# 研一 工作汇报

郭天宇

## MAIN LINE：

- GPGPU-Sim and Accel-Sim
- SMILE： LLC-based Shared Memory Expansion to Improve GPU Thread Level Parallelism (ICCD’23)

## BRANCH LINE：

- Teaching Assistant of “Computer Architecture” (SYSU-ARCH LAB)
- ConvNN (A simple CNN training and inferencing framework)
- PTX-EMU (A simple emulator for CUDA program)
- GEMM-MMA (Optimize GEMM by tensor core step by step)

# SIMULATOR

为什么选择GPU模拟器这个方向?
- CUDA
- GTX1060
- 毕业设计

遇到的一些问题?
- 上手门槛
- 遇事不决**读源码**

```
-gpgpu_cache:dl1 S:4:128:256,L:T:m:L:L,A:384:48,16:0,32
```

模拟器能给我带来什么?
- 体系结构的知识

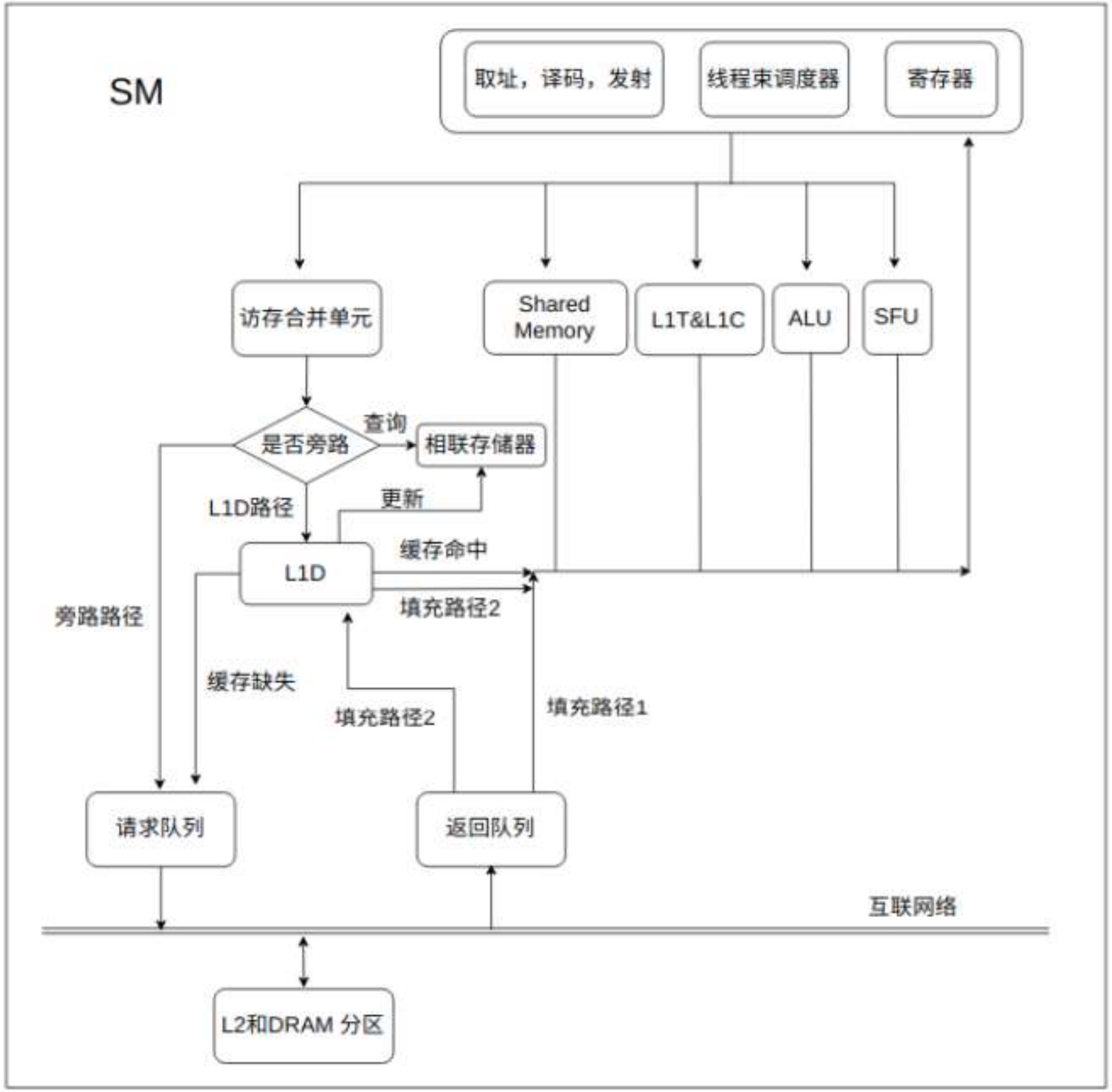



