Performance analysis on GPUs with NVIDIA tools

Dominik Ernst

NHR FAU

Example: 2D Jacobi

Log in to Alex:

```
  ssh -J <user>@csnhr.nhr.fau.de <user>@alex
```

Allocate an interactive Job on a GPU:

```
  srun --gres=gpu:a100:1 -p a100 --reservation
GPUperf  --time 4:00:00 --pty /bin/bash -l
```

Allocate an interactive Job from different account:

```
  srun --gres=gpu:a100:1 -p a100 -C a100_40 --time
4:00:00 --pty /bin/bash -l
```

Example: 2D Jacobi

Get the Source:

```
  git clone https://github.com/te42kyfo/omp_jacobi
```

or

```
  cp -r ../b53k0000/omp_jacobi
```

Load the compiler module

```
  module load nvhpc
  module load cuda
```

Build and run the CPU base line

```
  make main1
  ./main1
```

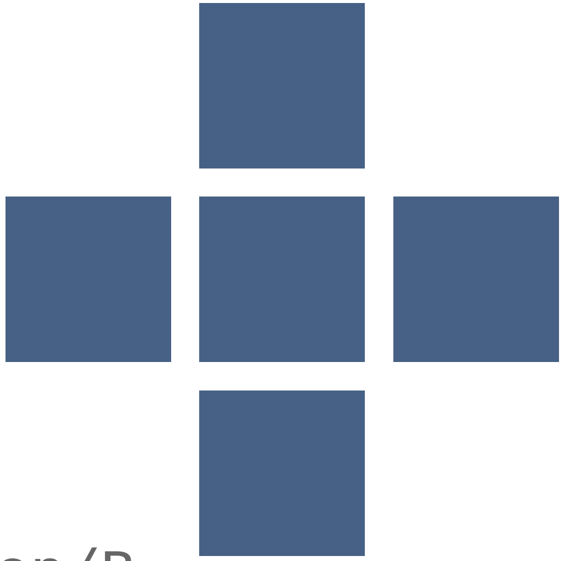
Roofline Analysis: 2D Jacobi

3 ADDs, 1 MUL per iteration:

```
A[o] = 0.25 * (B[^] + B[v] + B[<] + B[>])
```

Read entire grid A and B once each → on average: one value per iteration

```
= 2x8B / iteration
```



