GPU Programming

曾駿馳 Chun-Chih Tseng : Maintainer in Liger-Kernel

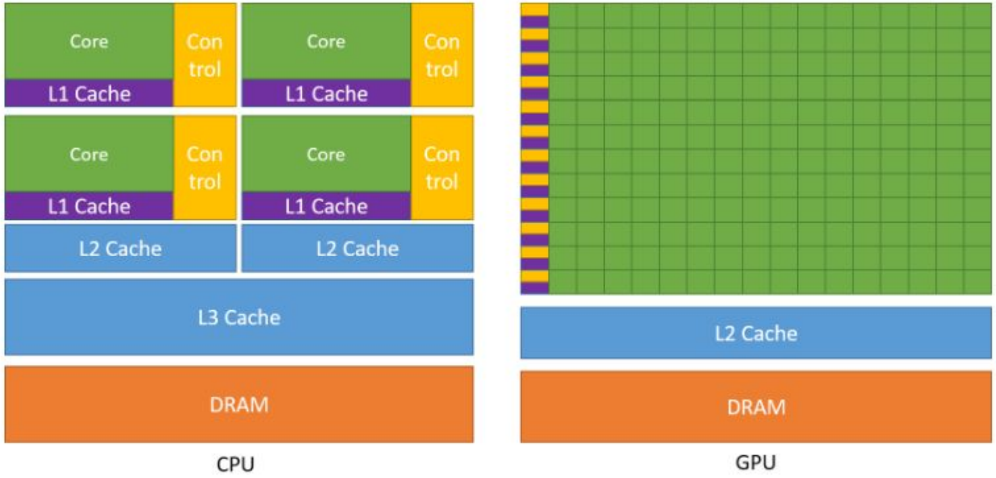
劉立行 Li-Hsing Liu : Maintainer in Liger-Kernel

Outline

- Why GPU Programming
- Why Triton
- Hands-on Labs
  - Vector Add – Triton Basics
  - Grayscale – Indexing
  - Fused Softmax – Kernel Fusion
  - Online Softmax – Better Algorithm

Why GPU Programming – CPU vs GPU

- CPU
  - Sequential
  - Latency oriented
- GPU
  - Parallel
  - Throughput oriented
  - FLOPs

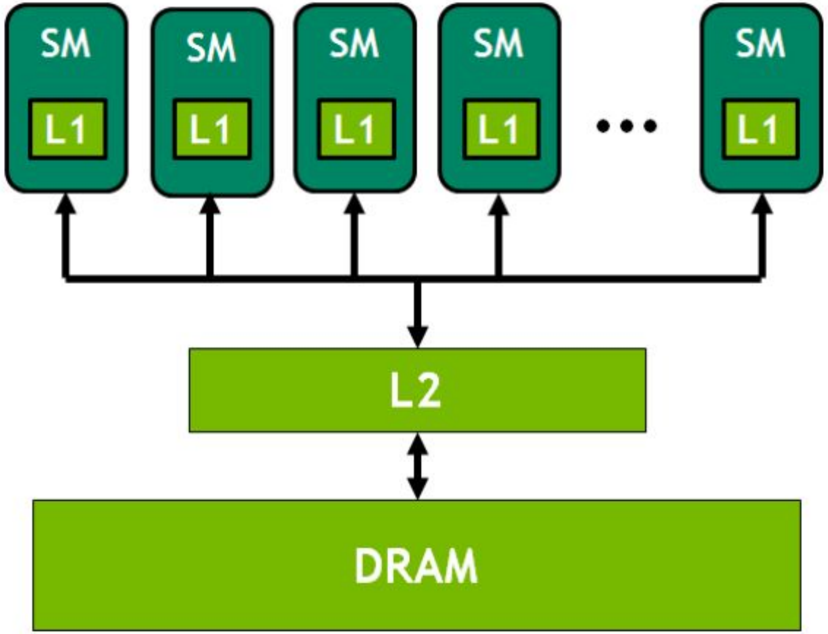


https://docs.nvidia.com/cuda/cuda-c-programming-guide/#the-benefits-of-using-gpus

