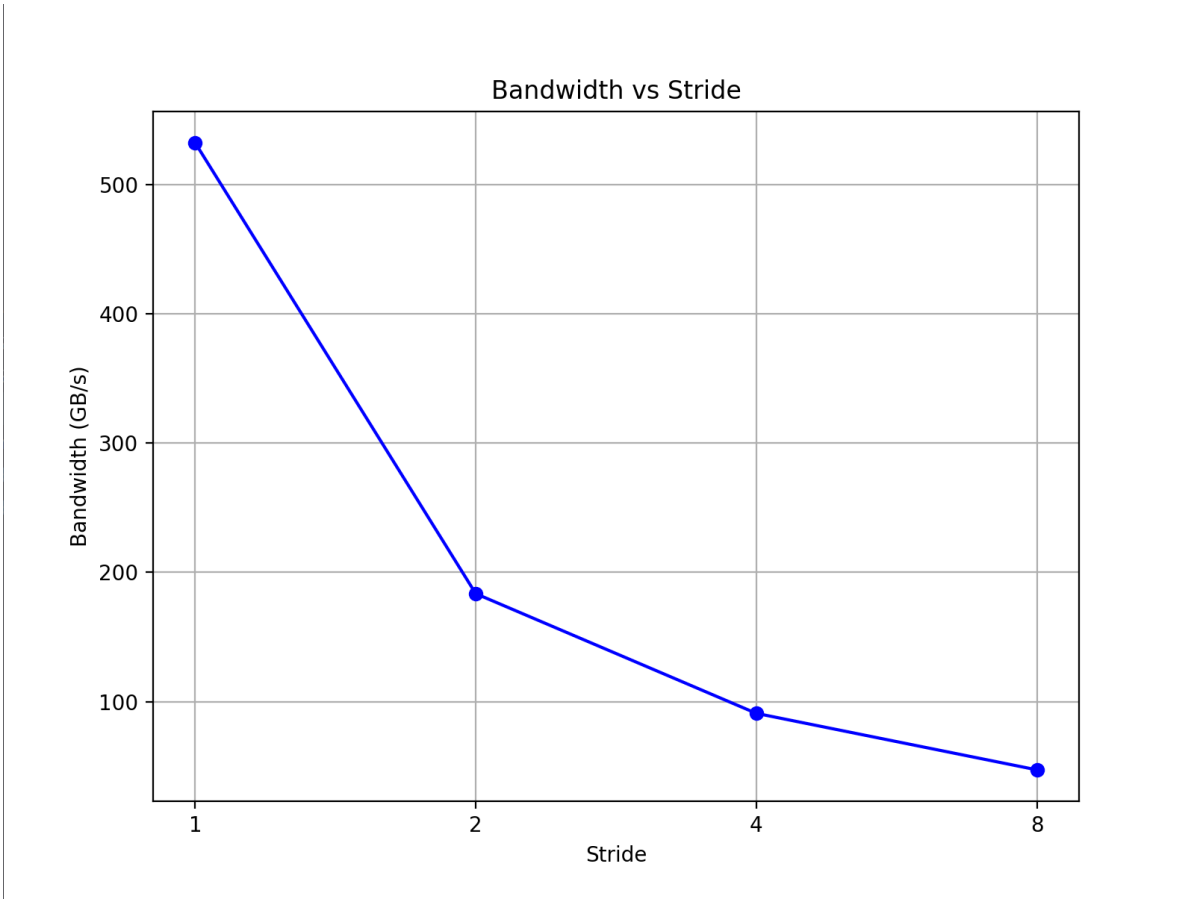
小作业六报告

宋建昊 2022010853

实验结果

gmem

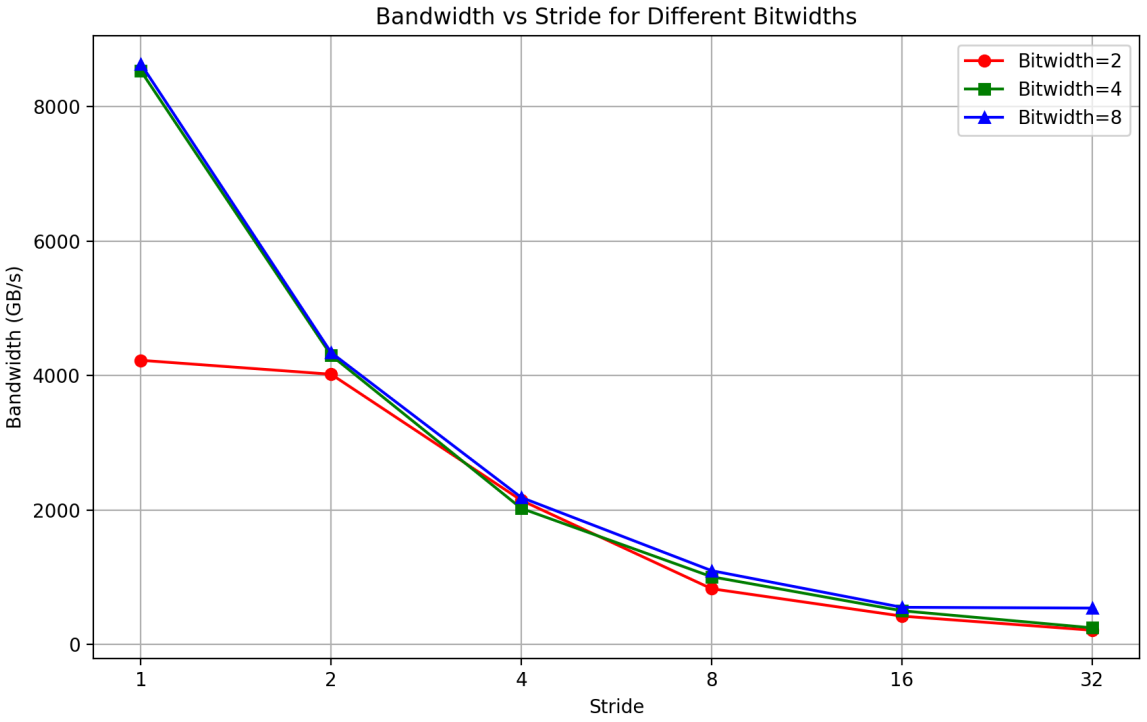
stride	bandwidth
1	532.121
2	183.471
4	90.9098
8	47.2128



smem

bitwidth	stride	bandwidth
4	1	8529.12
4	2	4301.24
4	4	2025.68
4	8	1009.35

bitwidth	stride	bandwidth
4	16	503.37
4	32	249.593
2	1	4227.10
2	2	4019.72
2	4	2145.04
2	8	831.123
2	16	423.803
2	32	214.139
8	1	8632.03
8	2	4342.10
8	4	2187.54
8	8	1099.67
8	16	554.569
8	32	543.169



`test_gmem.cu` 性能分析

性能变化的主要来源

GPU 机制：全局内存合并访问（Memory Coalescing）与事务效率

