GPU Programming II

Andrey Alekseenko

KTH Royal Institute of Technology & SciLifeLab

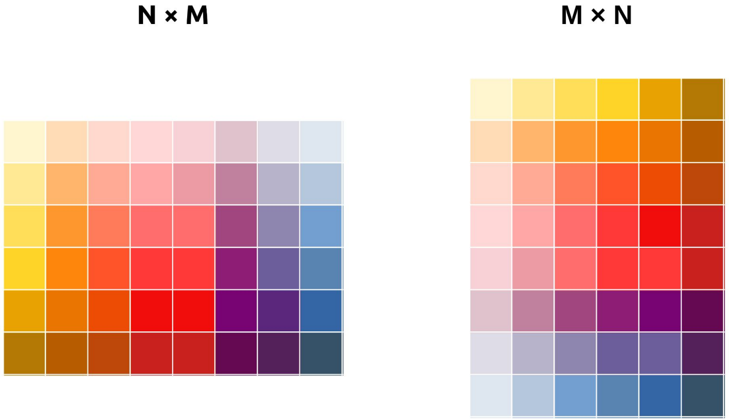
Refresher: Getting started on Dardel (lazy)

```
$ ssh abcd@dardel.pdc.kth.se
$ ml PDC/22.06 adaptivecpp/23.10.0-cpeGNU-22.06-rocm-5.3.3-llvm
$ export SLURM_ACCOUNT=edu23.aqti SLURM_TIMELIMIT=00:05:00
$ export SLURM_PARTITION=gpu SLURM_RESERVATION=lab-2023-12-05
$ srun acpp-info -l
=================Backend information===================
Loaded backend 0: OpenMP
  Found device: hipSYCL OpenMP host device
Loaded backend 1: HIP
  Found device: AMD MI250X
  Found device: AMD MI250X
  Found device: AMD MI250X
  Found device: AMD MI250X
```

Refresher: Getting started on TCBLab (interactive mode)

```
$ ssh wsXX@login.tcblab.org
$ salloc # Run only once; if disconnected, ssh to the same node
salloc: Nodes gpuYY are ready for job
$ ssh gpuYY
$ module load adaptivecpp/23.10.0-clang16-cuda12.1
$ acpp-info -l
=================Backend information===================
Loaded backend 0: CUDA
  Found device: NVIDIA RTX A5000
Loaded backend 1: OpenMP
  Found device: hipSYCL OpenMP host device
```

Exercise: Matrix transpose



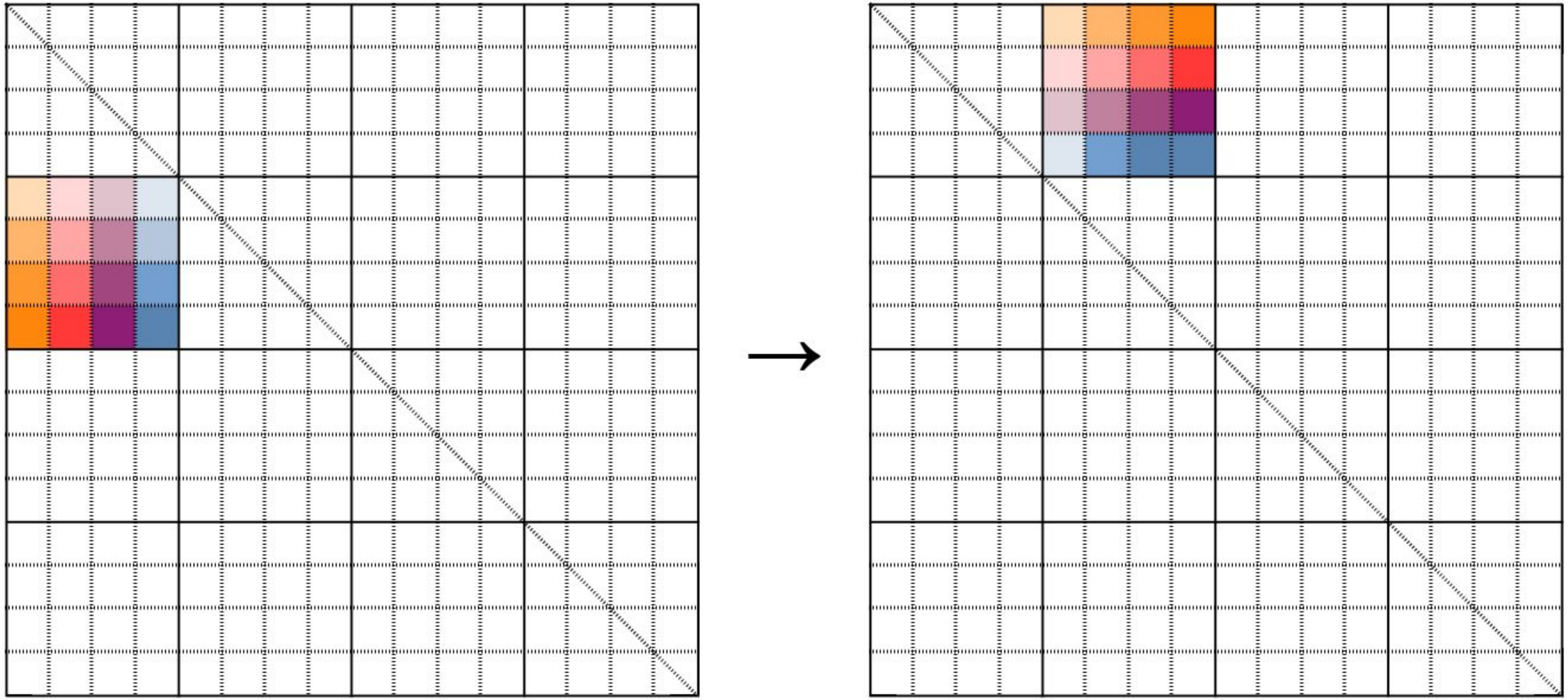
- Build and run `transpose_matrix_v0.cpp`
- For now, it just copies the matrix
  - The self-check will fail!
- Look at the source code
  - See the new constructs we learned today
- Modify the code to transpose the matrix
- Use local memory to achieve coalesced memory access
- Solutions:
  - Naive: `transpose_matrix_v1.cpp`
  - Optimized: `transpose_matrix_v2.cpp`