图 3.9 缓存旁路示意图

# SMILE——Motivation

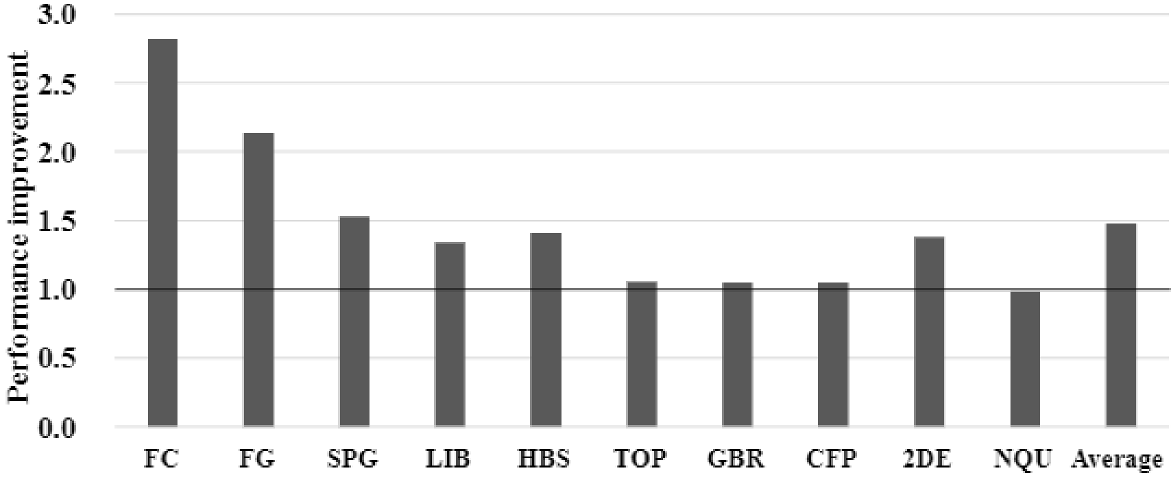



Fig. 3: Performance change after doubling SMEM capacity.

TABLE I: SM occupancy and SMEM usage of applications

| Applications | Occupancy(%) | SMEM(KB) |
|---|---|---|
| SPG | 2.08 | 78 |
| FG | 2.08 | 69 |
| FC | 2.08 | 69 |
| GBR | 6.18 | 32 |
| CFP | 6.24 | 32 |
| TOP | 6.24 | 32 |
| LIB | 9.77 | 20 |
| 2DE | 24.81 | 33.1 |
| NQU | 33.30 | 48 |
| HBS | 46.31 | 12 |

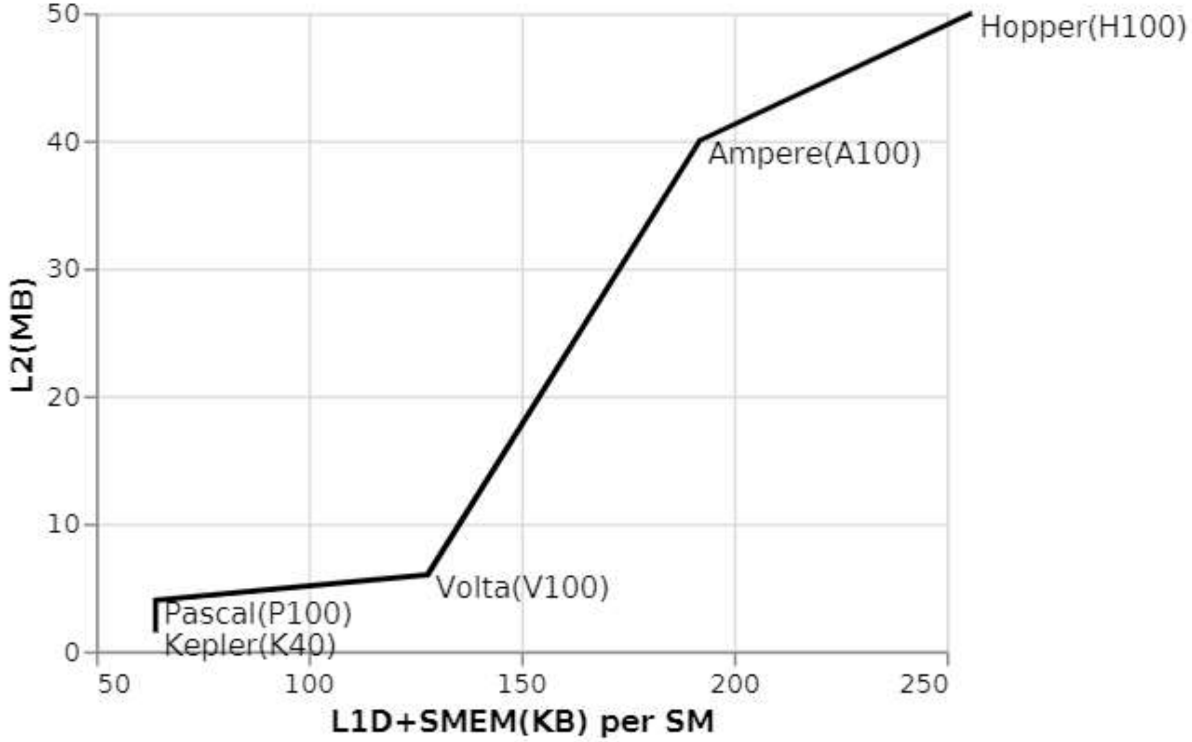



Fig. 1: Nvidia GPU evolution in terms of L1 (x-axis) and L2 (y-axis) capacities.

