Nov 22, 2024

APAI2024 - LAB06

PULP Tiling - part1

Authors: Davide Nadalini, Lorenzo Lamberti, Luka Macan, Alessio Burrello, Francesco Conti
Contacts: d.nadalini@unibo.it, lorenzo.lamberti@unibo.it

Links: ***GitHub Link (code)*** ***GDOC link (assignment)***

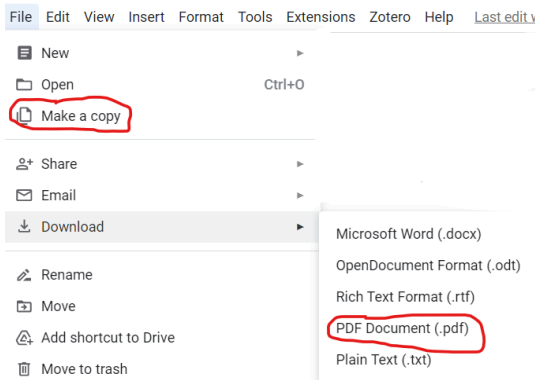
Summary

1. Subject(s):
    - 2D convolution in L1
    - 2D convolution in L2
    - Layer Tiling
2. Programming Language: C
3. Lab duration: 3h
4. Assignment:
    - Time for delivery: 1 week
    - **Submission deadline:** Dec 6, 2024 at 14:30

How to deliver the assignment

You will deliver ONLY THIS TEXT FILE, no code

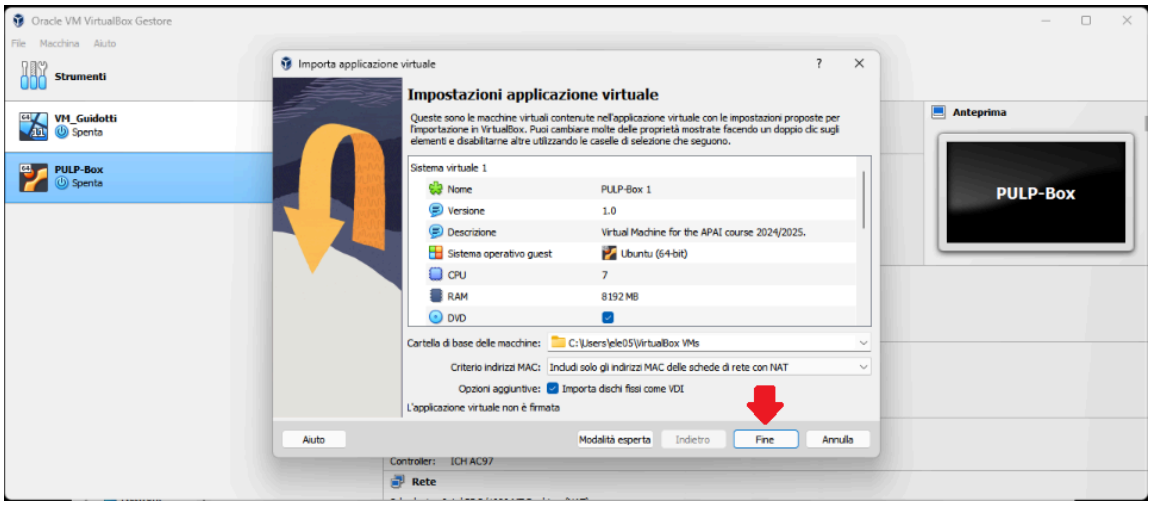
- Copy this google doc to your drive, so that you can modify it. (`File -> make a copy`)
- Fill the tasks on this google doc.
- Export to pdf format.
- Rename the file to: LAB<number_of_the_lesson>_APAI_<your_name>.pdf
- Use Virtuale platform to load ONLY your .pdf file



