1 IMPLEMENTATION OF BOX STENCIL KERNEL

Algorithm 1 Box stencil kernel in *TCstencil*

```
Input: mesh A, para_box1, para_box2, para_box3,para_box4
Output: the updated mesh C'
Shared Memory: sinput, soutput1,soutput2,soutput3,soutput4
Fragment: a_frag,b_frag,c_frag
p=[para_box1, para_box2, para_box3,para_box4]
sout=[soutput1,soutput2,soutput3,output4]
load A' of A to sinput
for i from 0 to 2 do
    wmma::fill_fragment(c_frag, 0.0f)
    wmma::load_matrix_sync(a_frag, p[i], 16)
    wmma::load_matrix_sync(b_frag, sinput, 16)
    wmma::mma_sync(c_frag, a_frag, b_frag, c_frag)
    wmma::store_matrix_sync(sout[i], c_frag, 16, wmma::mem_row_major)
calculate result by soutput1,soutput2,soutput3,soutput4 and store it to global memory
```

2 ROOFLINE ANALYSIS

We adopt the roofline model [2] to analyze the effectiveness of stencil computation by Artemis, *TCstencil*, and *TC-w/o tc*. Table 1 shows the the maximum operational intensity $I_{max} = \pi/\beta$ (FLOP/Byte) in FP16 precision of A100 and V100, where π (TFLOPS) is the maximum attainable performance the β (GB/s) is maximum memory bandwidth. We are only considering the multiplication-and-add for stencil as FLOP and the access to global memory (conservatively without considering cache). For updating the inner points in 16×16 mesh with 9pt stencil, the Arithmetic Intensity (AI, FLOP/Byte) of Artemis/*TC-w/o tc*/*TCstencil* is 0.50/2.53/5.33. As for 25pt stencil, the AI of Artemis/*TC-w/o tc*/*TCstencil* is 0.50/7.03/6.40 All the above implementations are memory bound since their AI are lower than I_{max} in Table 1. In A100, the highest TFLOPS of Artemis/*TC-w/o tc*/*TCstencil* is 0.65/1.06/7.52 for 9pt stencil and 0.88/1.55/10.67 for 25pt stencil. The computation details are in the appendix. In V100, the highest GFLOPS of Artemis/*TC-w/o tc*/*TCstencil* is 0.60/0.89/5.89 for 9pt stencil and 0.94/1.73/10.45 for 25pt stencil.

Table 1: The performance information of A100 and V100 in FP16 precision [1].

Item	A100 FMA	A100 TC	V100 FMA	V100 TC
π	78	312	31.4	125
β	1555		900	
I_{max}	50.1	200.64	34.88	128.8

3 APPENDIX

The AI computation detail:

- Artemis 9pt: 9 FLOP/(9*2) Byte. update every point, read 9 points in half and perfrom 9 FLOP.
- Artemis 25pt: 25 FLOP/(25*2) Byte.
- *TC-w/o tc* 9pt: (12*12*9) FLOP / (16*16*2) Byte. update every 12*12 points, load 16*16 points in global memory, perfrom 12*12*9 FLOP.
- *TC-w/o tc* 25pt: (12*12*25) FLOP / (16*16*2) Byte.
- *TCstencil* 9pt: (16*16*16*2) FLOP/ (3*16*16*2) Byte. update every 12*12 point, load 16*16 points and 2*16*16 elements in parameter matrices. perform twice Tensor Core MMA, each MMA perform 16*16*16 FLOP.
- *TCstencil* 25pt: (16*16*16*4) FLOP/ (5*16*16*2) Byte.

The FLOPS computation detail:

- Artemis 9pt: ((N-r)*(N-r))*9 FLOP/ (time) s. every point in mesh need 9 FLOP to be updated.
- Artemis 25pt: ((N-r)*(N-r))*25 FLOP/ (time) s.
- *TC-w/o tc* 9pt: ((N-r)*(N-r))*9 FLOP/ (time) s.
- *TC-w/o tc* 25pt: ((N-r)*(N-r))*25 FLOP/ (time) s.
- *TCstencil* 9pt: [(N-r)*(N-r)/(12*12)*(16*16*16*2)] FLOP/ (time) s. The needed FLOP is same with that of AI.
- *TCstencil* 25pt: [(N-r)*(N-r)/(12*12)*(16*16*16*4)] FLOP/ (time) s.

REFERENCES

[1] NVIDIA. 2021-1-30. NVIDIA Turing Architecture Whitepaper. online. https://www.nvidia.com/en-us/data-center/a100/ Accessed March 30, 2021.
[2] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM* 52, 4 (2009), 65–76.