

# 推理系统系列

# GPU工作原理



ZOMI



BUILDING A BETTER CONNECTED WORLD

Ascend & MindSpore

[www.hiascend.com](http://www.hiascend.com)  
[www.mindspore.cn](http://www.mindspore.cn)

# Talk Overview

## I. AI 计算体系

- 深度学习计算模式
- 计算体系与矩阵运算

## 2. AI 芯片基础

- 通用处理器 CPU
- 从数据看 CPU 计算
- 通用图形处理器 GPU
- AI 专用处理器 NPU/TPU
- 计算体系架构的黄金10年

### I. 硬件基础

- GPU 工作原理
- GPU AI 编程本质

### 2. 英伟达 GPU 架构

- 从 Fermi 到 Hopper 架构
- Tensor Code 和 NVLink 详解

### 3. GPU 图形处理流水线

- 图形流水线基础
- GPU 逻辑模块划分
- 图形处理算法到硬件

# Talk Overview

## I. 硬件基础

- GPU 工作原理
- GPU AI编程本质

## 2. 英伟达 GPU 架构

- GPU基础概念
- 从 Fermi 到 Volta 架构
- Turing 到 Hopper 架构
- Tensor Code 和 NVLink 详解

## 3. GPU 图形处理

- GPU 逻辑模块划分
- 算法到 GPU 硬件
- GPU 的软件栈
- 图形流水线基础
- 流水线不可编译单元
- 光线跟踪流水线

# Talk Overview

## I. GPU 工作原理

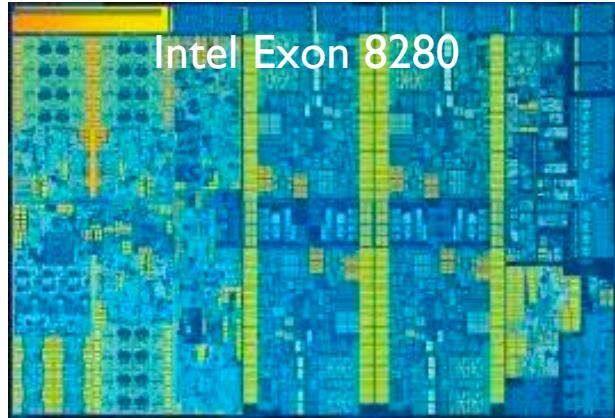
- AX+Y DEMO – AX+Y 例子
- What is inference system – 并发与并行
- Optimization objectives and constraints – GPU 缓存机制
- Difference bet inference system and engine – GPU线程原理

# AX+Y 计算 DEMO

- 2FLOPs : multiply & add
- 2 Memory Loads:  $x[i]$  &  $y[i]$  (per element)
- Single Operation: FMA(fused multiply-add)

```
void demo(double alpha, double *x, double *y)
{
    int n = 2000;
    for(int i = 0; i < n; ++i)
    {
        y[i] = alpha * x[i] + y[i];
    }
}
```

# AX+Y 计算 DEMO



Intel Exon 8280

Memory Bandwidth: **131** GB/sec

Memory latency: **89** ns

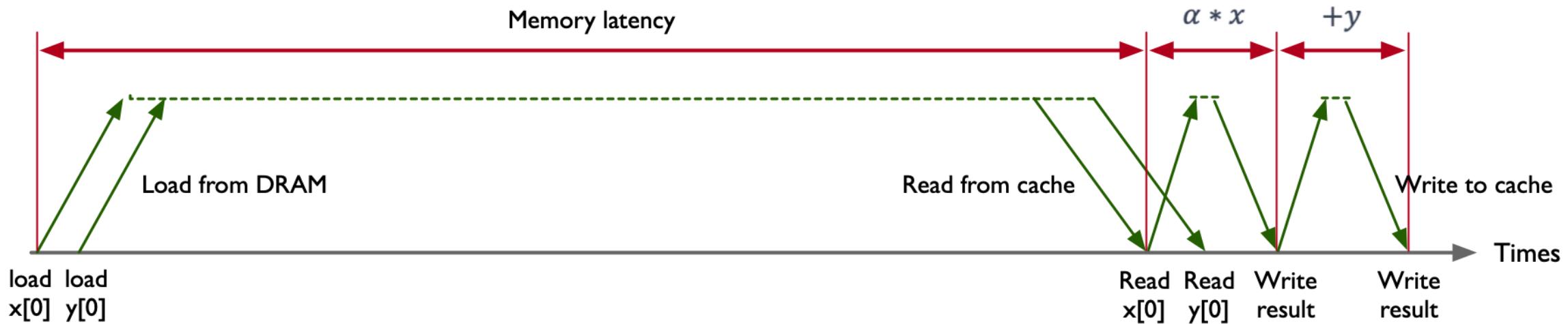


**11,659** bytes can be moved in **89** ns

AXY demo move **16** bytes per **89** ns latency

Memory efficiency = **0.14%**

# AX+Y 计算 DEMO



memory bus is idle 99.86% of the time

# AX+Y 内存利用率

	<b>AMD Rome 7742</b>	<b>Intel Xeon 8280</b>	<b>NVIDIA A100</b>
Memory B/W(GB/sec)	204	131	1555
DRAM Latency(ns)	122	89	404
Peak bytes per latency	24,888	11,659	628,220
Memory Efficiency	0.064%	0.14%	0.0025%