- 机制描述：GPU 的全局内存以较大块（通常为 32 字节或 128 字节，按缓存行对齐）访问。CUDA 通过内存合并优化性能，将一个 warp（32 个线程）的内存请求合并为尽可能少的事务（transactions），从而减少内存访问延迟并提高带宽利用率。
- 对性能的影响：
  - Stride=1：线程访问连续的内存地址（例如，src[0], src[1], src[2], ...），每个 float 为 4 字节，32 个线程的请求（128 字节）正好可以合并为一个或最少的事务。这种模式充分利用了全局内存的带宽，带宽达到 532.121 GB/s，接近高端 GPU（如 NVIDIA A100，理论带宽约 600-900 GB/s）的峰值。
  - Stride ≥ 2：线程访问非连续地址（例如，stride=2 时：src[0], src[2], src[4], ...，间隔 8 字节）。这破坏了内存合并，每个线程的请求可能需要单独的事务。例如，stride=8（间隔 32 字节）可能导致 32 个线程的请求分散到多个缓存行，显著增加事务数量。事务效率下降导致带宽急剧降低，例如 stride=8 时仅为 47.2128 GB/s（下降约 11 倍）。
  - 量化影响：带宽下降与事务数量成反比。假设每个事务为 128 字节，stride=1 时一个 warp 需要 1 次事务，而 stride=8 时可能需要接近 32 次事务（最坏情况），直接导致带宽下降。

