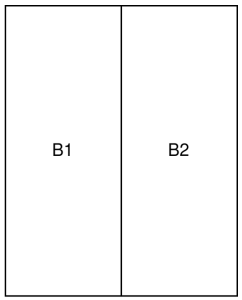
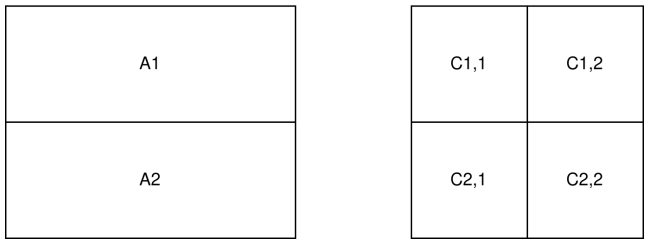
Presentation of TER project

Atte Torri - atte.torri@universite-paris-saclay.fr

Matrix multiplication with 1D tiling

Doing a matrix multiplication without taking into account transpositions, with A and B tiles in 1D and C in 2D.
Note: full operation is $\mathcal{C} = \alpha \times \mathcal{AB} + \beta \times \mathcal{C}$





Matrix multiplication with 1D tiling

- Assume column major ordering (contiguous columns)
- Each matrix has the same tile size (this defines the number of tiles)
- At least fill, compute gemm, check result and print functions
- Use StarPU tasks as much as possible (filling, computing, etc)

Code organisation

The skeleton code is organised as follows

- `gemm.cpp` contains the `main` function and parameters with which the code should work.
- `matrix.*` and `tile.*` contain skeleton code for the matrix and tile classes.
- `codelets.*` and `kernels.*` are for everything that StarPU needs for tasks.
- `blas*` files contain BLAS/LAPACK functions (such as `gemm`) that you will need during the TER project.

It is recommended to stick with this organisation, but you are free to change it or improve upon it, as long as it remains organised.
Note: The cleanliness, organisation and legibility of your code is a factor in grading.

Using BLAS

```
template <typename DataType>
struct blas {
  static void gemm(
    char transA,
    char transB,
    int m, int n, int k,
    DataType alpha,
    DataType* A, int ldA,
    DataType* B, int ldB,
    DataType beta,
    DataType* C, int ldC);
};
```

Initialise and shutdown

```
#include <starpu.h>

starpu_init(NULL);

/* Computation */

starpu_shutdown();
```

Codelets

Static and dynamic creation of codelets (Note: the codelet must not be freed before the end of the task)

```
struct starpu_codelet cl = {
    .cpu_funcs = { cpu_func },
    .nbuffers = 0
};


struct starpu_codelet cl;
starpu_codelet_init(&cl);
cl.cpu_funcs[0] = cpu_func;
cl.nbuffers = 0;
```

Tasks

Multiple ways to insert tasks

```
// Create, fill and submit
struct starpu_task *task = starpu_task_create();
task->cl = &cl;
task->handles[0] = vector_handle;
task->cl_arg = &factor;
task->cl_arg_size = sizeof(factor);
starpu_task_submit(task);

// Or use the helper function
starpu_task_insert(&cl, STARPU_RW, vector_handle,
    STARPU_VALUE, &factor, sizeof(factor), 0)
```

Tasks

To unpack arguments of type STARPU_VALUE in kernels you can use one of the starpu_codelet_unpack_* functions

```
void func_cpu(void *descr[], void *_args)
{
int ifactor;
float ffactor;
starpu_codelet_unpack_args(_args, &ifactor, &ffactor);
}
```

Tasks

- Ensure tasks are long enough to compensate for StarPU overhead, but short enough to have a lot of tasks.
- You can set a variable number of handles as arguments
- You can set callbacks on tasks to be executed once the task is done
- StarPU has many other tasks utilities some might be useful to modify and duplicate tasks, etc

Data management

Registering a matrix

```
float *matrix;
starpu_malloc(&matrix, width*height*sizeof(float));
starpu_data_handle_t matrix_handle;
starpu_matrix_data_register(&matrix_handle,
    STARPU_MAIN_RAM, (uintptr_t)matrix, width, width,
    height, sizeof(float));


starpu_matrix_data_register(&matrix_handle, -1, NULL,
    width, width, height, sizeof(float));
```

Data management

- Partitioning data with StarPU helper functions
- Possibility of creating custom data interfaces
- Commutative data accesses in tasks with `STARU_COMMUTE`
- Workspace buffers in tasks with `STARPU_SCRATCH`

Controlling the environment

- `STARPU_ENABLE_STATS` - Enable statistics
- `STARPU_WORKER_STATS` - Gives some more stats on workers executing tasks

Next

Start working on the TER project. Compiling the project is explained in the `README.md`

```
ssh qdcster_XX@chome.metz.supelec.fr
salloc --partition gpu_tp_resa --reservation M1QDCS_TERSTARPU12
       --exclusive --time 4:00:00
```

For next week read at least chapters 18, 32, 33 of the StarPU documentation
https://github.com/TER-StarPU/ter-starpu-gemm/blob/main/docs/starpu-documentation.pdf