# Z=AX+Y 通过并发进行循环展开

```
1
2 void fun_axy(int n, double alpha, double *x, double *y)
3 {
4     for(int i = 0; i < n; i += 8)
5     {
6         y[i+0] = alpha * x[i+0] + y[i+0];
7         y[i+1] = alpha * x[i+1] + y[i+1];
8         y[i+2] = alpha * x[i+2] + y[i+2];
9         y[i+3] = alpha * x[i+3] + y[i+3];
10        y[i+4] = alpha * x[i+4] + y[i+4];
11        y[i+5] = alpha * x[i+5] + y[i+5];
12        y[i+6] = alpha * x[i+6] + y[i+6];
13        y[i+7] = alpha * x[i+7] + y[i+7];
14    }
15 }
```

Keep Memory bus busy, run  $11,659/16=729$  iterations at once.

J2 }  
J3       $\lambda[j+1] = \alpha * x[j+1] + \lambda[j+1];$   
       $\lambda[j+2] = \alpha * x[j+2] + \lambda[j+2];$

# Z=AX+Y 通过并发进行循环展开

```
1
2 void fun_axy(int n, double alpha, double *x, double *y)
3 {
4     for(int i = 0; i < n; i += 8)
5     {
6         y[i+0] = alpha * x[i+0] + y[i+0];
7         y[i+1] = alpha * x[i+1] + y[i+1];
8         y[i+2] = alpha * x[i+2] + y[i+2];
9         y[i+3] = alpha * x[i+3] + y[i+3];
10        y[i+4] = alpha * x[i+4] + y[i+4];
11        y[i+5] = alpha * x[i+5] + y[i+5];
12        y[i+6] = alpha * x[i+6] + y[i+6];
13        y[i+7] = alpha * x[i+7] + y[i+7];
14    }
15 }
```

- 编译器很少对循环展开上100+
- 一个线程一次执行上千条指令
- 一个线程很难直接处理700个计算负载

Keep Memory bus busy, run  $11,659 / 16 = 729$  iterations at once.

# Z=AX+Y 通过并行进行循环展开

```
1
2     void fun_axy(int n, double alpha, double *x, double *y)
3     {
4         Parallel for(int i = 0; i < n; i++)
5         {
6             y[i] = alpha * x[i] + y[i];
7         }
8     }
9
10 }
```

Keep Memory bus busy, run  $11,659 / 16 = 729$  iterations at once.

# Z=AX+Y 通过并行进行循环展开

```
1 void fun_axy(int n, double alpha, double *x, double *y)
2 {
3     Parallel for(int i = 0; i < n; i++)
4     {
5         y[i] = alpha * x[i] + y[i];
6     }
7 }
8 }
```

- 每个线程独立负责相关计算
- 一共需要 729 个线程
- 程序会受到线程数和内存请求 Bound

Keep Memory bus busy, run  $11,659 / 16 = 729$  iterations at once.