该机制如何影响程序性能

- 高效情况（Stride=1）：内存合并使 32 个线程的请求合并为单个事务，最大化利用全局内存的带宽。每次事务传输 128 字节数据，延迟被均摊，带宽接近硬件极限。
- 低效情况（Stride ≥ 2）：非连续访问导致每个线程的请求触发独立的事务，增加内存控制器的负担。全局内存的延迟较高（数百个周期），事务数量增加直接导致带宽下降。例如，stride=4 时，间隔 16 字节，可能需要 8 次事务（128 字节 / 16 字节），带宽降至 90.9098 GB/s。
- 缓存影响：GPU 的 L1 和 L2 缓存会部分缓解非连续访问的影响，但当 stride 较大时，缓存命中率下降（地址分散），更多请求直接访问全局内存，进一步降低性能。
- 其他硬件功能及其对执行效率的影响

参与程序执行的其他硬件功能及其影响

- L1/L2 缓存：
  - 作用：全局内存访问通常通过 L1 和 L2 缓存，缓存行大小为 128 字节。Stride=1 时，缓存命中率高（连续地址落入同一缓存行），减少全局内存访问。
  - 影响：Stride 增加时，缓存命中率下降。例如，stride=8（间隔 32 字节）可能导致每个线程访问不同的缓存行，增加缓存未命中（cache miss）。这迫使更多请求直接访问全局内存，增加延迟并降低带宽。
- 内存控制器与带宽：
  - 作用：全局内存通过内存控制器访问，带宽受限于 DRAM 的物理带宽（例如，HBM3 或 GDDR6）。
  - 影响：Stride=1 时，内存控制器高效调度连续请求，接近峰值带宽。Stride 增加时，分散的请求导致控制器调度效率下降，带宽利用率降低。
- 线程调度与 warp 执行：
  - 作用：GPU 以 warp 为单位调度线程（32 个线程并行执行）。内存合并依赖于 warp 内线程的访问模式。
  - 影响：Stride=1 时，warp 内线程的请求对齐，合并效果最佳。Stride 增加时，warp 内请求分散，增加内存事务，降低执行效率。
- 指令流水线：
  - 作用：核函数仅执行简单的内存拷贝（dst[i * STRIDE] = src[i * STRIDE]），计算开销极低，性能瓶颈完全由内存访问决定。

- 影响：内存事务效率直接决定了程序的吞吐量，stride 增加导致内存延迟暴露，降低整体效率。

`test_smem.cu` 性能分析

固定 BITWIDTH 时，程序性能变化的硬件机制

GPU 机制：共享内存的银行冲突（Bank Conflict）

- 机制描述：GPU 的共享内存分为多个银行（bank，通常 32 个，每个银行 4 字节宽，基于 Volta/Ampere 架构）。每个 warp 的线程并行访问共享内存，若多个线程访问同一银行的不同地址，会发生银行冲突，导致访问串行化，降低带宽。
- 对性能的影响：
  - Stride=1：线程访问连续地址（shm[threadIdx.x]），例如，threadIdx.x=0,1,2,... 访问 shm[0], shm[1], shm[2], ...。假设 bitwidth=4（4 字节），每个线程访问不同银行（银行索引 = 地址 / 4 字节 % 32），无银行冲突。带宽极高，例如 bitwidth=4 时为 8529.12 GB/s，接近共享内存的理论峰值（现代 GPU 共享内存带宽可达数 TB/s）。
  - Stride ≥ 2：线程访问非连续地址（shm[threadIdx.x * STRIDE]），地址间隔为 STRIDE * BITWIDTH 字节，可能导致多个线程访问同一银行。例如：
    - Stride=2，bitwidth=4：间隔 8 字节，银行索引可能重复（例如，threadIdx.x=0 访问 shm[0] 在 bank 0，threadIdx.x=1 访问 shm[2] 在 bank 2，但某些线程可能冲突）。
    - Stride=32，bitwidth=4：间隔 128 字节，可能导致所有线程访问同一银行（银行索引 = (threadIdx.x * 128) / 4 % 32），造成严重冲突。
  - 量化影响：银行冲突使内存请求串行化，带宽与冲突线程数成反比。例如，stride=32 时，带宽降至 249.593 GB/s（约 34 倍下降），表明几乎所有线程访问同一银行，串行化程度极高。

