1. Introduction

This lab aims at porting the following SGEMM function:

void sgemm(int n, float alpha, float v1[n][n], float v2[n][n], float beta, float vout[n][n])

to a CUDA device thanks to HMPP directives.

SGEMM stands for Single precision General Matrix Multiply. It performs the following operation:

C=alpha . A*B + beta . C

A, B, C being matrices and alpha, beta being scalars

You will “play” with HMPP directives to drive codelet production and see how it affects code efficiency.

2. Learning goals

The goals of this lab are:

To become familiar with OpenHMPP directives.

To observe the effect of directives on performances when using a CUDA device.

3. Assignment

You are provided with source code files. You are asked to add the proper OpenHMPP directives in order to drive code production. When using CUDA, make sure that your environment is set.

Version	Source files	Remarks
SGEMM1	sgemm1.c sgemm1-codelet.c	Simple HMPP application (at-call transfer policy)
SGEMM2	sgemm2.c sgemm2-codelet.c	Transfers optimization (manual transfer policy)

You will be able to compare performances of different codelet versions.

3.1. Guidelines

The version SGEMM1 corresponds to a simple HMPP application. The version SGEMM2 corresponds to an HMPP application with improved transfers. Follow the different TODO instructions given in the source files and insert OpenHMPP directives.

For sgemm1.c and sgemm1-codelet.c:

- (1) Write the codelet directive of label sgemm1:
  - For CUDA target

    - All parameters should follow the at-call transfer policy
    - Array parameters should be mirrored on the device
- (1) Write the corresponding callsite to execute the SGEMM on the GPU
- (2) Write allocation and de-allocation directives for the arrays

For sgemm2.c and sgemm2-codelet.c:

- Write the codelet directive of label sgemm2:
    - For CUDA target
    - Alpha and beta will follow atfirstcall transfer policy
    - N will follow the at call transfer policy
    - Vin1, vin2 and vout will follow the manual transfer policy
    - Array parameters should be mirrored on the device
- Write the corresponding callsite to execute the SGEMM on the GPU
- Write allocation and de-allocation directives for the arrays
- Write transfer directives for the arrays

3.2. Notice performance improvements of transfer optimizations

You can compare the performances of two versions the SGEMM depending on the input matrix size.

Compile those applications with:

```
$ make sgemm1.exe

$ make sgemm2.exe
```

Execute them with:

```
$ export RUNCMD=srun

(to enable execution on a compute node)

$ make sgemm1.run

$ make sgemm2.run
```

To enable the logger and follow the operations of HMPP runtime application:

```
$ export HMPPRT_LOG_LEVEL=info
```

Display graph with the following command:

```
$ make display
```