# 并行 Parallelism

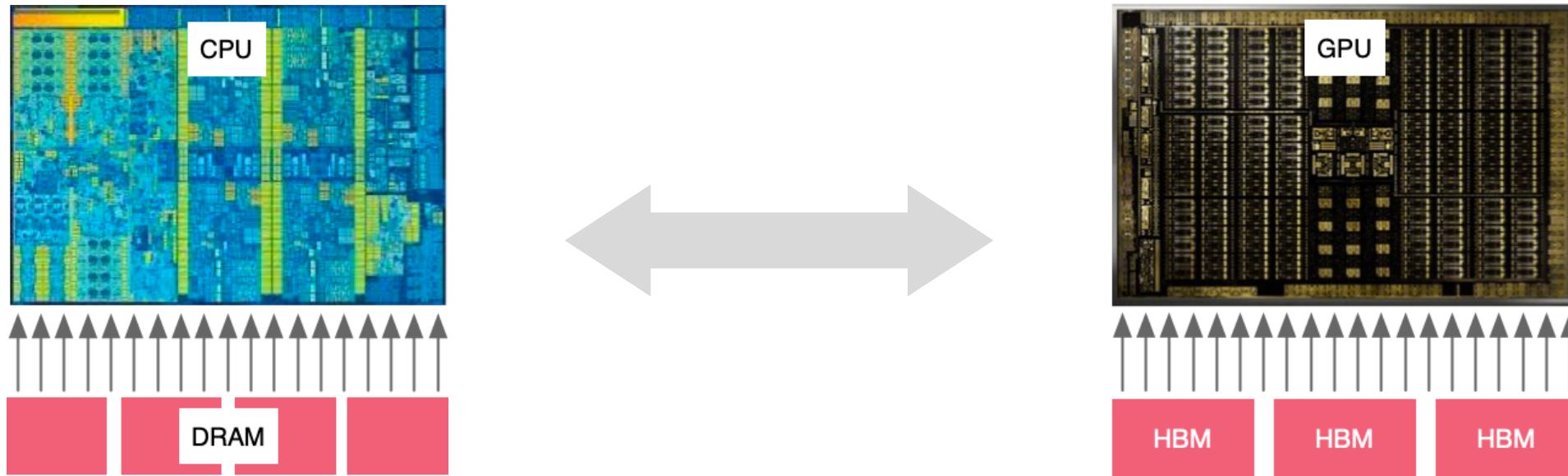
# 并发 Concurrency



# AX+Y 硬件架构线程区别

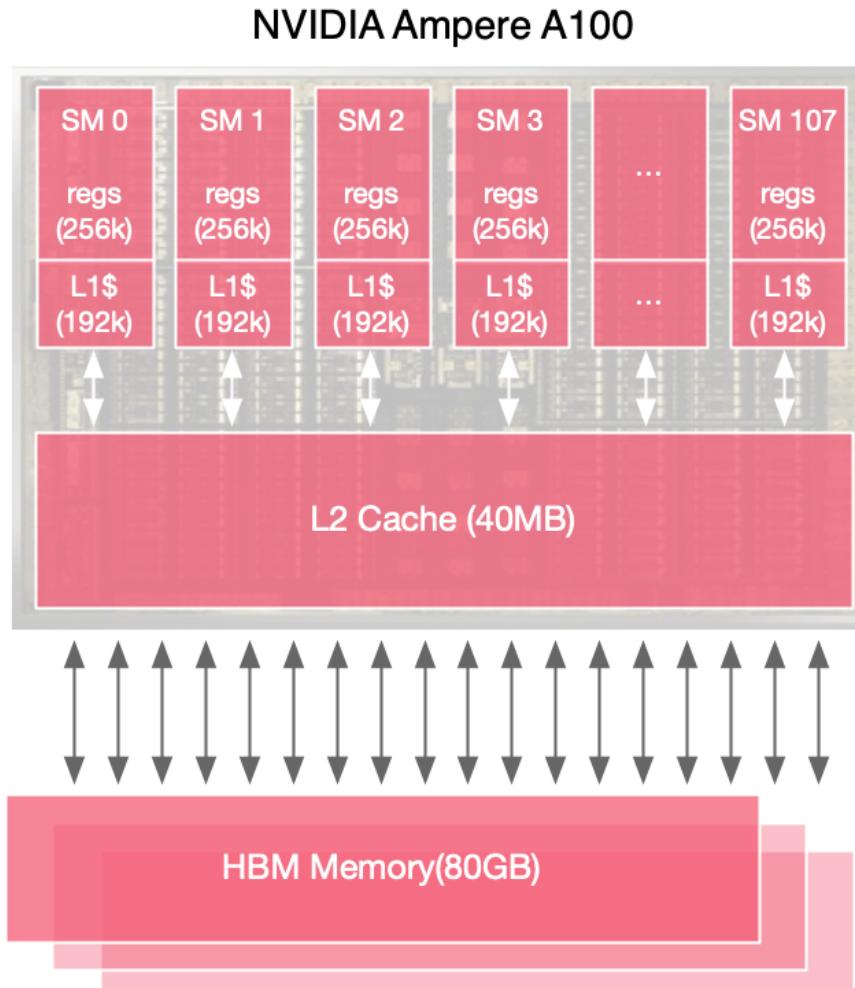
	<b>AMD Rome 7742</b>	<b>Intel Xeon 8280</b>	<b>NVIDIA A100</b>
Memory B/W(GB/sec)	204	143	1555
DRAM Latency(ns)	122	89	404
Peak bytes per latency	24,888	12,727	628,220
Memory Efficiency	0.064%	0.13%	0.0025%
Threads required	1,556	729	39,264
Threads available	2048	896	221,184
Thread Ration	1.3X	1.2X	5.6X

# 架构线程工作原理



	AMD Rome 7742	Intel Xeon 8280	NVIDIA A100
Threads required	1,556	729	39,264
Threads available	2048	896	221,184
Thread Ration	1.3X	1.2X	5.6X