Code Intensity:

```
4 Flop / 16 B = 0.25 Flop/B
```

A100 Machine Intensity:

```
9.7 Tflop/s / 1555 GB/s = 6.2 Flop/B
```

Build/Run/Profile

Build and run the Nth version

```
make main<N>
./main<N>
```

Create a profile

```
nsys profile main<N>
```

Launch the profiling GUI

```
nsys-ui
```

Kernel Profiling

Kernel profiling

```
ncu <application>
```

List metric sections

```
ncu --list-sections
```

Collect all sections

```
ncu --set full -f -o <output file> <application>
```

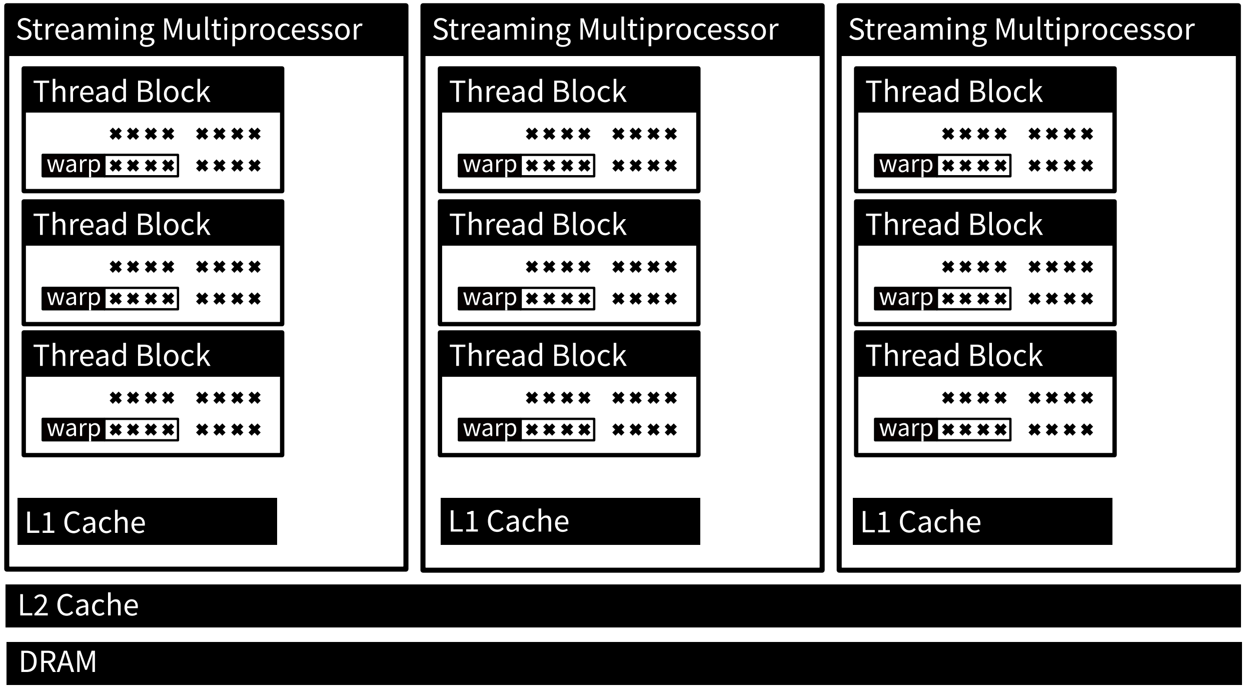
Launch the ncu profiling GUI

```
ncu-ui
```

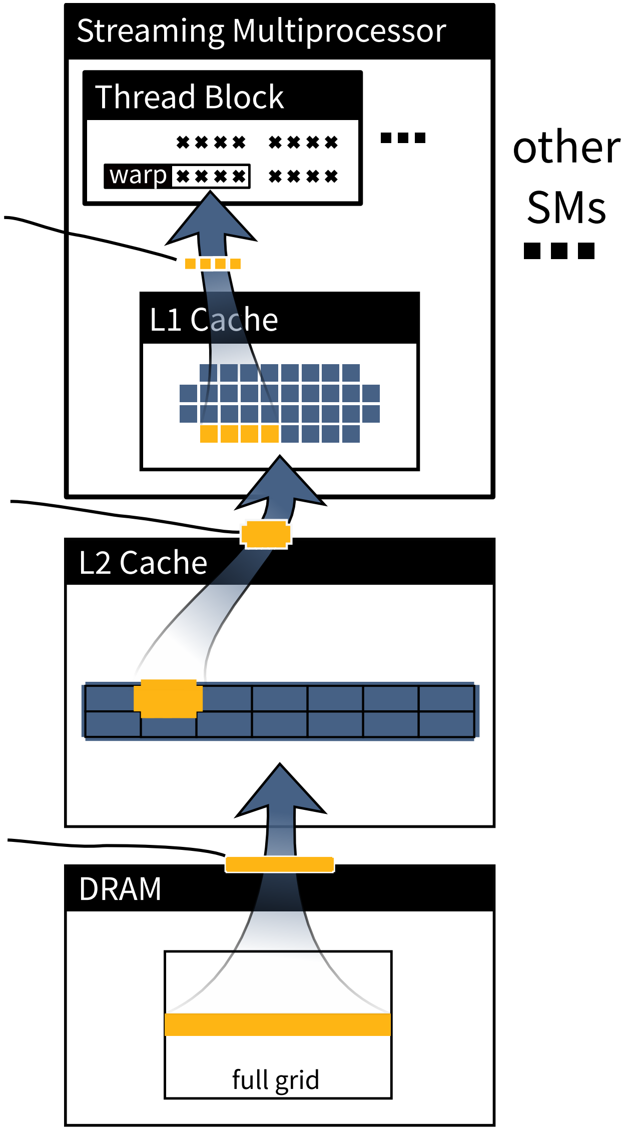
GPU Architecture

32 threads → 1 warp
up to 1024 threads / 32 warps → 1 thread block
up to 64 warps / 2048 threads → 1 SM
108 SM → A100 GPU