Exercise: Matrix transpose

```
auto copyKernel(const float *in, float *out, int width, int height) {
  return [=](sycl::nd_item<2> item) {
    int x_index = item.get_global_id(1);
    int y_index = item.get_global_id(0);
    int index = y_index * width + x_index;
    out[index] = in[index];
  };
}
```

```
int index  = Y * width  + X // :)
int indexT = X * height + Y // :(
```

Exercise: Matrix transpose



Exercise: Matrix transpose

```
int index  = Y * width  + X //  :)
int indexT = X * height + Y //  :(
```

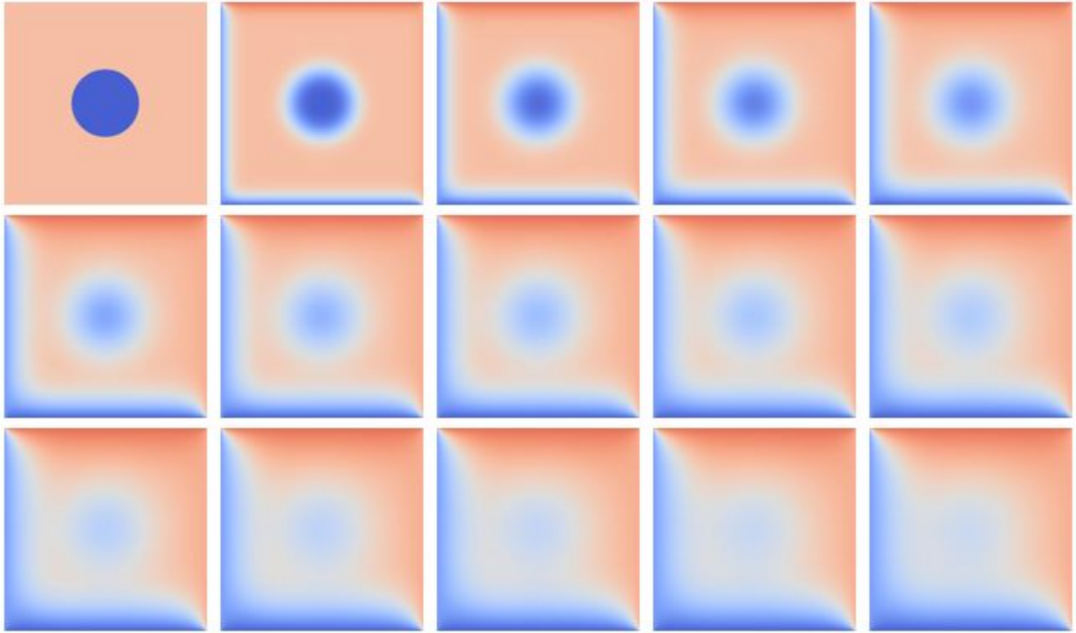
```
auto transposeKernel(sycl::handler &cgh,
                     const float *in, float *out,
                     int width, int height) {
  sycl::local_accessor<float, 1> tile{{tile_dim * tile_dim}, cgh};
  return [=](sycl::nd_item<2> item) {
    int x_tile_index = item.get_group(1) * tile_dim;
    int y_tile_index = item.get_group(0) * tile_dim;
    int x_local_index = item.get_local_id(1);
    int y_local_index = item.get_local_id(0);

    tile[...] = in[...]; // TODO: Coalesced read from in
    item.barrier();
    out[...] = tile[...]; // TODO: Coalesced write to out
  };
}
```