# SMILE——Design

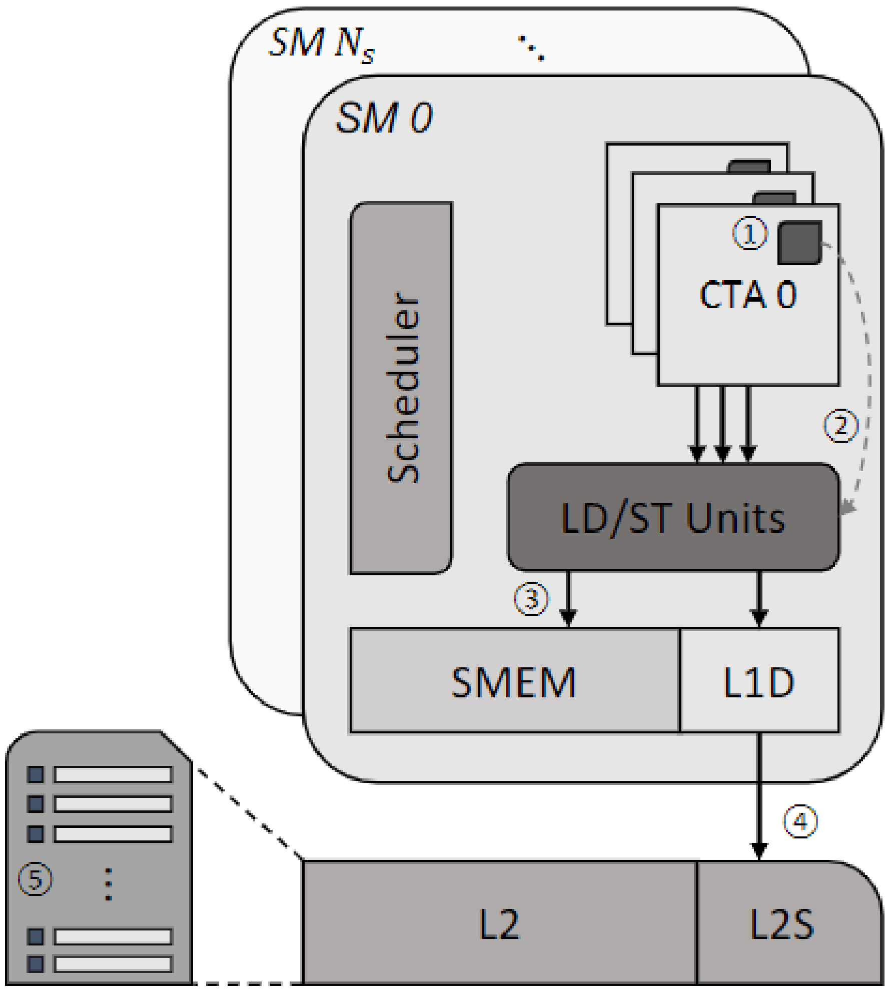



Fig. 5: SMILE overall architecture.

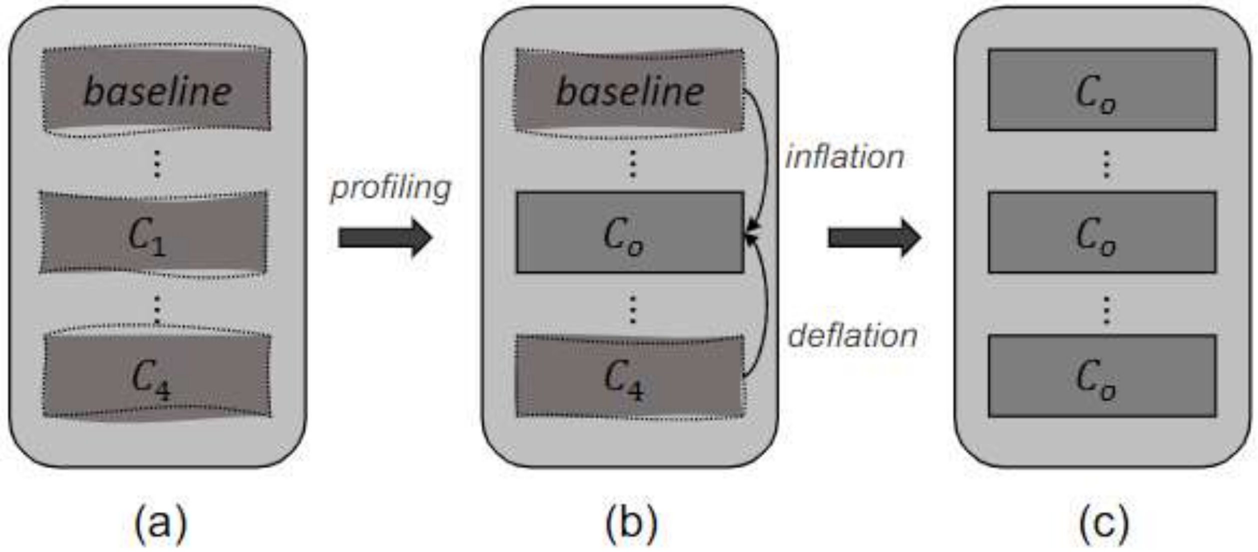



Fig. 6: SMILE RPG flowchart where large rounded rectangle represents GPU and small dark rectangle stands for SM groups (baseline + $C_1 \dots C_4$). There are two primary phases in RPG: profiling (a) $\rightarrow$ (b) and alignment (b) $\rightarrow$ (c).