2048 threads / SM * 108SM → ~200‘000 threads / GPU



GPU Architecture



per SM: 192 kB L1 cache
shared for all SM: 40MB L2 cache
shared for all SM: 40 GB DRAM

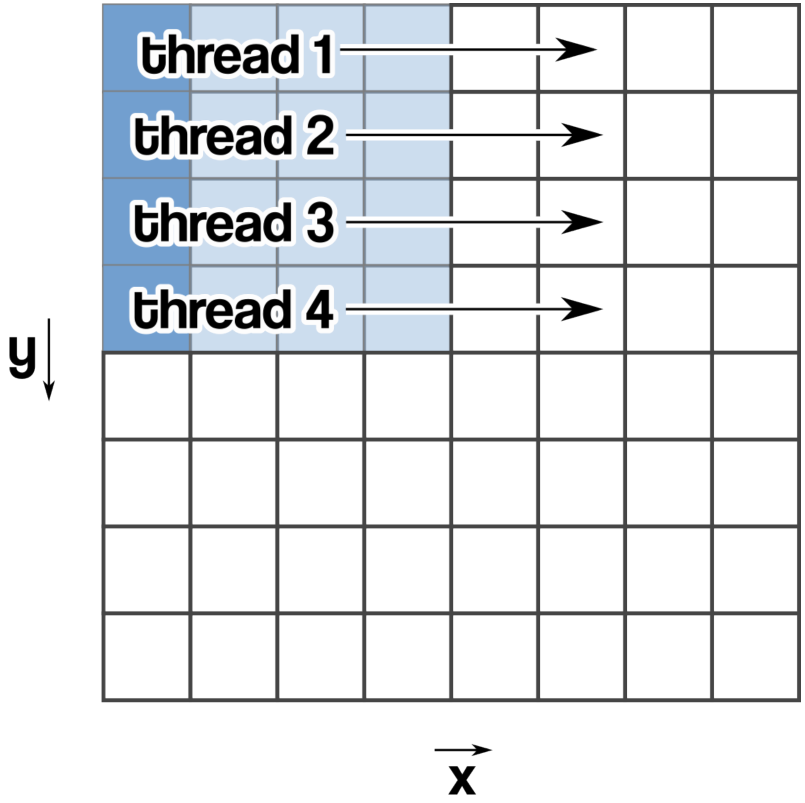
(A100-SXM4-40GB)

OpenMP Loop main4

```
#pragma omp target parallel for
for (int y = 1; y < height - 1; y++)
  for (int x = 1; x < width - 1; x++)
    A[y][x] = ...
```

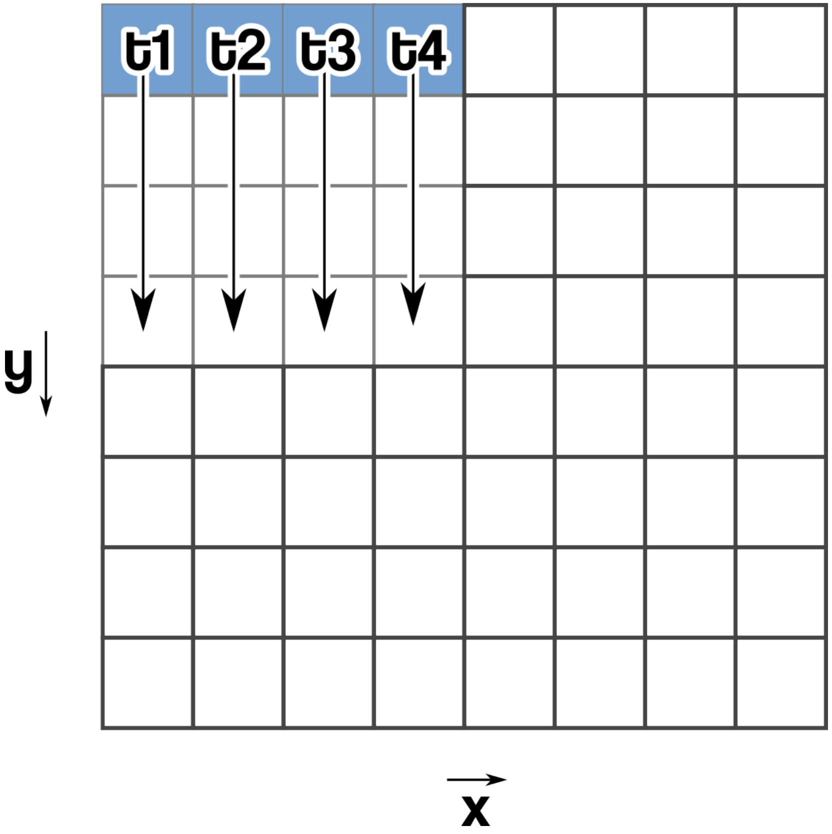
```
kernel:
  y = global_thread_id.x;
  for (int x = 1;
       x < width – 1;
       X++)
    A[y][x] = ...

kernel<<< rows >>>(...)
```