Why GPU Programming – GPU Architecture

-

Figure 7. GA100 Streaming Multiprocessor (SM)



https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9926-tensor-core-performance-the-ultimate-guide.pdf

Why Triton – Triton vs Cuda

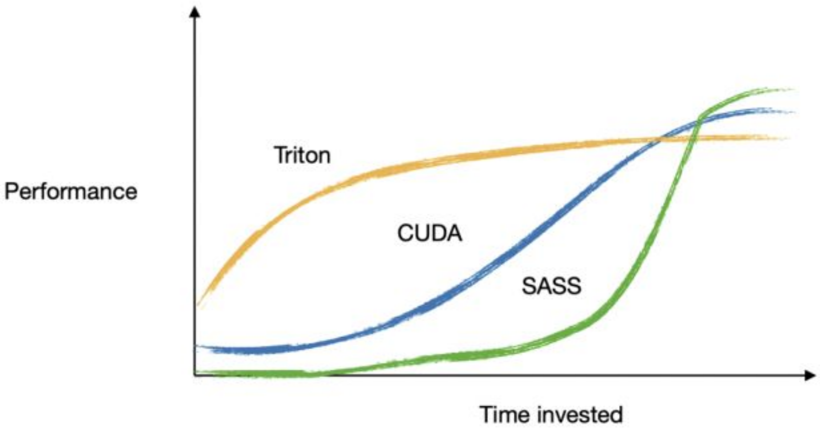
NVIDIA
CUDA

- open-source
- Nvidia, AMD & Intel
- Python
- 90% as fast
- easier

- closed-source
- Nvidia GPUs only
- C/C++
- gold standard for speed
- more difficult

Why Triton

- Automated optimization
- Think in Numpy
- Fast iteration



Triton Basics

2D Indexing

h = 4, w = 7

	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	7	8	9	10	11	12	13
2	14	15	16	17	18	19	20
3	21	22	23	24	25	26	27

offset_0 $\begin{bmatrix} 2 & 3 \end{bmatrix}$ (1d)

offset_1 $\begin{bmatrix} 4 & 5 \end{bmatrix}$ (1d)

offset_0[:,None] $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ (2d)

offset_1[None,:] $\begin{bmatrix} 4 & 5 \end{bmatrix}$ (2d)

w * offset_0[:,None] $\begin{bmatrix} 14 \\ 21 \end{bmatrix}$ (2d)

w * offset_0[:,None] + offset_1[None,:]

$\begin{bmatrix} 18 & 19 \\ 25 & 26 \end{bmatrix}$ (2d)

GPU Performance

GPUs have processing elements (SMs), on-chip memories (e.g. L2 cache), and off-chip DRAM

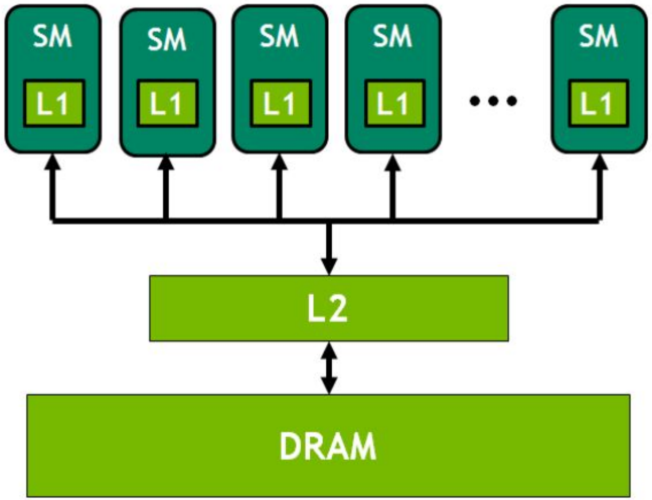
Tesla V100: 125 TFLOPS, 900 GB/s DRAM

What limits the performance of a computation?