# GPU 缓存机制



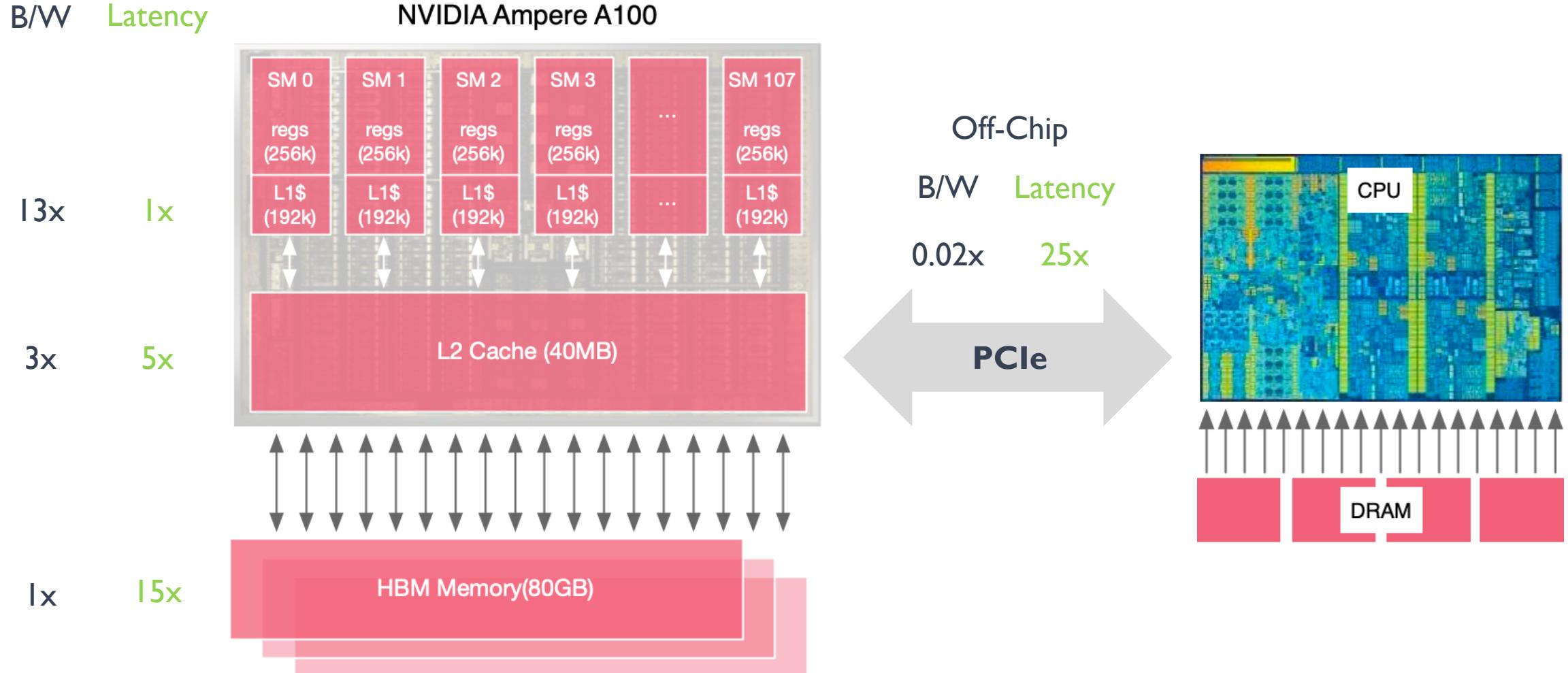
- 108 Streaming Multiprocessors(SMs)

- 256kB Register File per SM(27MB Total)
- 192kB L1 Cache & Shared Mem Per SM(20MB Total)
- 40MB L2 Cache Shared across all SMs

