ACCELERATOR BASED PROGRAMMING
UPPSALA UNIVERSITY
FALL 2023

EXERCISE 4: FIRST STEPS IN KOKKOS

This exercise is a preparation for the fourth assignment.

1. Exercise Goal. The goal of this exercise is to get started with Kokkos. The goal is to run the exercises 01 and 02 listed athttps://github.com/kokkos/kokkos-tutorials/tree/main/Exercises on both the CPU and the GPU of UPPMAX/Snowy.

The tasks are as follows:

- Familiarize yourself with Kokkos by looking at the lecture material, and in particular the Kokkos guide https://github.com/kokkos/kokkos-tutorials/blob/main/LectureSeries/KokkosTutorial_01_Introduction.pdf.

- On your computer or UPPMAX, go to your favorite directory, e.g.
  `${HOME}/Kokkos`
  and download Kokkos:
  `git clone https://github.com/kokkos/kokkos`

- Download the Kokkos tutorials
  `git clone https://github.com/kokkos/kokkos-tutorials`

- Download the Kokkos tutorials
- We also need to load the compilers for gcc and nvidia
  ```
  module load gcc/8.4.0
  module use /sw/EasyBuild/snowy/modules/all/
  module use /sw/EasyBuild/snowy-GPU/modules/all/
  module use python_ML_packages/3.9.5-gpu
  module load CUDA/11.7.0
  ```
- Go to the first exercise
  `cd kokkos-tutorials/Exercises/01/Begin`

- Inspect the `Makefile` and adjust the paths to your system. If you have Kokkos in a directory `Kokkos/kokkos` of your home directory, you do not have to adjust it.

- To compile for the CPU, you do not have to do further adjustments and you can just run `make -j8`. To compile for the GPU, you need to either pass options to the command line when calling `make` or adjust the settings in the `Makefile`. To compile for CUDA, change the second line to

  `KOKKOS_DEVICES = "Cuda"`
  and line 14 to
  `KOKKOS_ARCH = "TURING75"`
  Also, to code on the CPU nodes of Snowy you need to adjust the CPU architecture as well, line 19,
  `KOKKOS_ARCH = "SNB"`

- Follow *exercise 1* and initialize Kokkos. Compare with the respective code in the solution. This is a CPU only code we paralellize with OpenMP. A possible runscript could look like this

```
#!/bin/bash -l

#SBATCH -A uppmax2023-2-36
#SBATCH -M snowy
#SBATCH -p node
#SBATCH -N 1
#SBATCH -t 0:10:00
#SBATCH -J Exercise1
#SBATCH -D ./

export OMP_PROC_BIND=spread OMP_PLACES=threads
echo "8 threads"
export OMP_NUM_THREADS=8
./01_Exercise.host -N 13
./01_Exercise.host -N 14
./01_Exercise.host -N 15
./01_Exercise.host -N 16

echo "16 threads"
export OMP_NUM_THREADS=16

./01_Exercise.host -N 13
```

- Work on *exercise 2* regarding the memory spaces. Note that this will use unified memory between CPU and GPU, as seen by the `force_uvm` CUDA option. Run the CUDA code for this example and compare bandwidht for various sizes of the matrix