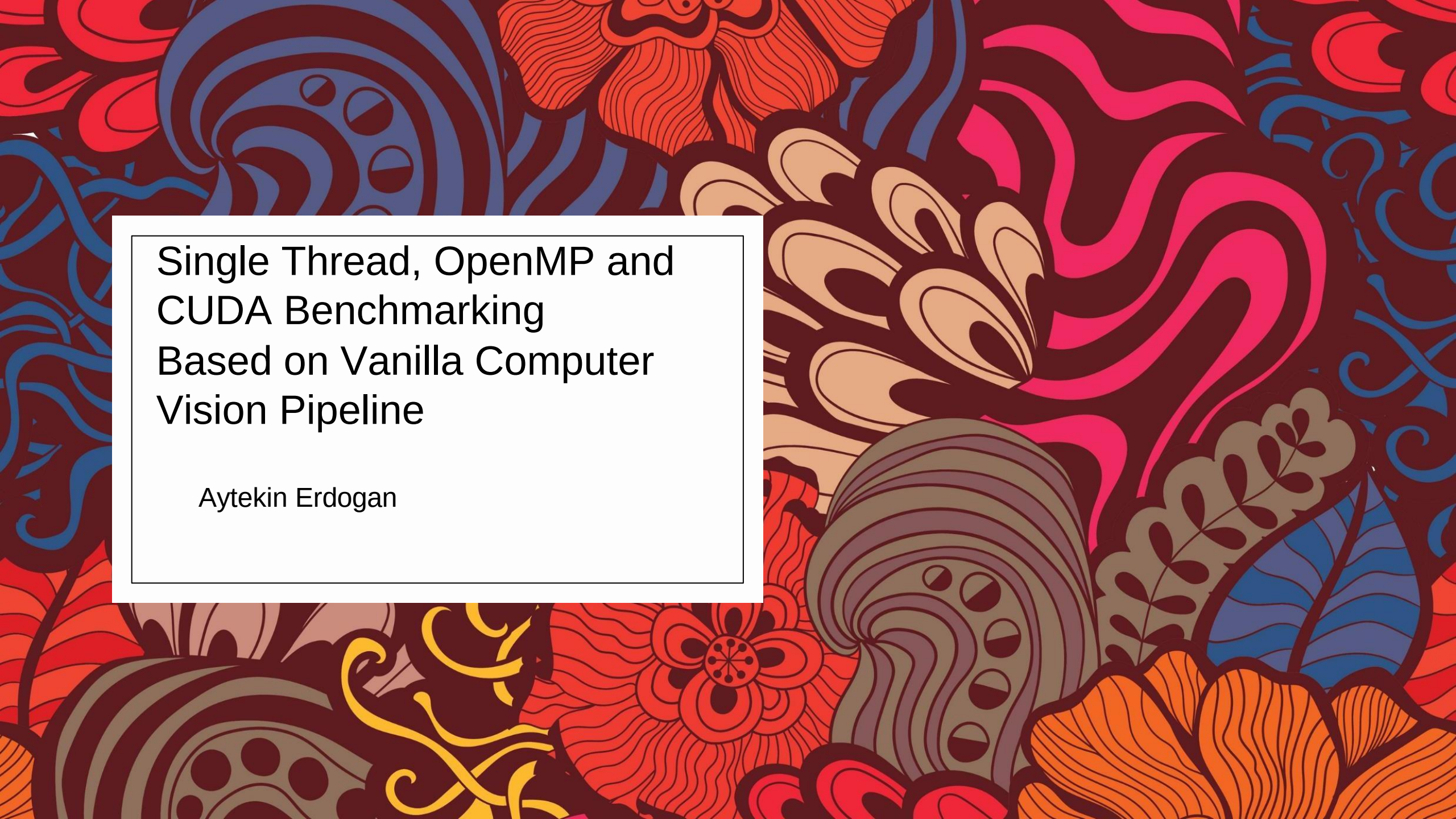


Single Thread, OpenMP and CUDA Benchmarking Based on Vanilla Computer Vision Pipeline

Aytekin Erdogan

Outline

- Introduction
- Problem Definition
- Literature Survey
- Project Results
- References
- ?/+

Introduction

- “Can all the processes be made parallel?“
- “What is the performance improvement of parallel computing in terms of timing?”