**Caches 缓存**

- 80GB High Bandwidth Memory

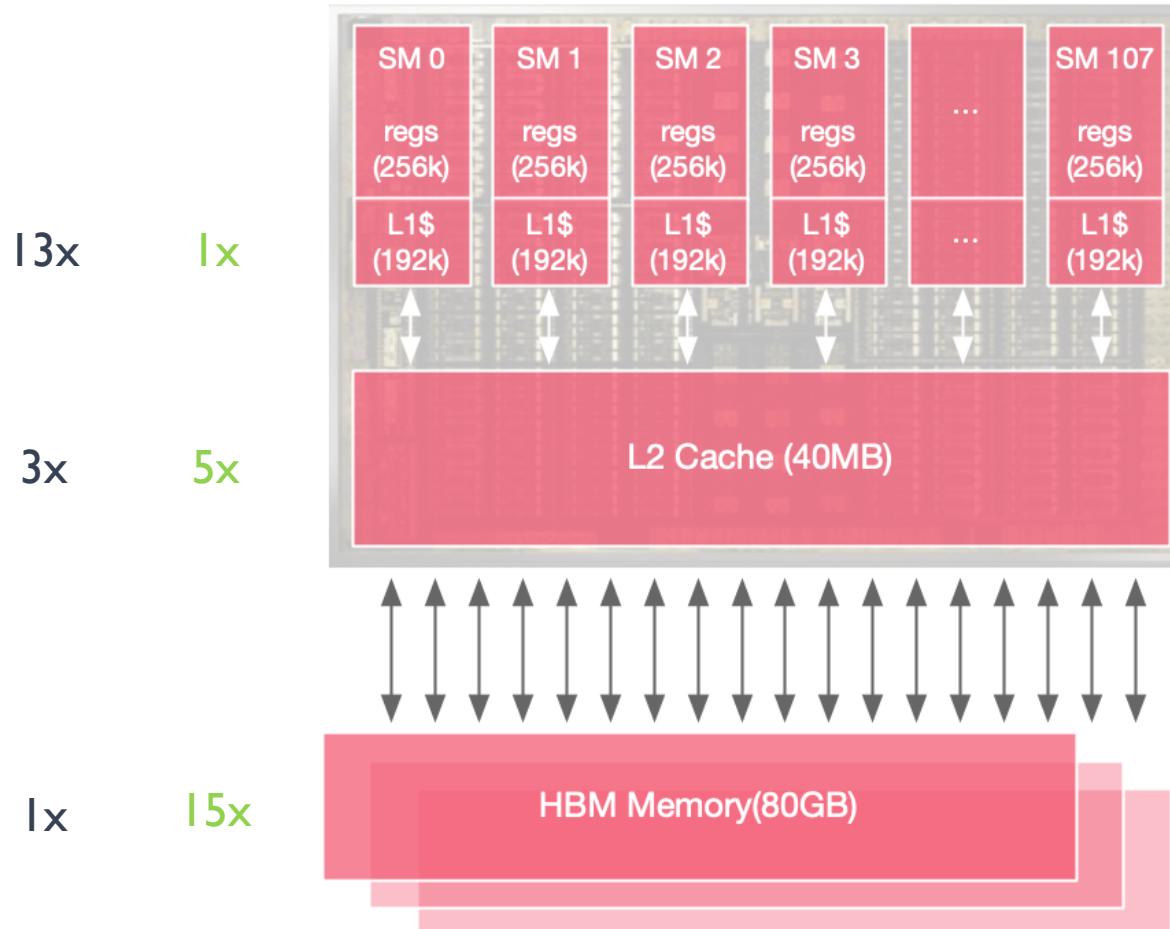
# GPU 缓存机制



# GPU 缓存机制

B/W Latency

NVIDIA Ampere A100

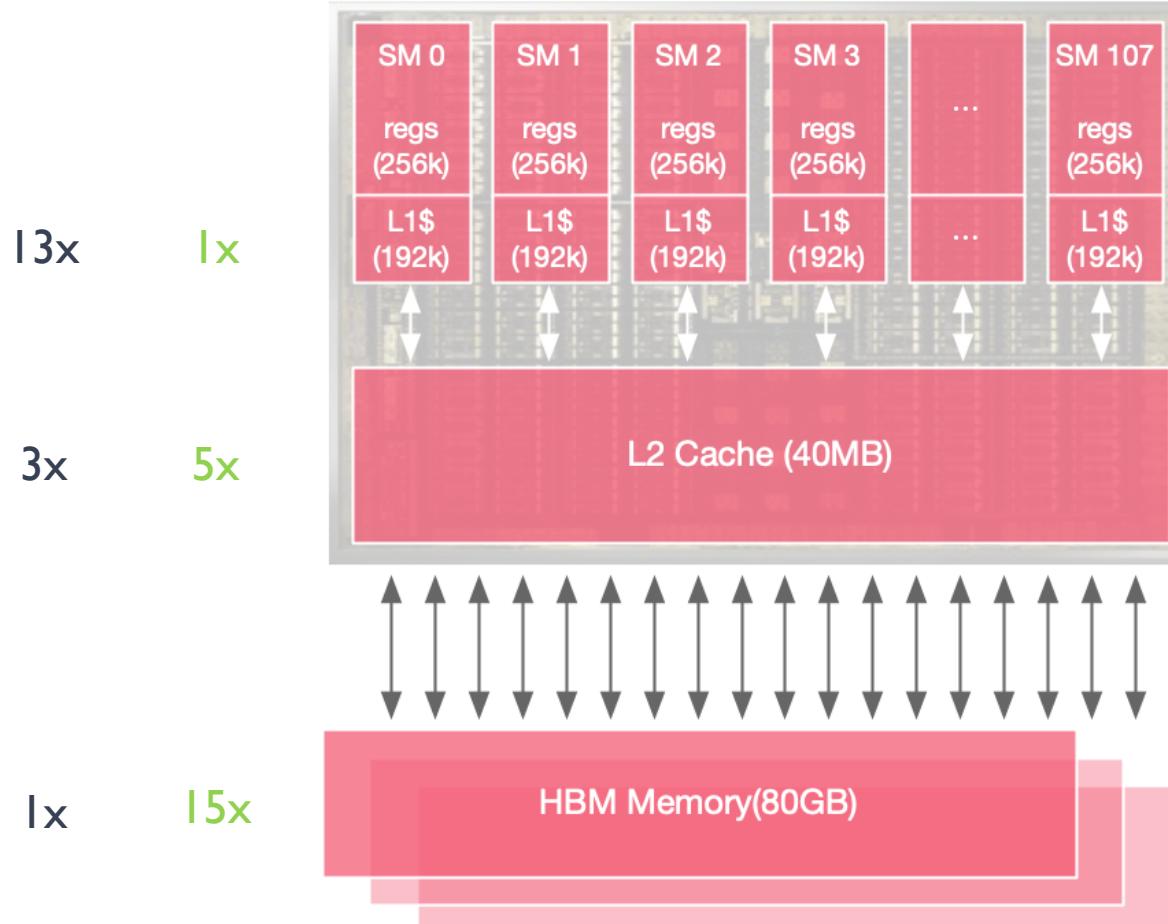


Data Location	Bandwidth (GB/sec)	Compute Intensity
L1 Cache	19,400	8
L2 Cache	4,000	39
HBM	1,555	100
NVLink	300	520
PCIe	25	6240