# SMILE——Result

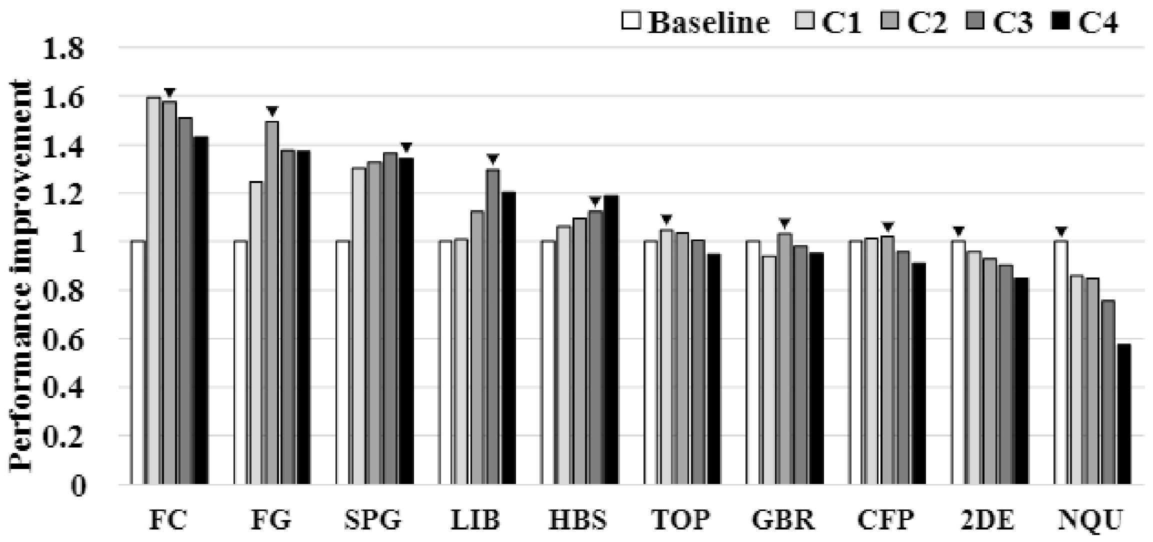



Fig. 8: Performance of varying extra SMEM configurations, i.e., *C1* - *C4*, from which `Ideal` chooses the one resulting in highest speedup. Marked (▼) bars correspond to the eventual decisions made by online profiling in `SMILE`.

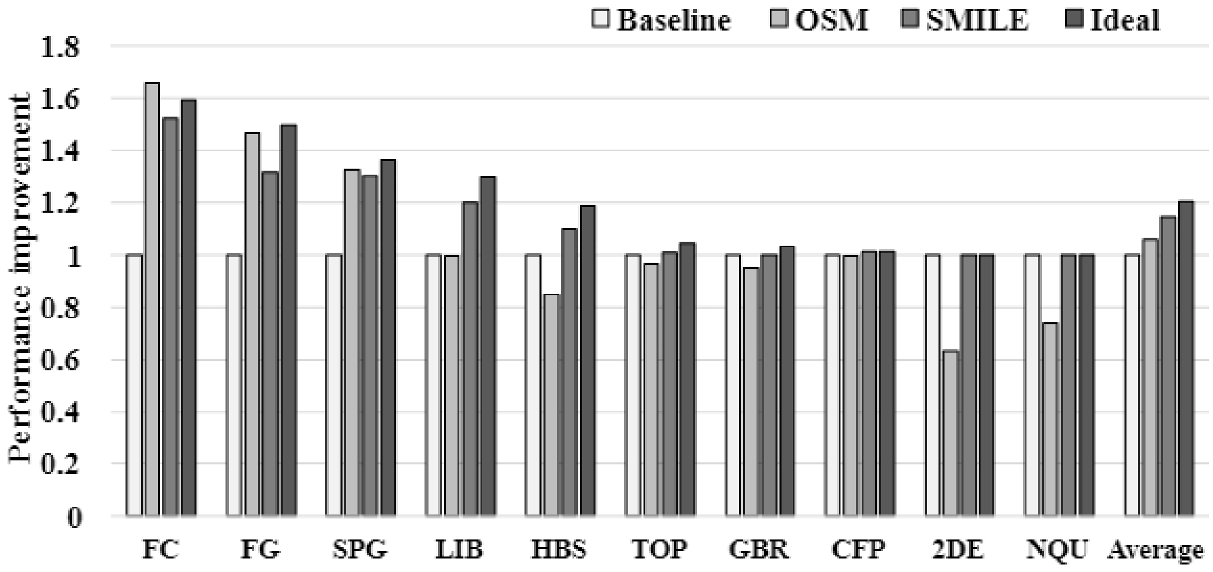



Fig. 7: Comparison of performance improvement among the schemes.

# SYSU-ARCH

https://arcsysu.github.io/SYSU-ARCH/

Search SYSU-ARCH

Dark Mode

- Home
- Before the LAB
- I.Familiar with gem5
- II.Hotspot Analysis
- III.Implement FSUBR
- IV.Implement NMRU cache replacement policy
- V.Explore GPGPU-SIM and GEMM
- VI.Explore ACCEL-SIM and Cache
- FAQ
- arcSYSu on github

# SYSU-ARCH stars 5

## Introduction

> version 2022F

SYSU-ARCH is a LAB that focuses on the use and extending of simulators.