We will try to answer these questions on *a vanilla computer vision project*,

by benchmarking different parallel programming libraries and frameworks

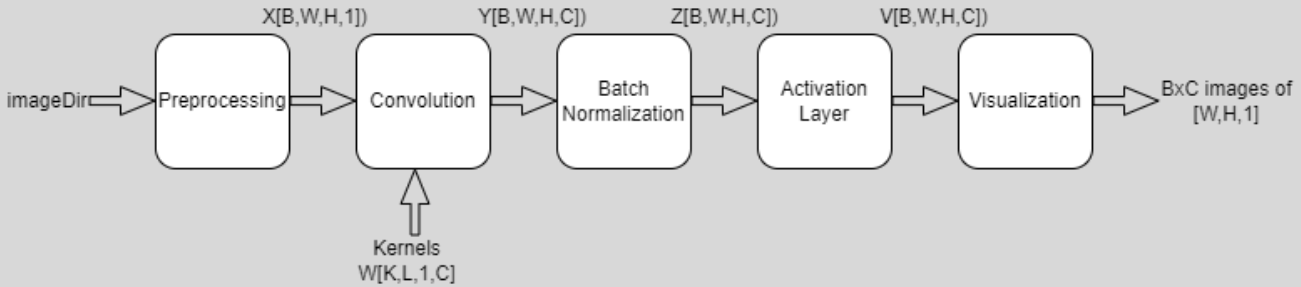
Problem Definition

We A vanilla computer vision pipeline will be implemented; We will create a timing benchmark of vanilla computer vision pipeline from scratch using:

- CUDA,
- openMP/MPI and,
- single thread implementation

Operations are:

- Batch Normalization
- 2D Convolution layers,
- Activation (ReLU) layers,



Problem Definition

Solution Pipeline using operations in the first stage:

1. ***Preprocessing Step***: we will operate on multiple images at the sametime (e.g., 32 images, thus batch size (B) is 32) and resize them into the same resolution (e.g., W=512 and H=512). Output is ***X*** with dimensions [B, W, H, C] .
2. ***Convolution Layer***: This layer takes a batch of images X and a set of kernels W (i.e, [K, L, 1, C]) as input and computes a representation ***Y*** with dimensions [B, W, H, C]. This basically computes a convolution operation.
3. ***Batch Normalization Layer***: This layer takes the output of convolution layer Y and estimates a normalized representation ***Z*** using mean and variance across batch and spatial dimensions.
4. ***Activation Layer***: Similarly, the output of batch normalization layer is fed to this layer. Output dimensions are [B, W, H, C]. This layer is a pixel-wise operation and a simple function ***V***=max(Z,0).
5. ***Visualization***: The output of activation layer must be visualized for each channel and image (i.e., [W, H, 1]).

Literature Survey

A study [1] for a vanilla matrix summation task:

- OpenMP vs Serial CPU: 7.1x faster
- CudaSlow vs Serial CPU: 10.5x faster
- CudaFast vs Serial CPU: 82.8x faster
- CudaSlow vs OpenMP: 1.5x faster
- CudaFast vs OpenMP: 11.7x faster