# GPU 缓存机制

B/W Latency

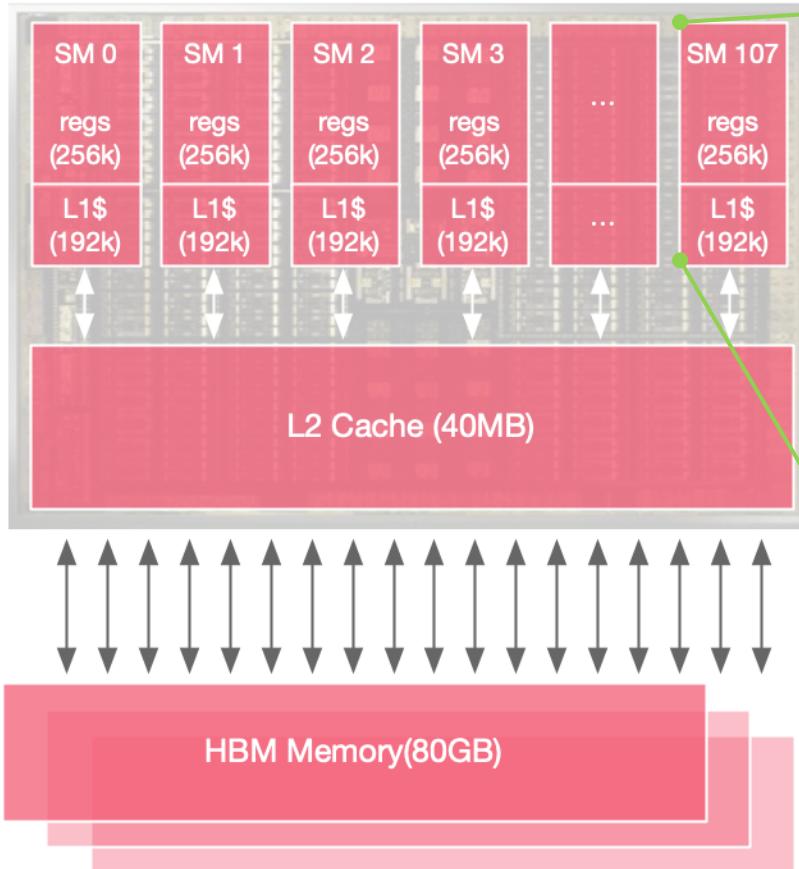
NVIDIA Ampere A100



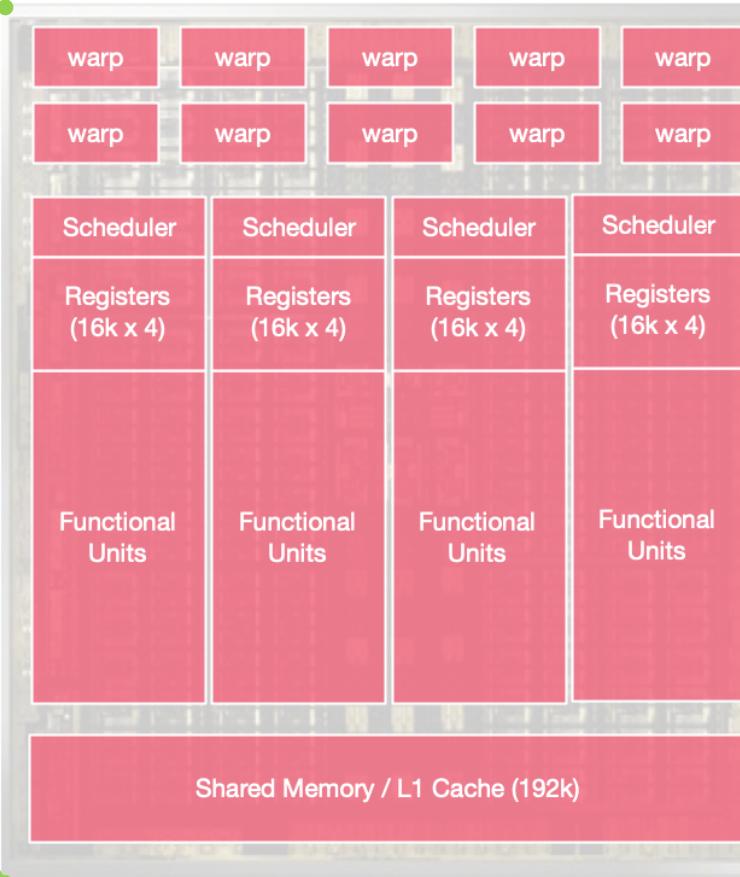
Data Location	Latency (ns)	Threads Required
L1 Cache	27	32,738
L2 Cache	150	37,500
HBM	404	39,264
NVLink	700	13,125
PCIe	1470	2297

# GPU 线程机制

NVIDIA Ampere A100



A100 Streaming Multiprocessor (SM)



64 warps/SM

4x concurrent warp exec

64k 4-byte registers

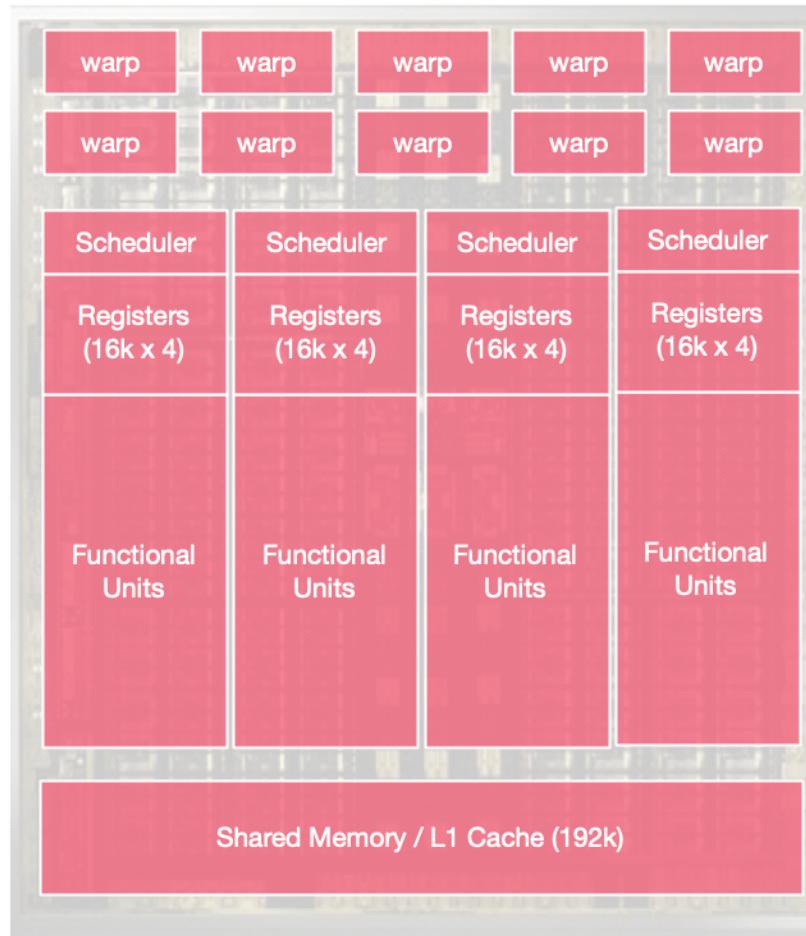
192KB L1/shared memory  
(configurable split)