After finishing SYSU-ARCH, you will learn

- what is gem5, Accel-Sim and GPGPU-SIM
- the basic use of gem5, Accel-Sim and GPGPU-SIM
- how to extend in simulator
- how to use simulator to research
- tools like docker and wsl

> reference gem5 101(add **changes** to fit current version of gem5 and new ideas)

# ConvNN https://github.com/gty111/ConvNN

- CPU (C++)
- GPU (CUDNN)

```
nn.add(new Conv(64,3,1,&in));
nn.add(new BatchNorm2D(nn.lastOut()));
nn.add(new Activation(CUDNN_ACTIVATION_RELU,nn.lastOut()));
nn.add(new Conv(64,3,1,nn.lastOut()));
nn.add(new BatchNorm2D(nn.lastOut()));
nn.add(new Activation(CUDNN_ACTIVATION_RELU,nn.lastOut()));
nn.add(new Pooling(CUDNN_POOLING_MAX_DETERMINISTIC,2,2,nn.lastOut()));
nn.add(new Conv(128,3,1,nn.lastOut()));
nn.add(new BatchNorm2D(nn.lastOut()));
nn.add(new Activation(CUDNN_ACTIVATION_RELU,nn.lastOut()));
nn.add(new Conv(128,3,1,nn.lastOut()));
nn.add(new BatchNorm2D(nn.lastOut()));
nn.add(new Activation(CUDNN_ACTIVATION_RELU,nn.lastOut()));
nn.add(new Pooling(CUDNN_POOLING_MAX_DETERMINISTIC,2,2,nn.lastOut()));
nn.add(new Fc(10,nn.lastOut()));
nn.add(new Activation(CUDNN_ACTIVATION_RELU,nn.lastOut()));
nn.add(new Softmax(nn.lastOut(),&label));
```

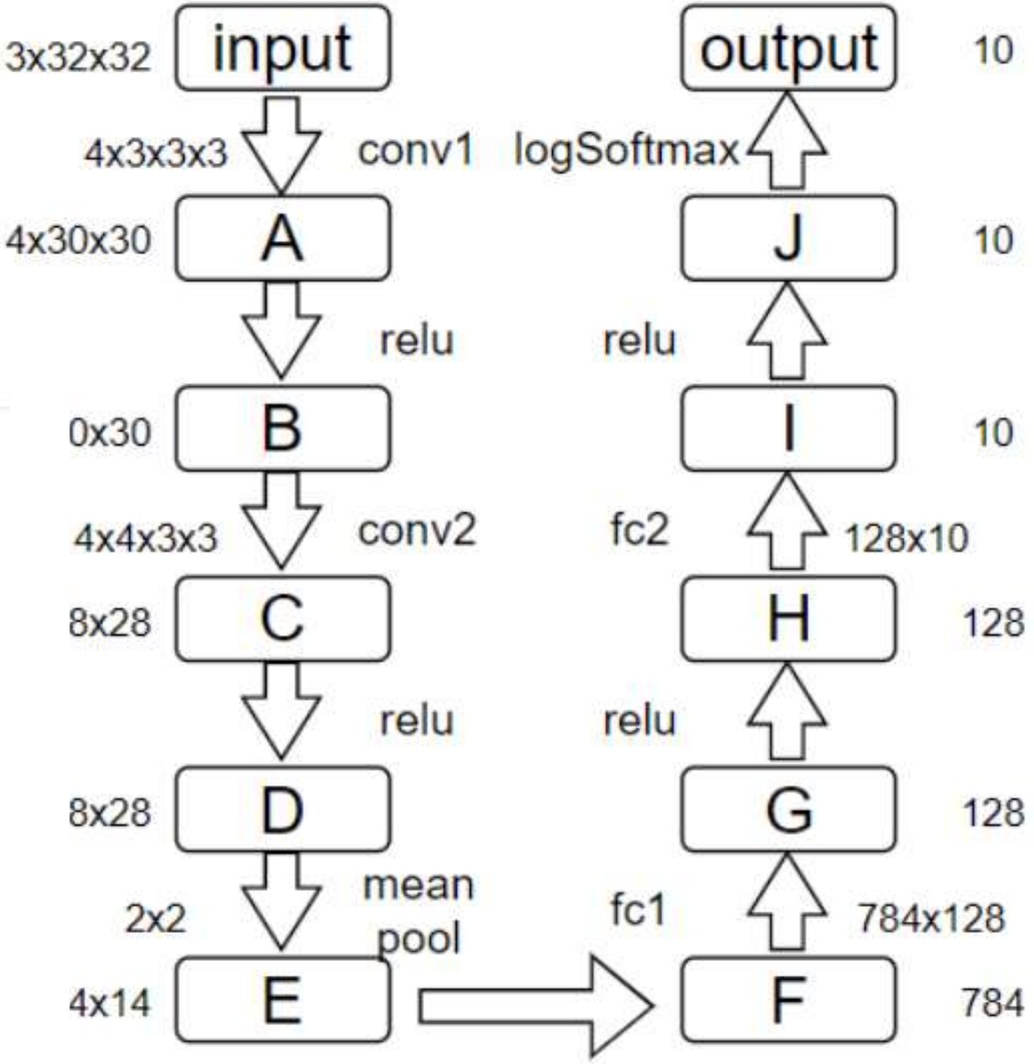
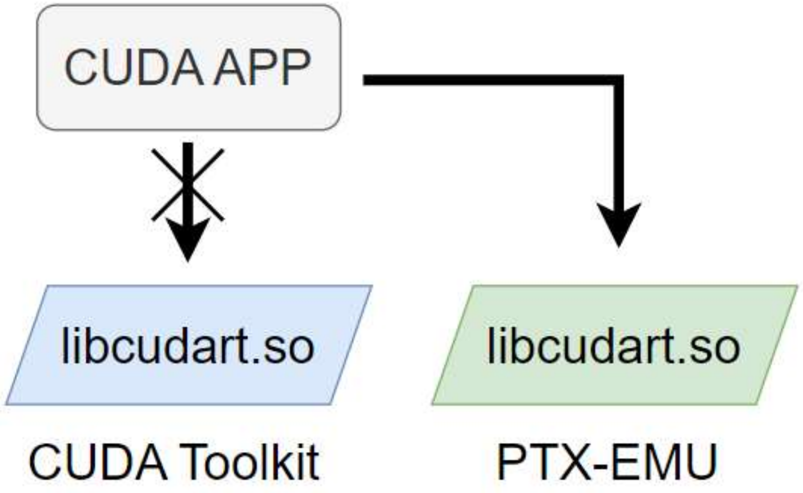



