Operating System (CSC 3150)

Tutorial 7

SHIHAO HONG

SCHOOL OF SCIENCE AND ENGINEERING

E-MAIL: 220019037@LINK.CUHK.EDU.CN

03, NOV 2021

Target

In this tutorial, we will discuss Assignment 3.

Assignment 3 Code Structure

- CUDA Global Memory (take it as a secondary storage)
  - Typically implemented in DRAM
  - High access latency: 400-800 cycles
  - In Assignment 3, size is 128KB

- CUDA Shared Memory (take it as a physical memory)
  - Extremely fast
  - Configurable cache
  - Memory size is small (16 or 48 KB)
  - In Assignment 3, size is 48KB (32KB for data accessing, 16KB for page table setting)

Assignment 3 Code Structure

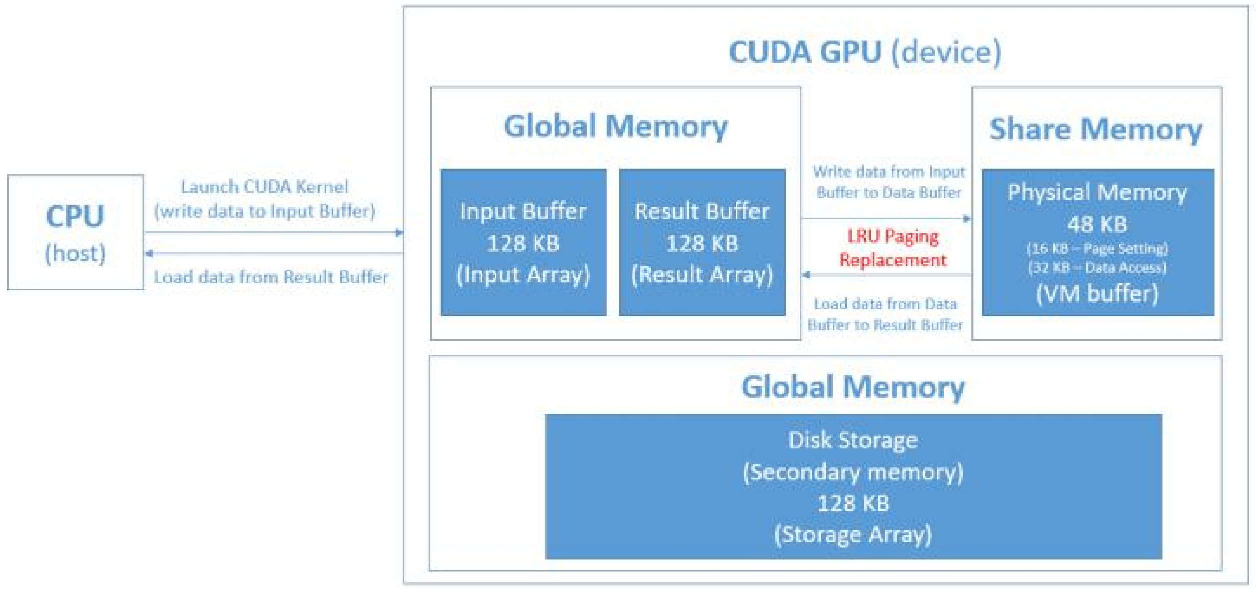
- In main function (executed in host), load the binary file “data.bin” and calculate the data size.
- Initialize the page table and entries when entering GPU kernel.
- Under vm_write, you should complete the program to write data into buffer (physical memory).
- Under vm_read, you should complete the program to read data from buffer (physical memory).
- Under Snapshot, you should complete the program to load the elements of buffer (in shared memory, as physical memory) to results buffer (in global memory, as secondary storage), using LRU algorithm.