OpenMP Loop main41

```
#pragma omp target parallel for
for (int x = 1; x < width - 1; x++)
  for (int y = 1; y < height - 1; y++)
    A[y][x] = ...
```

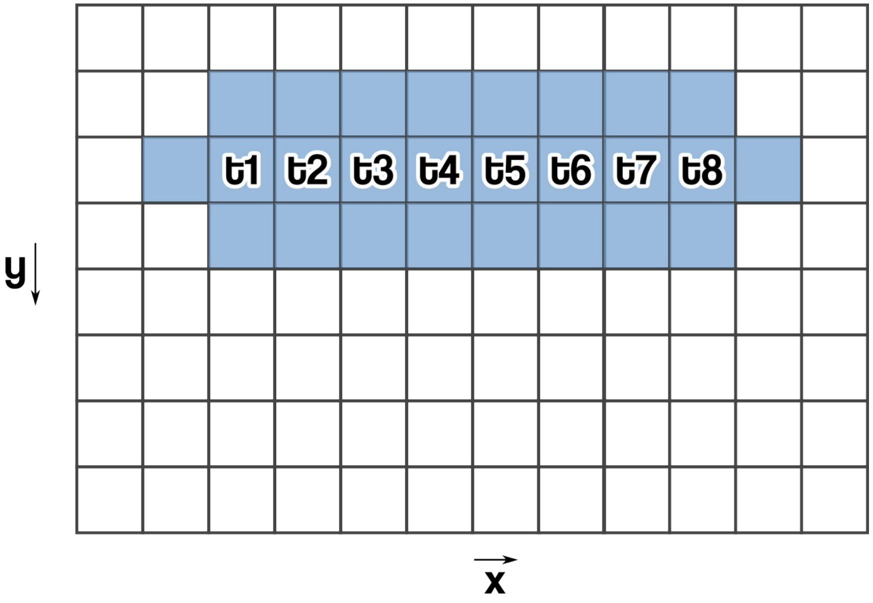


OpenMP main51

```
#pragma omp target parallel for collapse(2)
for (int y = 1; y < height - 1; y++)
  for (int x = 1; x < width - 1; x++)
    A[y][x] = ...
```