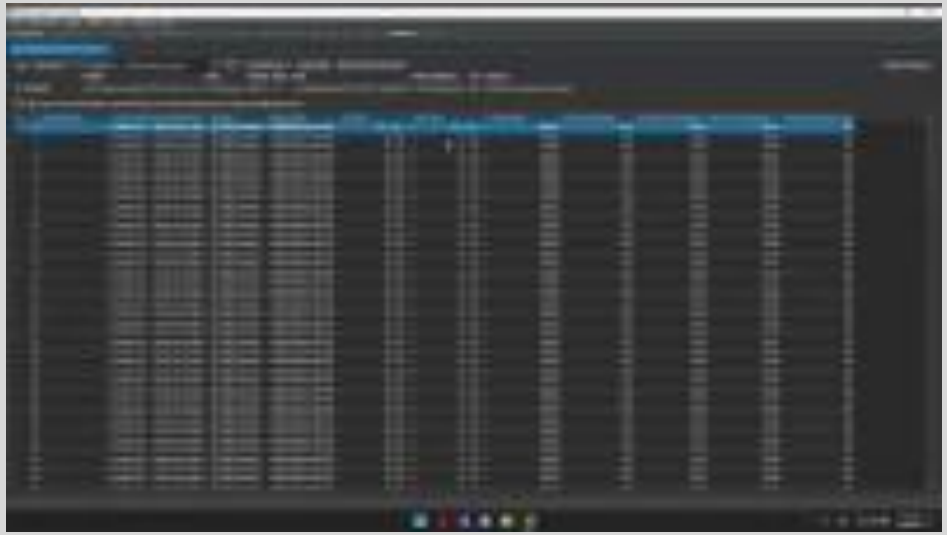
Several studies on GPU accelerated convolution:

- [2] reorganizes the convolution algorithm to prefetch image regions to register
- [3] limits the memory back and forth operations with adaptive tiling operation

Video Demonstration

CPU and OpenMP

CUDA



Results

TABLE I
METHODS' TIMINGS

Computer Vision Modules	HW Occupation Times in ms		
	Single Thread CPU	*OpenMP(12 thread)*	*CUDA*
Preprocess	35.6	20.6	NA
Convolutions	1401.1	299.7	13
Batch Normalization	422.9	131.7	63
ReLU	1054.8	1001.8	1
Overall Timing	2914.4	1453.1	68

TABLE II
METHODS' SPEED-UP

Computer Vision Modules	**Speed-Ups compared to each other**		
	CPU to OpenMP	***CPU to CUDA***	***OpenMP to CUDA***
Preprocess	1.72	NA	NA
Convolutions	4.67	107.77	23.1
Batch Normalization	3.21	6.71	2.1
ReLU	1.1	1054.8	1001.8
Overall Timing	2.0	42.85	21.4

Lessons Learned

• Some of kernels are utilizing greater than 80.0% of the available compute or memory performance of the device. To further improve performance, **work will likely need to be shifted from the most utilized to another unit**.

• The difference between calculated theoretical (100.0%) and measured achieved occupancy (86.8%) can be the result of **warp scheduling overheads or workload imbalances during the kernel execution**. Load imbalances can occur between warps within a block as well as across blocks of the same kernel.

• Some kernel **grids are too small** to fill the available resources on this device, resulting in only 0.1 full waves across all SMs.

• Some kernel's theoretical occupancy (66.7%) is limited by the **required amount of shared memory which is limited by the number of required registers**.

References

1. https://forums.developer.nvidia.com/t/when-to-use-serial-cpu-cuda-openmp-and-mpi/48379
2. https://ieeexplore.ieee.org/abstract/document/6738436?casa_token=uizErUdtOTYAAAAA:PRSlIUQbsyHelFIp7JSqOdnfOZCENbf1ZbbHrU_oNrj-gRDY_uvfQAarJsdLvWo16g4lbmvu48Rt
3. https://www.sciencedirect.com/science/article/pii/S0167739X13001829?casa_token=t5uE8FRZwfsAAAAA:jz1dIAT4E4wrogNdmOP2_XIzHGdK73wrfVb9UETil6Zs9_rxwmYwy0hAH94RUDa8c5XjTOJbqKqP

?/+

- Thank you for your time and contributions.