CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Introduction to GPUs in HPC

Sebastian Keller, Prashanth Kanduri
and Ben Cumming, CSCS

2D and 3D Launch Configurations

Launch Configuration

- So far we have used one-dimensional launch configurations:
  - Threads in blocks indexed using `threadIdx.x`.
  - Blocks in a grid indexed using `blockIdx.x`.
- Many kernels map naturally onto 2D and 3D indexing:
  - e.g. Matrix-matrix operations;
  - e.g. Stencils.

Full Launch Configuration

Kernel launch dimensions can be specified with `dim3` structs

```
kernel<<<dim3 grid_dim, dim3 block_dim>>>(...);
```

- `dim3.x`, `dim3.y` and `dim3.z` specify the launch dimensions;
- Can be constructed with 1, 2 or 3 dimensions;
- Unspecified `dim3` dimensions are set to 1.

launch configuration examples

```
// 1D: 128x1x1 for 128 threads
dim3 a(128);
// 2D: 16x8x1  for 128 threads
dim3 b(16, 8);
// 3D: 16x8x4  for 512 threads
dim3 c(16, 8, 4);
```

The `threadIdx`, `blockDim`, `blockIdx` and `gridDim` can be treated like 3D points via the `.x`, `.y` and `.z` members.

matrix addition example

```
__global__
void MatAdd(float *A, float *B, float *C, int n) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if(i<n && j<n) {
        auto pos = i + j*n;
        C[pos] = A[pos] + B[pos];
    }
}
int main() {
    // ...
    dim3 threadsPerBlock(16, 16);
    dim3 numBlocks(n / threadsPerBlock.x, n / threadsPerBlock.y);
    MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
    // ...
}
```

Exercise: Launch Configuration

- Write the 2D diffusion stencil in `diffusion/diffusion2d.cu`
- Set up 2D launch configuration in the main loop
- Draw a picture of the solution to validate it
  - a plotting script is provided for visualizing the results
  - use a small domain for visualization

```
# Build and run after writing code
srun diffusion2d 8 1000000

# Do the plotting
module load daint-gpu
module load jupyterlab
python plotting.py -s
```