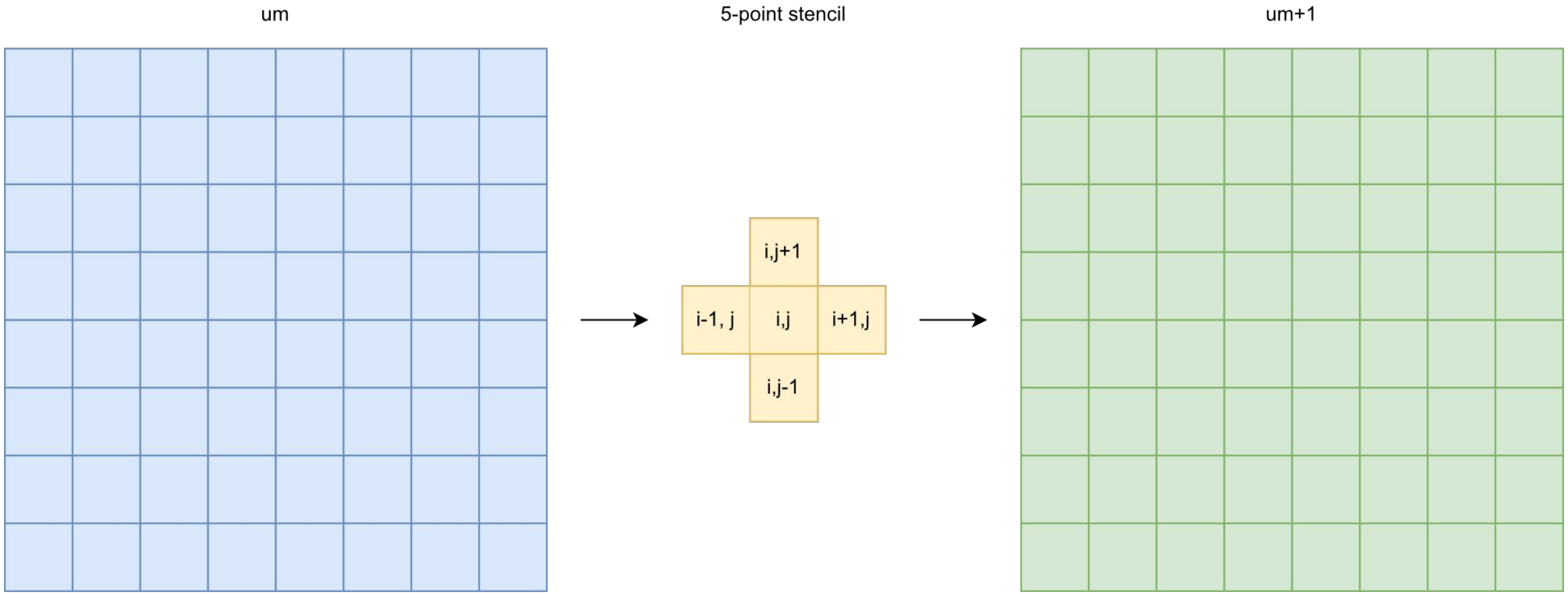
Stencil computation

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial x^2} \right)$$

- Heat flow in a 2D area
- Continuous eqn. is discretized in space and time
  - We’ll ignore the finer aspects of it
- Higher spatial resolution necessitates smaller time steps



Stencil computation



Code

- `/cfs/klemming/home/a/andreyal/Public/stencil/` (Dardel)
- `/mnt/cephfs/home/aqtivate-ws/stencil/` (TCBLab)
    - `openmp/`
        - On Dardel, `module load PrgEnv-amd/8.3.3` first
        - `core.cpp, main.cpp`: OpenMP on CPU
            - Use `OMP_NUM_THREADS=4` env. variable to set number of CPU threads to 4
        - `core-naive.cpp`: OpenMP on GPU, naive version
        - `core-data.cpp, main-data.cpp`: OpenMP on GPU, fast version
    - `sycl/`
        - `core-naive.cpp, main-naive.cpp`: SYCL on GPU, naive version
        - `core-data.cpp, main-data.cpp`: SYCL on GPU, fast version
            - (you will get runtime warnings, they are harmless)