Night Compute

triton文件生成并使用ui界面分析 .ncu-rep 流程:

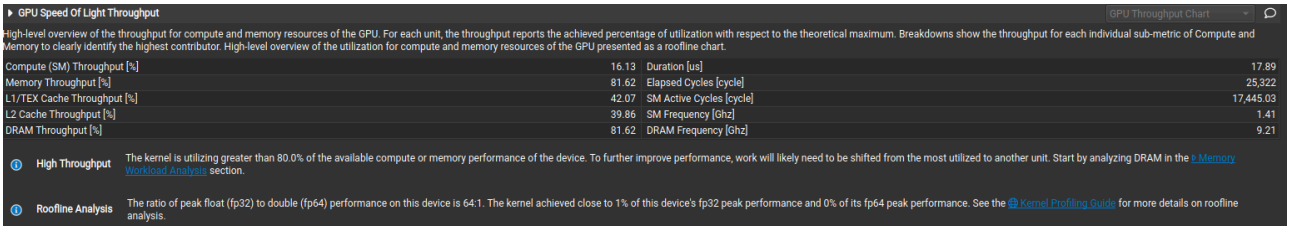
```
yst@yst-X299-WU8 ~/文/y/C/c/C/lec1_profile_CUDA_kernels (main)> /usr/local/NVIDIA-Nsight-Compute-2024.3/ncu --set full -o matrix_squre python triton_sample.py
==PROF== Connected to process 2203444 (/home/yst/miniconda3/envs/yst_pytorch/bin/python3.8)
==PROF== Profiling "distribution_elementwise_grid..." - 0: 0%....50%....100% - 37 passes
==PROF== Profiling "square_kernel_0d1d234" - 1: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 2: 0%....50%....100% - 38 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 3: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 4: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 5: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 6: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 7: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 8: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 9: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 10: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 11: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 12: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 13: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 14: 0%....50%....100% - 37 passes
==PROF== Profiling "vectorized_elementwise_kernel" - 15: 0%....50%....100% - 37 passes
==PROF== Profiling "reduce_kernel" - 16: 0%....50%....100% - 37 passes
passed
==PROF== Disconnected from process 2203444
==PROF== Report: /home/yst/文档/yst/CUDA/cuda-mode/CUDA-Optimization/lec1_profile_CUDA_kernels/matrix_squre.ncu-rep
yst@yst-X299-WU8 ~/文/y/C/c/C/lec1_profile_CUDA_kernels (main)> /usr/local/NVIDIA-Nsight-Compute-2024.3/ncu-ui matrix_squre.ncu-rep
```

源码:

```python
# 装饰器: 使用triton的jit编译器来编译这个函数
# BLCOK_SIZE: thread数量, 用于处理的元素数量, 使用 tl.constexpr 指定为编译时常量
@triton.jit
def square_kernel(output_ptr, input_ptr, output_row_stride, input_row_stride, n_cols, BLOCK_SIZE: tl.constexpr):
    row_idx = tl.program_id(0)  # 获取当前程序id, block_id
    row_start_ptr = input_ptr + row_idx * input_row_stride;
    col_offsets = tl.arange(0, BLOCK_SIZE)  # thread_id
    input_ptrs = row_start_ptr + col_offsets

    # 从GPU中加载数据
    # mask: BLOCK_SIZE = triton.next_power_of_2(n_cols) 可能大于 n_cols
    row = tl.load(input_ptrs, mask=col_offsets < n_cols, other=-float('inf'))

    # 计算加载元素的平方
    square_output = row * row

    output_row_start_ptr = output_ptr + row_idx * output_row_stride
    output_ptrs = output_row_start_ptr + col_offsets
    tl.store(output_ptrs, square_output, mask=col_offsets < n_cols)

def square(x):
    n_rows, n_cols = x.shape
    BLOCK_SIZE = triton.next_power_of_2(n_cols) # 找到最接近的大于所给参数的2^n eg: n_cols: 20 -> 返回32
    num_warps = 4
    if BLOCK_SIZE >= 2048:
        num_warps = 8
    if BLOCK_SIZE >= 4096:
        num_warps = 16

    y = torch.empty_like(x)

    # []: grid_size
    # (n_rows, ): 一维grid
    square_kernel[(n_rows, )](y, x, y.stride(0), x.stride(0), n_cols, BLOCK_SIZE)

    return y
```