Fig. 12. The model used to train CIFAR-10

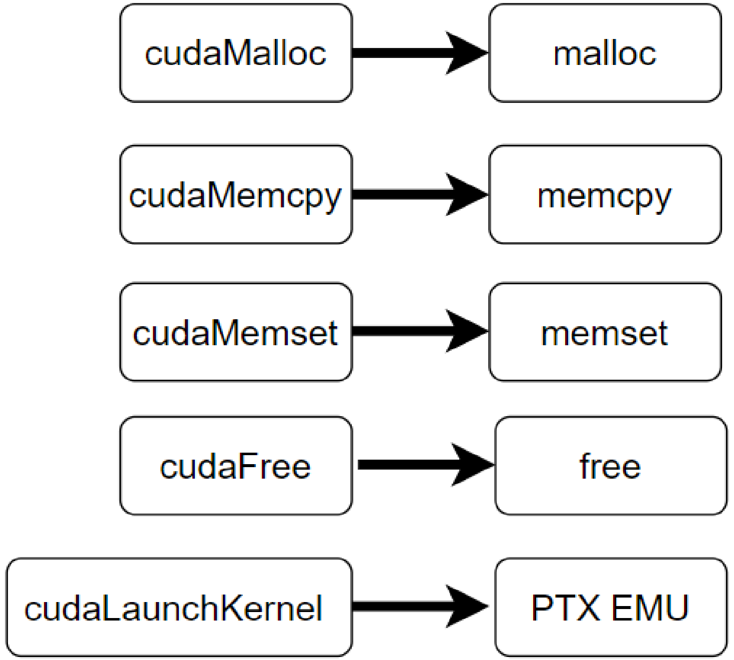
# PTX-EMU 一个简单的CUDA模拟器

https://github.com/gty111/PTX-EMU

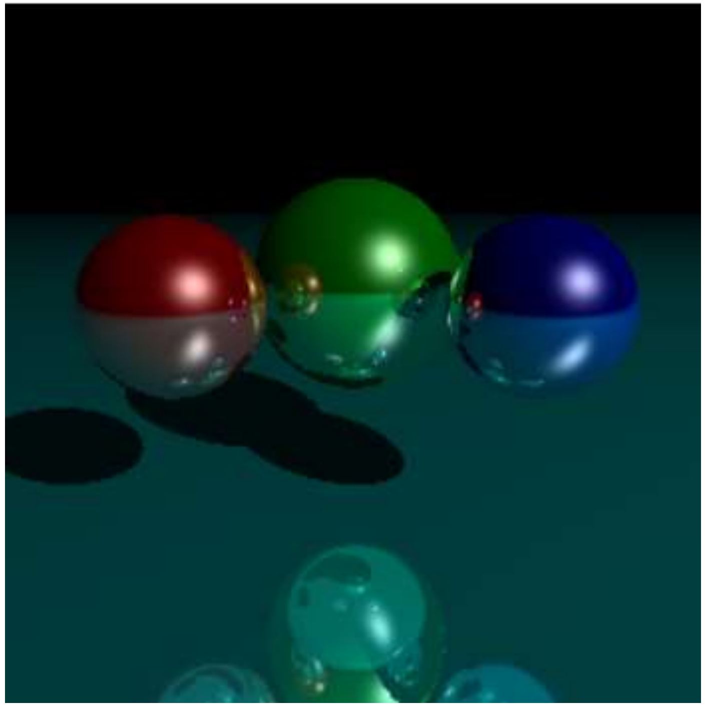
- CUDA运行时库替换



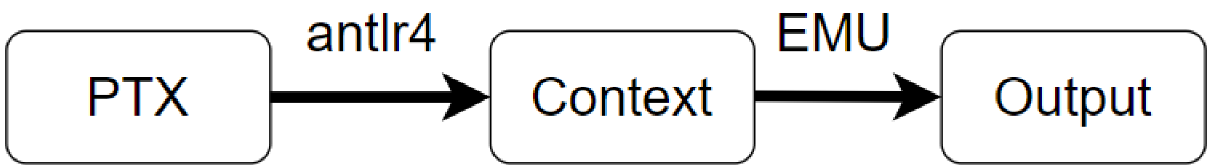

- CUDA运行时模拟