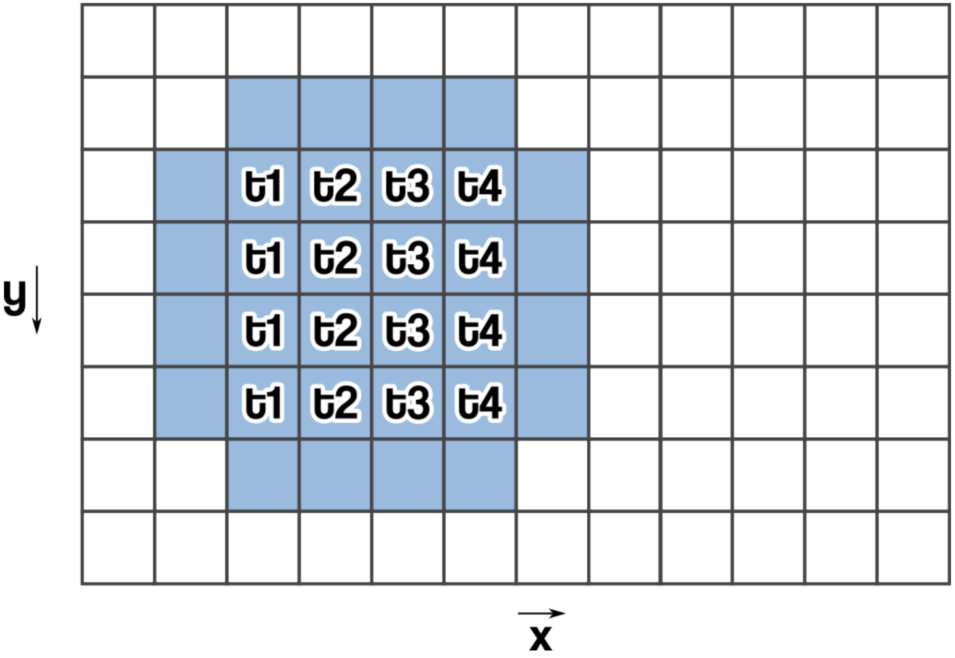
```
kernel:
  y = global_thread_id.x / N;
  X = global_thread_id.x % N;
  A[y][x] = ...

kernel<<< rows*columns >>>(...)
```

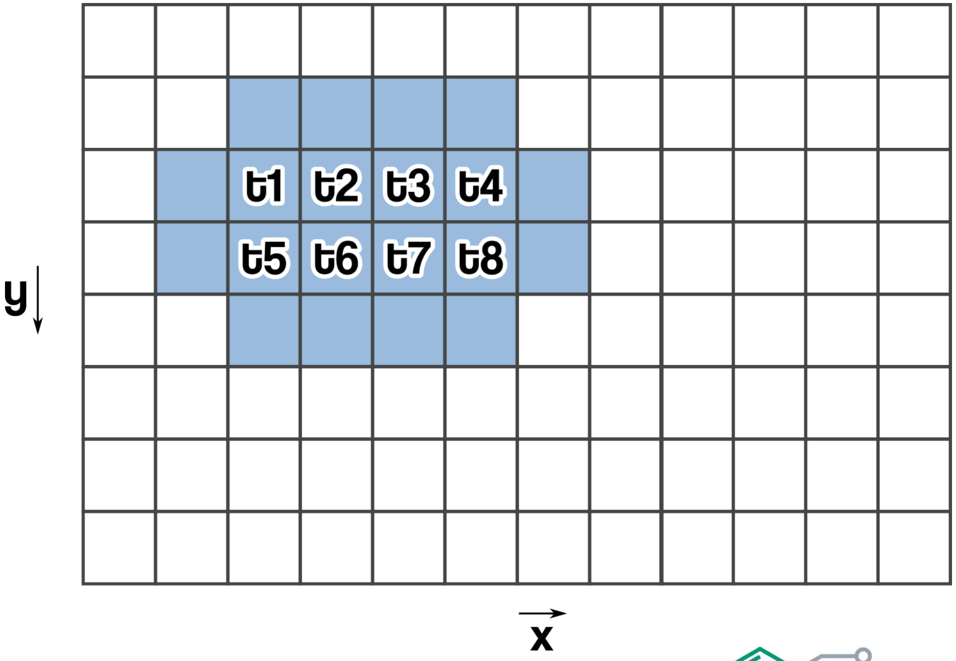


OpenMP Loop main6 / main7

```
#pragma omp target parallel for collapse(2)
for (int oy = 1; oy < height - 1; oy += 4)
  for (int x = 1; x < width - 1; x++)
    for (int iy = 0; iy < 4; iy++) {
      int y = oy + iy;
```



t1
t2
t3
t4
t1
t2
t3
t4
t1
t2
t3
t4
t1
t2
t3
t4
y
x



t1
t2
t3
t4
t5
t6
t7
t8
y
x