Night Compute分析:

1. GPU Spedd of Light Throughtput



GPU Speed Of Light Throughput

High-level overview of the throughput for compute and memory resources of the GPU. For each unit, the throughput reports the achieved percentage of utilization with respect to the theoretical maximum. Breakdowns show the throughput for each individual sub-metric of Compute and Memory to clearly identify the highest contributor. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

Compute (SM) Throughput [%]	16.13	Duration [us]	17.89
Memory Throughput [%]	81.62	Elapsed Cycles [cycle]	25,322
L1/TEX Cache Throughput [%]	42.07	SM Active Cycles [cycle]	17,445.03
L2 Cache Throughput [%]	39.86	SM Frequency [Ghz]	1.41
DRAM Throughput [%]	81.62	DRAM Frequency [Ghz]	9.21

High Throughput The kernel is 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. Start by analyzing DRAM in the Memory Workload Analysis section.

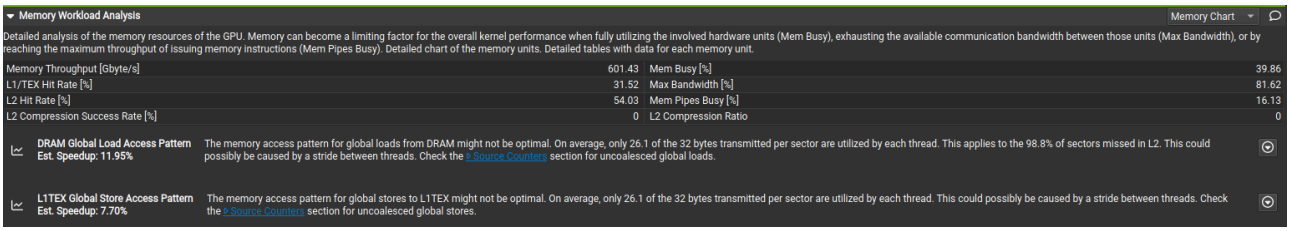
Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 64:1. The kernel achieved close to 1% of this device's fp32 peak performance and 0% of its fp64 peak performance. See the Kernel Profiling Guide for more details on roofline analysis.

可以看出, L1 和 L2 利用率不高, 并且Memory > Compute, 说明是访存密集型算子

eg: x: [H, W]