$$time_{math\ operations} > time_{data\ movement}$$

$$\frac{FLOPS}{math\ throughput} > \frac{bytes}{memory\ bandwidth}$$

$$\frac{FLOPS}{bytes} > \frac{math\ throughput}{memory\ bandwidth}$$



GPU Performance – Arithmetic intensity

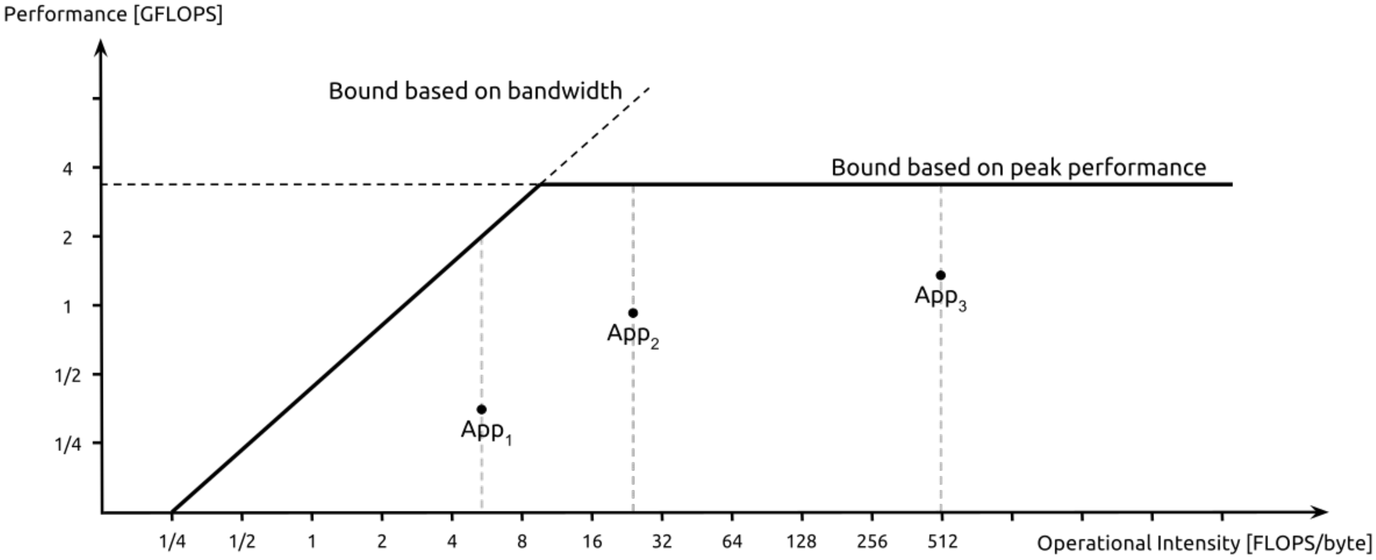
Math limited if: $\frac{FLOPS}{bytes} > \frac{math\ throughput}{memory\ bandwidth}$

Left metric is algorithmic mix of math and memory ops called arithmetic intensity

Right metric is the processor's **ops/byte ratio** - e.g. V100 can execute 125/0.9=139 FLOPS/B

Comparing arithmetic intensity to ops/byte ratio indicates what algorithm is limited by!

GPU Performance – Roofline Model



Online Softmax

Algorithm 2 Safe softmax

$m_0 \leftarrow -\infty$
for $k \leftarrow 1, V$ **do**
    $m_k \leftarrow \max(m_{k-1}, x_k)$
end for
$d_0 \leftarrow 0$
for $j \leftarrow 1, V$ **do**
    $d_j \leftarrow d_{j-1} + e^{x_j - m_V}$
end for
for $i \leftarrow 1, V$ **do**
    $y_i \leftarrow \frac{e^{x_i - m_V}}{d_V}$
end for

Algorithm 3 Safe softmax with online normalizer calculation

$m_0 \leftarrow -\infty$
$d_0 \leftarrow 0$
for $j \leftarrow 1, V$ **do**
    $m_j \leftarrow \max\left(m_{j-1}, x_j\right)$
    $d_j \leftarrow d_{j-1} \times e^{m_{j-1} - m_j} + e^{x_j - m_j}$
end for
for $i \leftarrow 1, V$ **do**
    $y_i \leftarrow \frac{e^{x_i - m_V}}{d_V}$
end for

GPU MODE: https://www.gpumode.com/