    BITWIDTH=2 和 BITWIDTH=8 相比 BITWIDTH=4 的性能变化趋势及原因
- 趋势概述：
  - Bitwidth=4（基准）：
    - Stride=1 时带宽最高（8529.12 GB/s），随 stride 增加快速下降，至 stride=32 时为 249.593 GB/s。
    - 性能表现稳定，带宽与银行冲突程度直接相关。
  - Bitwidth=2：
    - Stride=1 时带宽较低（4227.10 GB/s，仅为 bitwidth=4 的 49.6%），但随 stride 增加的下降速度较慢（例如，stride=32 时为 214.139 GB/s，下降约 20 倍）。
    - 对于小 stride（如 1、2），带宽接近 bitwidth=4，但在高 stride 下表现略差。
  - Bitwidth=8：
    - Stride=1 时带宽最高（8632.03 GB/s，略高于 bitwidth=4），随 stride 增加下降趋势与 bitwidth=4 类似，但高 stride 下表现稍好（例如，stride=32 时为 543.169 GB/s，下降约 16 倍）。
    - 整体带宽优于 bitwidth=4，尤其在高 stride 下。
- 原因分析：
  - Bitwidth=2（uint16_t，2 字节）：

        - 低带宽原因：共享内存银行宽度为 4 字节（Volta/Ampere 架构）。访问 2 字节数据无法充分利用银行带宽，可能需要额外的事务来对齐数据。例如，threadIdx.x=0 访问 shm[0]（2 字节，可能只占用银行 0 的半宽），导致带宽浪费。Stride=1 时带宽仅为 4227.10 GB/s，远低于 bitwidth=4。
        - 下降较缓：由于每次访问的数据量小（2 字节），高 stride 下的银行冲突影响相对较轻。例如，stride=32（间隔 64 字节），银行索引分布可能更均匀，冲突程度低于 bitwidth=4 或 8。
        - 趋势差异：相比 bitwidth=4，bitwidth=2 在小 stride 下表现较差（因银行利用率低），但高 stride 下带宽下降幅度较小，表明其对银行冲突的敏感性较低。
    - Bitwidth=8（uint64_t，8 字节）：
        - 高带宽原因：8 字节数据跨越两个连续银行（例如，访问 shm[0] 占用 bank 0 和 bank 1）。Stride=1 时，线程访问连续地址，银行分配均匀，无冲突，且每次访问传输更多数据（8 字节 vs 4 字节），带宽略高于 bitwidth=4（8632.03 GB/s）。
        - 高 stride 优势：高 stride 下（例如，stride=32，间隔 256 字节），8 字节访问可能仍能利用多个银行（银行索引跨度更大），冲突程度低于 bitwidth=4。例如，stride=32 时带宽为 543.169 GB/s，优于 bitwidth=4 的 249.593 GB/s。
        - 趋势差异：相比 bitwidth=4，bitwidth=8 在所有 stride 下带宽略高，尤其在高 stride 下更明显，表明其对银行冲突的容忍度更高，且每次访问的数据量大提高了效率。
    - Bitwidth=4（uint32_t，4 字节）：
        - 基准表现：4 字节与银行宽度完美对齐，stride=1 时无冲突，带宽高（8529.12 GB/s）。高 stride 下冲突显著（例如，stride=32 时所有线程可能访问同一银行），带宽下降明显。
        - 中间特性：相比 bitwidth=2，带宽更高（因充分利用银行宽度）；相比 bitwidth=8，数据量较小导致带宽略低，尤其在高 stride 下冲突更明显。
- 银行冲突的具体影响：
    - 假设 32 个银行，每个 4 字节宽，线程访问地址为 threadIdx.x * STRIDE * BITWIDTH 字节：
        - Stride=1：无论 bitwidth，所有线程访问不同银行，无冲突。
        - Stride=32：
            - Bitwidth=2：间隔 64 字节，银行索引 = (threadIdx.x * 64) / 4 % 32，分布较均匀，冲突较少。
            - Bitwidth=4：间隔 128 字节，银行索引可能集中（例如，所有线程访问 bank 0），冲突严重。
            - Bitwidth=8：间隔 256 字节，访问跨多个银行，冲突程度介于两者之间。
    - Bitwidth=8 的高数据量和跨银行访问使其在高 stride 下更高效；bitwidth=2 的小数据量使其对冲突不敏感，但基础带宽低。