GPU runs threads in groups of **32** – each group is known as a **warp**

# GPU SMs 线程超配

	Pre SM	A100
Total Threads	2048	221,184
Total Warps	64	6,912
Active Warps	4	432
Waiting Warps	60	6,480
Active Threads	128	13,824
Waiting Threads	1,920	207,360

A100 Streaming Multiprocessor (SM)



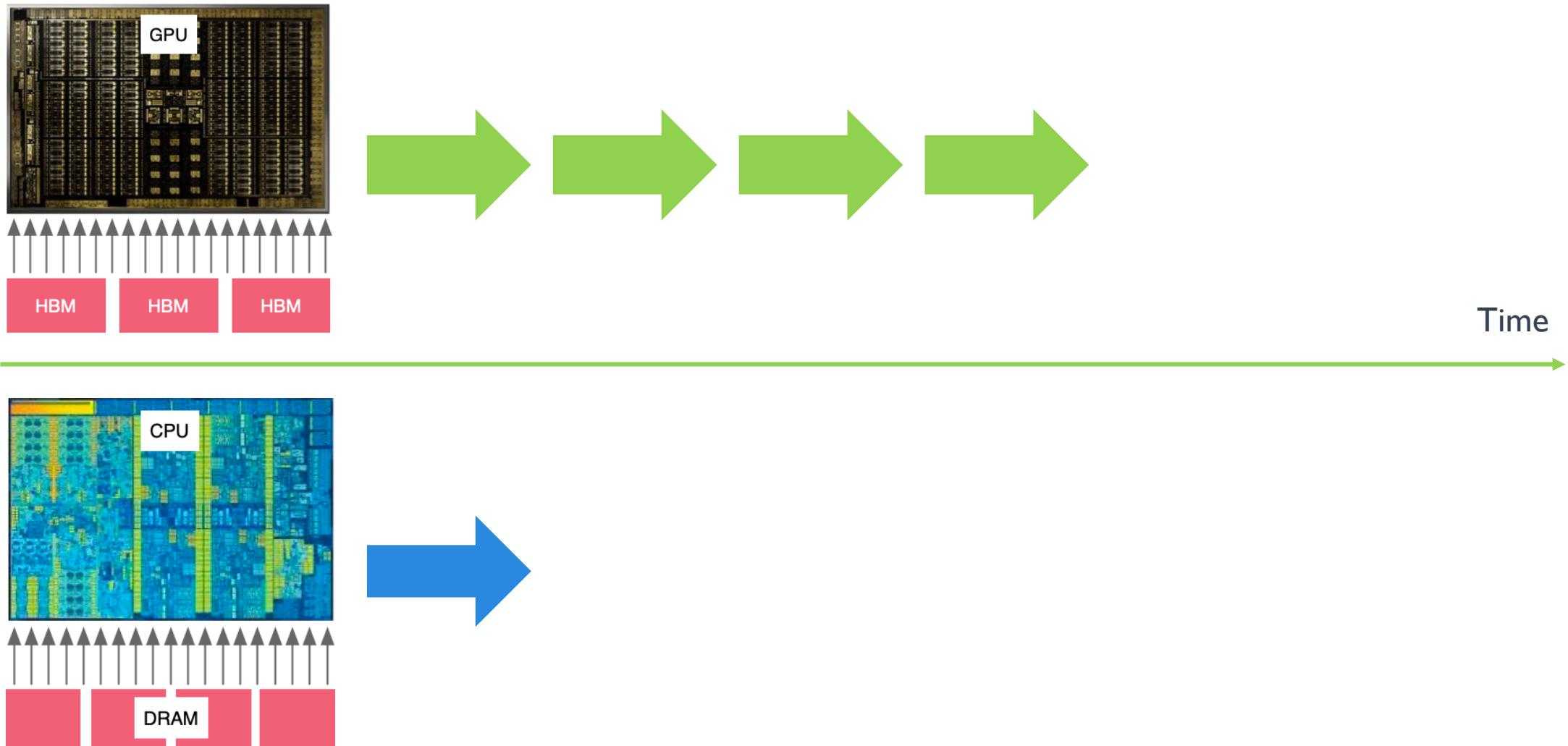
64 warps/SM

4x concurrent warp exec

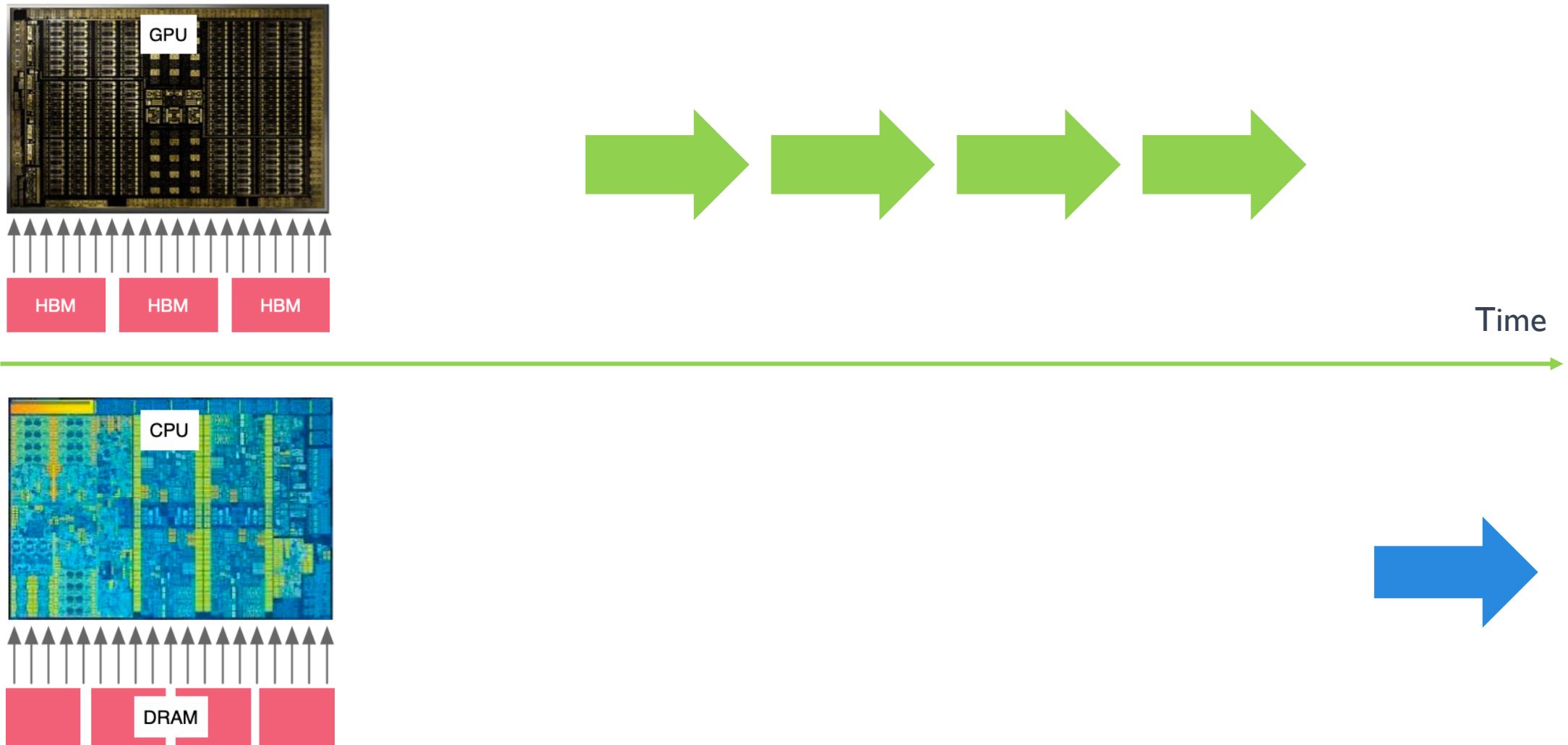
64k 4-byte registers

192KB L1/shared memory  
(configurable split)

# CPU/GPU 并行才是本质



# CPU/GPU 并行才是本质



# Reference

- <https://www.techtarget.com/searchvirtualdesktop/definition/GPU-graphics-processing-unit>
- <https://www.techtarget.com/searchvirtualdesktop/definition/GPU-graphics-processing-unit#:~:text=GPUs%20work%20by%20using%20a,%2C%20high%2Dquality%20graphics%20rendering.>
- <https://computer.howstuffworks.com/graphics-card.htm>
- [https://www.cs.cmu.edu/afs/cs/academic/class/15462-f11/www/lec\\_slides/lec19.pdf](https://www.cs.cmu.edu/afs/cs/academic/class/15462-f11/www/lec_slides/lec19.pdf)
- <https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html>
- <https://www.investopedia.com/terms/g/graphics-processing-unit-gpu.asp>
- [https://www.youtube.com/watch?v=0\\_TN845dxUU](https://www.youtube.com/watch?v=0_TN845dxUU)
- <https://www.youtube.com/watch?v=nEMzlwzmJT8>
- <https://www.heavy.ai/technical-glossary/cpu-vs-gpu>
- <https://www.youtube.com/watch?v=3I10o0DYJXg>
- [https://www.youtube.com/watch?v=658n\\_Ym8dkk](https://www.youtube.com/watch?v=658n_Ym8dkk)
- <https://www.youtube.com/watch?v=5BiAlaFGCoE>
- <https://www.youtube.com/watch?v=bZdxcHEM-uc>



BUILDING A BETTER CONNECTED WORLD

THANK YOU

Copyright©2014 Huawei Technologies Co., Ltd. All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.