Rodinia 中的 RAY 应用



- PTX仿真



# GEMM-MMA 一步步优化GEMM by tensorcore

https://zhuanlan.zhihu.com/p/638522893

思考：DSA > CPU(GPU)？

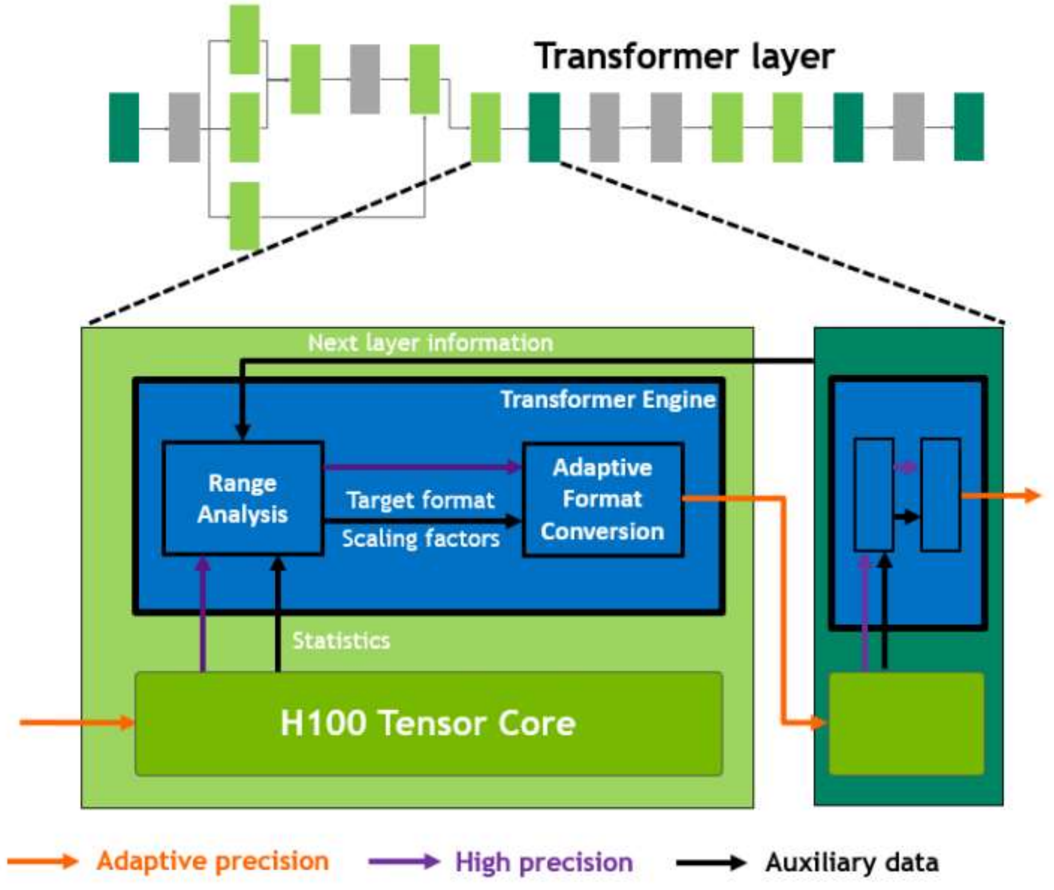


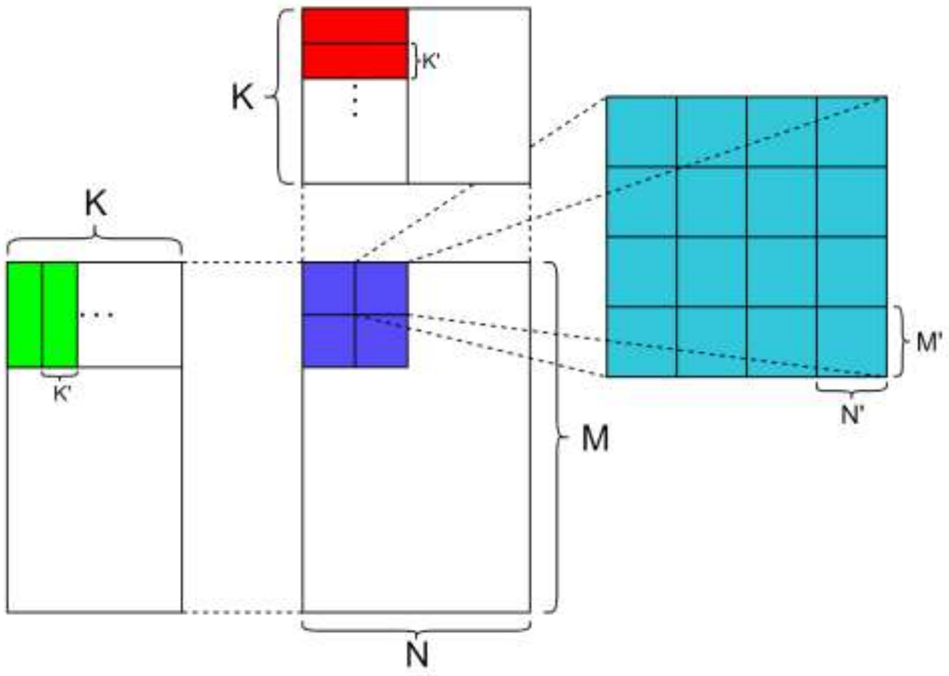

Figure 25. Transformer Engine Conceptual Operation.