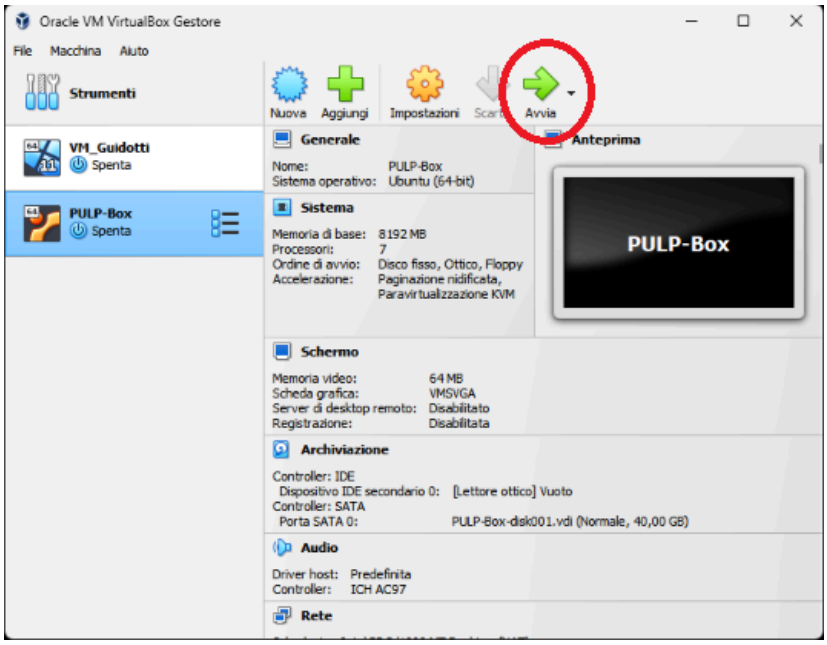
Setup

0. Access to the local VM and setup pulp-sdk

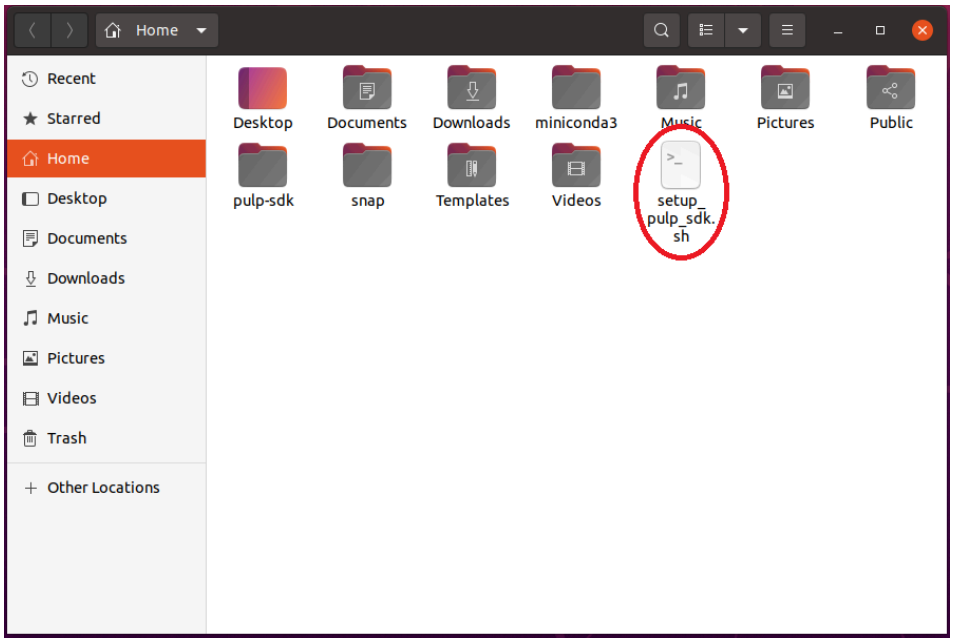
- On the lab's PCs, open the file explorer and go to This PC, C:/VM_Nadalini2
- Double click on PULP-box.ova
- VirtualBox opens, just click on "Fine"



- Wait for the VM to be imported
- Open the VM with "Avvia"



- Password is ‘pulp’
- Open a terminal (right click – open a new terminal)
- **Setup the PULP-SDK:** `source setup_pulp_sdk.sh`



- Clone GitHub repository of today’s lab: `git clone https://github.com/EEESlab/APAI24-LAB06-PULP-Tiling-part1`

- `cd APAI24-LAB06-PULP-Tiling-part1`

Run code

```
$ python parameters_generate.py --channels=1
--spatial_dimension=1
$ make clean all run
```

Remember to change these lines in Src/main.c

```
#define CHANNELS 32
#define SPATIAL_DIM 32
```

LAB STARTS HERE

Case study: Convolutional Layer

Input Size: C x N x N

Output Size: K x N x N

Filter Size: C x C x 1 x 1

Padding: P, Stride: 1

Fixed parameters: F = 1, P = 0.

Setup:

- Open VSCode.
- Go to your exercise folder
- Every time you want to run the code, **SAVE your file** and write in the terminal : `make clean all run`

How to run the code:

1. *Choose the exercise by uncommenting one of the following defines in main.h:*

```
#define EXERCISE1
//#define EXERCISE2
// #define EXERCISE3
```

2. *To generate input.h, weight.h, output.h use the parameters_generate.py script present in the same folder, specifying the number of channels and the spatial dimension as command line parameters.*
   Example: `python3 parameters_generate.py --channels=1 --spatial_dimension=1`
3. *Code execution:* `make clean all run`

Exercise 1: find maximum dimensions of layers fitting L1 without tiling

We tackle a 2D convolution with this size:

- Input = SPATIAL_DIM ➜ defined by you
- Output = SPATIAL_DIM ➜ defined by you
- Kernel = 1x1
- Stride = 1
- Padding = 0

Task 1.1. Implementing missing code:

- Add channels and spatial dimensions. File: `main.c`
- Add L1 vector allocation dimensions. File: `layer_execution.c`
- Add code for performance computation File: `layer_execution.c`

```
typedef enum {
  PI_PERF_CYCLES        = 17, /*!< Total number of cycles (also includes the
    cycles where the core is sleeping). Be careful that this event is using a
    timer shared within the cluster, so resetting, starting or stopping it on
    one core will impact other cores of the same cluster. */
  PI_PERF_ACTIVE_CYCLES = 0,  /*!< Counts the number of cycles the core was
    active (not sleeping). */
  PI_PERF_INSTR         = 1,  /*!< Counts the number of instructions executed.
  */
  PI_PERF_LD_STALL      = 2,  /*!< Number of load data hazards. */
  PI_PERF_JR_STALL      = 3,  /*!< Number of jump register data hazards. */
  PI_PERF_IMISS         = 4,  /*!< Cycles waiting for instruction fetches, i.e.
    number of instructions wasted due to non-ideal caching. */
  PI_PERF_LD            = 5,  /*!< Number of data memory loads executed.
    Misaligned accesses are counted twice. */
  PI_PERF_ST            = 6,  /*!< Number of data memory stores executed.
    Misaligned accesses are counted twice. */
  PI_PERF_JUMP          = 7,  /*!< Number of unconditional jumps (j, jal, jr,
    jalr). */
  PI_PERF_BRANCH        = 8,  /*!< Number of branches. Counts both taken and
    not taken branches. */
  PI_PERF_BTAKEN        = 9, /*!< Number of taken branches. */
  PI_PERF_RVC           = 10, /*!< Number of compressed instructions
    executed. */
  PI_PERF_LD_EXT        = 12, /*!< Number of memory loads to EXT executed.
    Misaligned accesses are counted twice. Every non-TCDM access is considered
    external (cluster only). */
  PI_PERF_ST_EXT        = 13, /*!< Number of memory stores to EXT executed.
    Misaligned accesses are counted twice. Every non-TCDM access is considered
    external (cluster only). */
  PI_PERF_LD_EXT_CYC    = 14, /*!< Cycles used for memory loads to EXT.
  Every non-TCDM access is considered external (cluster only). */
  PI_PERF_ST_EXT_CYC    = 15, /*!< Cycles used for memory stores to EXT.
  Every non-TCDM access is considered external (cluster only). */
  PI_PERF_TCDM_CONT     = 16, /*!< Cycles wasted due to TCDM/log-interconnect
  contention (cluster only). */
} pi_perf_event_e;
```

Task 1.2. Finds the maximum spatial dimension:

Fill the following table by:

- (1) Compute (by hand) the maximum spatial dimensions (N) to allow to store input, output, weights and in L1 (consider 50KB±2KB as Maximum). N.B. Consider only multiple of 8.
- (2) Search in the code the im2col vector size (File: pulp_nn_conv.c). Then fill the table with the tot. Value for each input size dimension.
- (3) Compute (by hand) MACs for each Spatial Dimension found
- Calculate the performance with the performance counters
- Compute the metric MACs/cycle.

Note: to calculate the performance you will have to divide the total number of MAC operations with the measured latency. The formula to calculate the total number of MAC operations is:

$$MACs = Kernel\,Height * Kernel\,Width * Channels_{in} * Height_{out} * Width_{out} * Channels_{out}$$

Channels (C)	Spatial Dim. (N)	(1) Memory Occupation (input+weight+output)	(2) Im2Col size	(3) MAC	Cycles	MAC/cycles
16						
32						
64						
128						

Reply to the following questions

- Why performance (MACs/cycle) improves with more channels?

Error1: *when you* ***overflow the L1 memory*** *available you will get this:*

Error2: *If you forget to generate the network parameters of the right size, you will get a similar error (****wrong checksum****)*

Exercise 2: fetch data from L2

Task 2.2. Testing performance degradation when fetching from L2:

- Test all layers found in the previous exercise.

Dimensions (C, N)	MAC	Cycles	MAC/cycles

16, 40			
32, 24			
64, 16			
128, 8			

- Increase the spatial dimensions to 64 in the first two cases and 32 in the last 2 and measure again the performance

16, 64			
32, 64			
64, 32			
128, 32			

Reply to the following questions

- Why is fetching the data from L2 slower?
- Which is the dimension that most influences the performance, channel or spatial? Why?

Exercise 3: Tiling layer

Task 3.1. Implementing missing code:

- Define tiling parameter
- Complete number of tile iteration

Task 3.2. Find the minimum Tiling factor to fit L1:

- Test the four layers specified in the table.
- Find the minimum tiling factor for which the spatial dimension is divided, to fit the layer in L1 (tiling factor must be a divisor of the spatial dimension (N)).

- Compute the corresponding memory occupation in L1

Dimensions (C, N)	L2 Memory Occupation	L1 Memory Occupation	Tiling Factor	Cycles	MAC/cycles
16, 64					
32, 64					
64, 32					
128, 32					

Reply to the following questions

- How do these results compare with full L1 execution?
- How do these results compare with full L2 execution?

Task 3.3. Find the optimal Tiling factor to maximize performance:

- Test the four layers specified in the table.
- Try different tiling factors. Find the optimal one.

Dimensions (C, N)	L2 Memory Occupation	L1 Memory Occupation	Tiling Factor	Cycles	MAC/cycles
16, 64					
32, 64					
64, 32					
128, 32					

Reply to the following questions

- Have you found any difference between different tiling factors? If so, when?