=?:phi,walltime=...`), and the Knights Landing compute server, which is accessed by `ssh compute-002-1` once in the SeARCH frontend.

This lab tutorial includes one homework assignment (HW 3.1) and three exercises to be solved during the lab class (Lab 3.x).

The goal of improving the performance of scientific applications is often to process more data per unit of time rather than to process a given data set faster (see Gustafson's Law). One example is the analysis of molecule docking, where faster applications are capable to process more time steps in a simulation. The goal is to test the throughput of these systems on a matrix dot product algorithm, where the matrix size should be proportional to the number of threads used. To achieve this goal, the first two exercises feature an implementation of vector dot product, whose workload increases automatically with the number of threads, to get acquainted with these computing systems. Moreover, the impact of vectorization and data alignment must be assessed for the best performing configuration of the code in each system.

Measure and document the throughput of every code variation in each exercise in a spreadsheet. Ideally, a K-best measurement heuristic should be used but, due to the short duration of this session, the best of three measurements should be used. A graph comparing the throughput of all implementations must be plotted at the end of the lab session.

To load the compiler in the environment use both commands:

Intel Compiler: `source /share/apps/intel/compilers_and_libraries_2017/linux/bin/compilervars_global.sh intel64`.

GNU Compiler (for several system libraries): `module load gnu/4.9.0`.

To copy the libraries and binaries to the Intel Xeon Phi see the commands in the `Makefile`.

3.1 The Xeon CPU Device and Knights Corner Accelerator

Goals: to develop skills in the design of parallel and vectorisable code for the Xeon CPU device and Knights Corner accelerator on native mode.

Consider the code provided in the attached file, with the vector dot product algorithm.

HW 3.1 Run the provided code on a dual-socket cluster node on the `mei` queue. Measure and record the best throughput using all cores in one and two devices, and finally with hyper threading.

The code is highly vectorisable but the `Makefile` explicitly excludes this compiler option. Remove this restriction and measure the code for the best performing thread configuration. Repeat this test with vectorisation and data alignement, and plot the results. How faster did you expect the code would be run and how fast it was? Can you explain this result?

OMP_NUM_THREADS: environment variable to set the number of threads.

Lab 3.1 Open two sessions on the compute node with a Knights Corner device (not the frontend this time): use one to compile the code and copy the binary and libraries to the device (see the `Makefile`), and use the other to access the device, using `ssh user@mic0`.

Run the code in native mode with 60, 120, and 240 threads. Note that you need to force a compact thread affinity, otherwise a 2-thread version will run in 2 separate cores (by default).

Measure, record, and compare the throughput between the multicore Xeon and the Knights Corner implementations. Test the Xeon Phi code with 1, 2 and 4 threads/core, with both compact and scatter thread affinity. Repeat the tests for the best thread configuration with vectorisation, vectorisation and data alignement, and plot the results. Justify the obtained results.

KMP_AFFINITY: environment variable to set the thread affinity.

3.2 The Knights Landing Compute Server

Goals: to develop skills in developing efficient code for the Knights Landing compute server.

Lab 3.2 Consider the parallel code from the previous exercise.

Access the Knights Landing compute server. Note that your home is already mounted on the server. Run the code with 64, 128, and 256 threads. Measure and record the throughput. Repeat the tests for the best thread configuration with vectorisation, vectorisation and data alignement, and plot the results. How does it compare to the previous systems? Does the vectorization have the expected impact on performance?

Setup the environment by running the following commands:

```
export PATH=/share/apps/gcc/4.9.0/bin:$PATH

export LD_LIBRARY_PATH=/share/apps/gcc/4.9.0/lib:$LD_LIBRARY_PATH

source /opt/intel/compilers_and_libraries/linux/bin/compilervars.sh intel64
```

3.3 Matrix Dot Product on Xeon Systems

Goals: to comprehend the concepts of the **Lab 3.1** and **Lab 3.2** by implementing an efficient matrix dot product code for the test systems.

Lab 3.3 Develop a simple parallel implementation of the matrix dot product algorithm. Study the impact of vectorisation and data alignment on the systems used in the previous exercises.