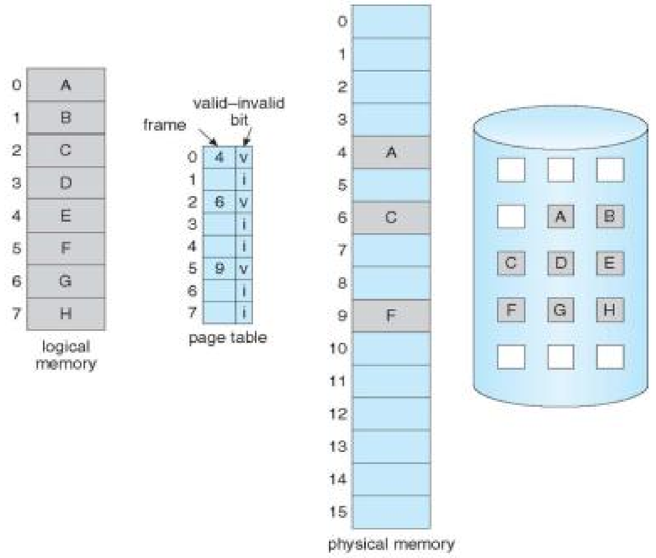
Assignment 3 Code Structure

- Count the page_fault when executing paging replacement.
- In Host, dump the contents of binary file into “snapshot.bin”
- Print out PAGEFAULT when the program finish execution.

Assignment 3 Discussion

- Code structure
  - Physical Memory (Share Memory): 16KB for page settings and 32KB for data access
  - Secondary Storage (Global Memory): 128KB for vm_storage





Assignment 3 Discussion

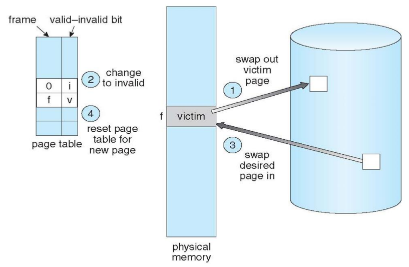
- How to control data access in different memory in CUDA
  - Share Memory: VM buffer(it is declared as “__shared__”)
  - Share Memory: page table (it is declared as “extern __shared__” and size is defined in third parameter in kernel launching function. It will be auto allocated when kernel is launched.)
  - Global Memory: Input buffer / Result array (they are declared as “__device__ __managed__” or use “cudaMalloc”)
  - Global Memory: vm_storage buffer (it is declared as “__device__”)

Assignment 3 Discussion

- Page Table
  - Page table setting is initialized as 16KB.
  - Page table has 1024 entries. (32 KB / 32 bytes)
  - invert_page_table[i] (from 0 to PAGE_ENTRIES - 1) stores valid-invalid bit (initialized as false)
  - invert_page_table[i] (from PAGE_ENTRIES to 2 * PAGE_ENTRIES) stores page number.
  - Page table: Index indicates the page number in logical memory; Value indicates the frame number in physical memory.
  - Hint: Considering situation that real page number exceeds PAGE_ENTRIES, should redesign the page table. Invert page table is one of the design could meet the requirement. (Index indicates frame number in physical memory, and value indicates the page number in logical memory)

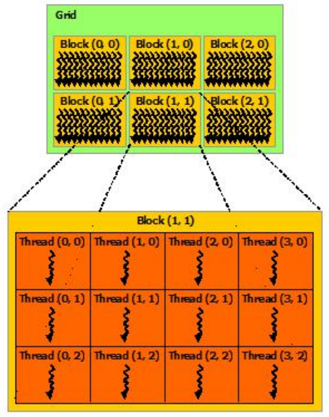
Assignment 3 Discussion

- Page Replacement
  - When writing data to vm_buffer, if share memory is available, place data to available page.
  - Otherwise, replace the LRU set. Swap the least recent used page out of share memory and swap it in secondary storage.
  - Swap the designed page into share memory for data writing.
  - Update the page table.
  - When reading data from vm_buffer, if such page exits in share memory, read it out.
  - Otherwise, replace the LRU set. Swap the least recent used page out of share memory and swap it in secondary storage.
  - Swap the designed page into share memory for data reading.
  - Update the page table.



Assignment 3 Discussion

- Bonus Hints
  - When laucnhing the kernel, defining the numbers of kernels per block.
  - Thread index: threadId.x; threadId.y
  - It can specify synchronization points in the kernel by calling the __syncthreads() intrinsic function; __syncthreads() acts as a barrier at which all threads in the block must wait before any is allowed to proceed.



Reference

- CUDA programming guide
  - https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Thank you