计算次数: H *W 访存次数: H* W * 2

2. Memory Workload Analysis



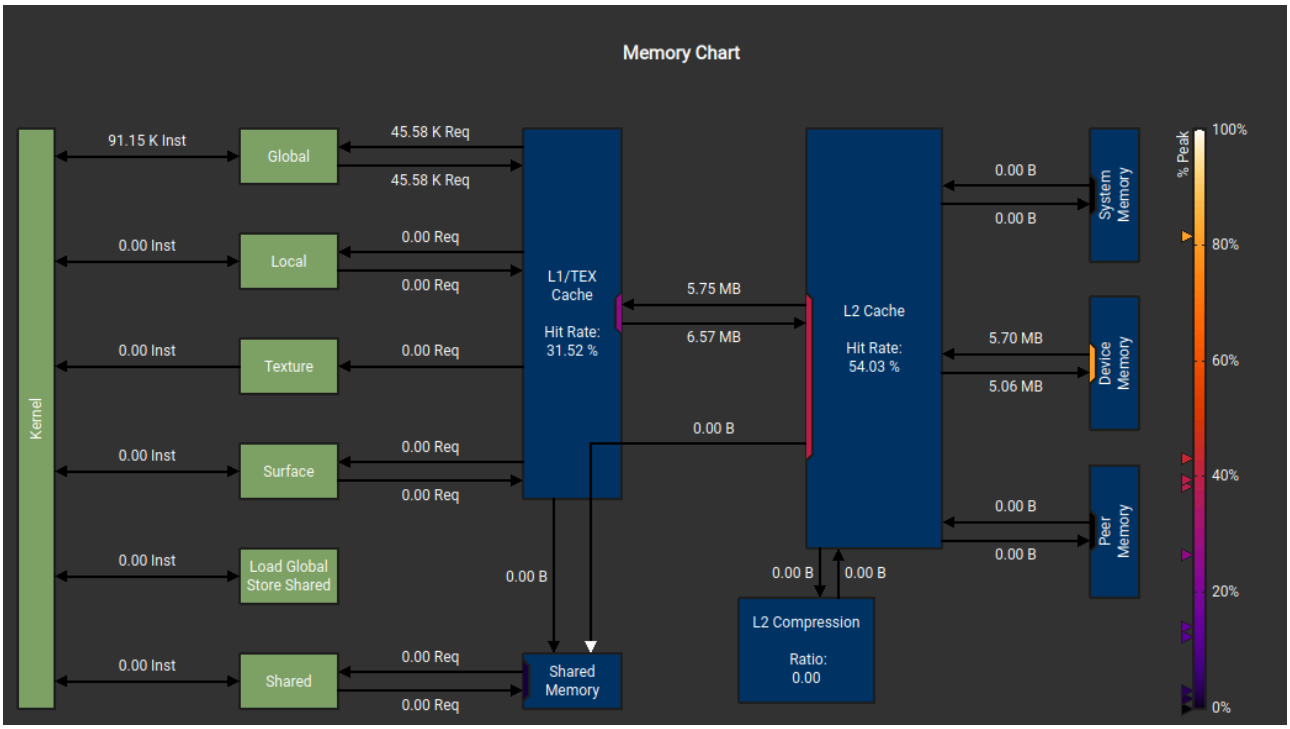
Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit.

Memory Throughput [Gbyte/s]	601.43	Mem Busy [%]	39.86
L1/TEX Hit Rate [%]	31.52	Max Bandwidth [%]	81.62
L2 Hit Rate [%]	54.03	Mem Pipes Busy [%]	16.13
L2 Compression Success Rate [%]	0	L2 Compression Ratio	0

DRAM Global Load Access Pattern Est. Speedup: 11.95% The memory access pattern for global loads from DRAM might not be optimal. On average, only 26.1 of the 32 bytes transmitted per sector are utilized by each thread. This applies to the 98.8% of sectors missed in L2. This could possibly be caused by a stride between threads. Check the Source Counters section for uncoalesced global loads.

L1TEX Global Store Access Pattern Est. Speedup: 7.70% The memory access pattern for global stores to L1TEX might not be optimal. On average, only 26.1 of the 32 bytes transmitted per sector are utilized by each thread. This could possibly be caused by a stride between threads. Check the Source Counters section for uncoalesced global stores.

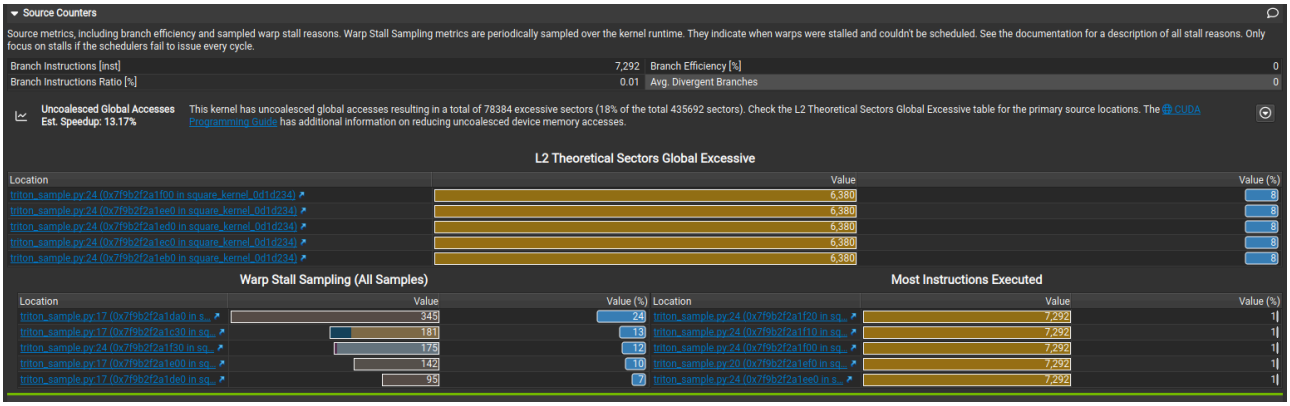


L1 和 L2 Hit Rate 不高, Max BandWidth很低(3080： 790GB/s)

eg: x: [1823, 781] FP32

数据量: 1823 *781* 32 / 8 / 1024 / 1024 = 5.43MB

3. Source Counters



分支指令:

一共7292个分支, 分支效率和平均发散分支都是0 -> 分支预测做得很好(因为确实没有分支)

未合并的全局访问:

有未合并的全局访问, 导致78384个多余的sector

L2 未有效利用:

问题出在24行

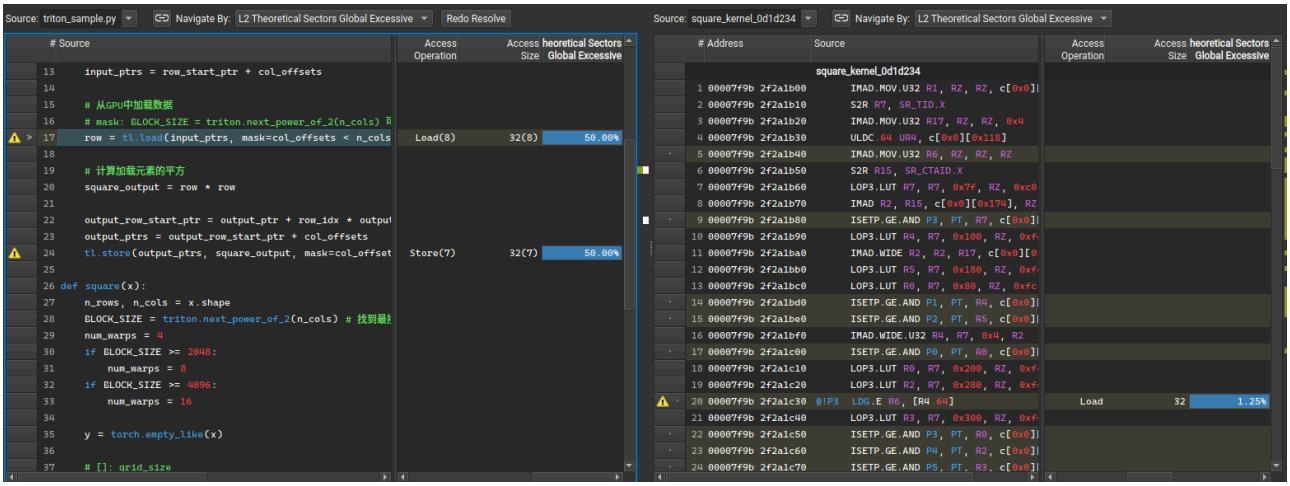
tl.store(output_ptrs, square_output, mask=col_offsets < n_cols)

Warp停滞

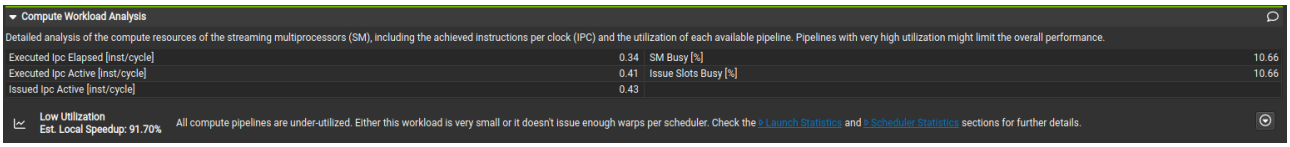
问题处在17和24行(一个读一个存)

row = tl.load(input_ptrs, mask=col_offsets < n_cols, other=-float('inf'))

tl.store(output_ptrs, square_output, mask=col_offsets < n_cols)



4. Compute Workland Analysis



计算pipeline没有被充分利用(因为是访存密集)

5. Launch Statistics