Table 2. H100 speedup over A100 (Preliminary H100 Performance, TC=Tensor Core)

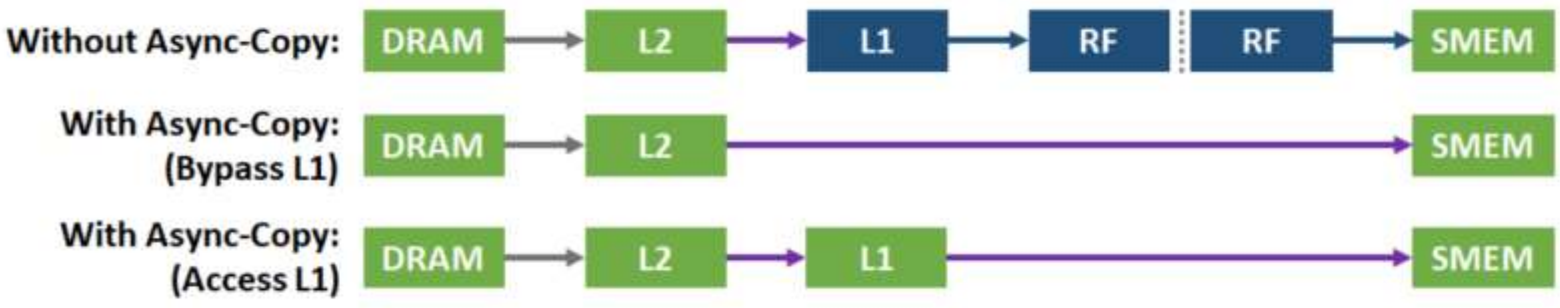
| | A100 | A100 Sparse | H100 SXM5[1] | H100 SXM5[1] Sparse | H100 SXM5[1] Speedup vs A100 |
|---|---|---|---|---|---|
| FP8 Tensor Core | NA | NA | 2000 TFLOPS | 4000 TFLOPS | 6.4x vs A100 FP16 |
| FP16 | 78 TFLOPS | NA | 120 TFLOPS | NA | 1.5x |
| FP16 Tensor Core | 312 TFLOPS | 624 TFLOPS | 1000 TFLOPS | 2000 TFLOPS | 3.2x |
| BF16 Tensor Core | 312 TFLOPS | 624 TFLOPS | 1000 TFLOPS | 2000 TFLOPS | 3.2x |
| FP32 | 19.5 TFLOPS | NA | 60 TFLOPS | NA | 3.1x |
| TF32 Tensor Core | 156 TFLOPS | 312 TFLOPS | 500 TFLOPS | 1000 TFLOPS | 3.2x |
| FP64 | 9.7 TFLOPS | NA | 30 TFLOPS | NA | 3.1x |
| FP64 Tensor Core | 19.5 TFLOPS | NA | 60 TFLOPS | NA | 3.1x |
| INT8 Tensor Core | 624 TOPS | 1248 TOPS | 2000 TFLOPS | 4000 TFLOPS | 3.2x |

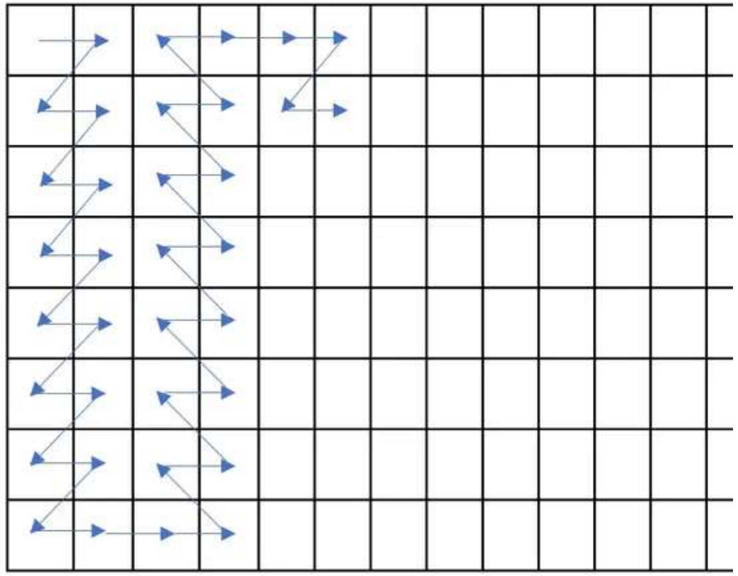
# GEMM-MMA 一步步优化GEMM by tensorcore

https://zhuanlan.zhihu.com/p/638522893




- 向量化load/store
- 异步拷贝
- 数据预取
- 调整线程块和warp计算矩阵大小
- 线程块Swizzle



# THANKS
# Q&A