CSC3150 Project3 Report

Liang MingRui 120090723

2022.11.7

Environment of running my program:

I use the environment in the cluster provided by TA.

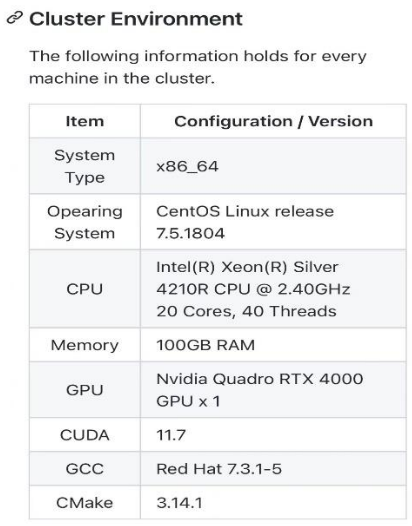
Brief information:

OS is LINUX

VS code version is 1.73

CUDA version is 11.7.99

A screenshot of information:



Cluster Environment

The following information holds for every machine in the cluster.

Item	Configuration / Version
System Type	x86_64
Opearing System	CentOS Linux release 7.5.1804
CPU	Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz 20 Cores, 40 Threads
Memory	100GB RAM
GPU	Nvidia Quadro RTX 4000 GPU x 1
CUDA	11.7
GCC	Red Hat 7.3.1-5
CMake	3.14.1

Execution steps of running your program:

There is a .sh file provided by TA in the folder. I can directly use command "sbatch slurm.sh" to run my program.

My way to design my program:

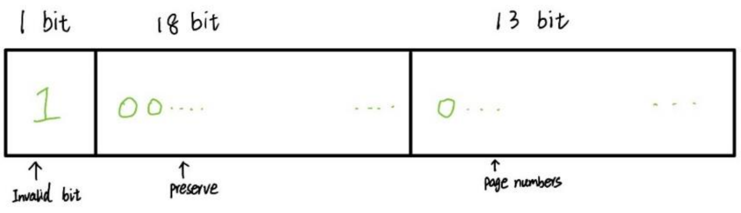
Source:

In order to complete the mission to write and read, I have to take care of lots of things including page table, LRU, swap table and so on.

1. Implementation of inverted page table.

I use this table to storage the relationship between page number and the physical memory. When the content in the physical memory needs to be swapped to second memory, the correlated entry needs to be put into swap table.

For one entry of page table, I use the LSB as the invalid bit to show if the status in this position is valid or not. (1 means invalid; 0 means valid) Then using the last 13 bit to store the page number. (using 13 bit is to support 160KB). The rest bits are kept as preserve.

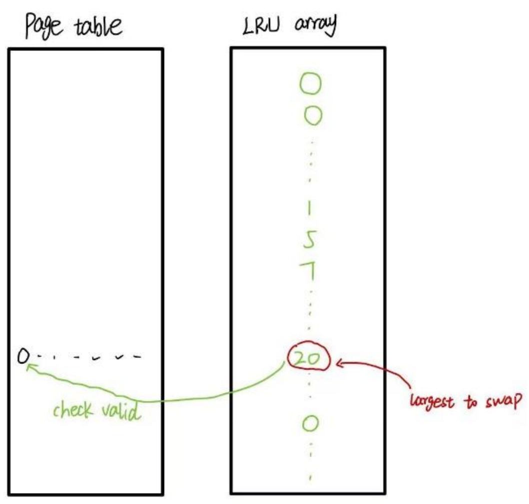


In my inverted page table, I set 1024 entries to support my program, which use 4KB.

2. LRU

I use array to implement the function of LRU. I firstly create 1024 entries to count the number, and set these as 0. Every time, the operation write or read affects the inverted page table, the program will first check whether the entry in page table is valid, if it is valid, the count in the array will reset to 0. Other entries will plus 1 if they are valid. When the program needs to do swapping, it will find the largest count in the array to swap.

This struct use 4KB.



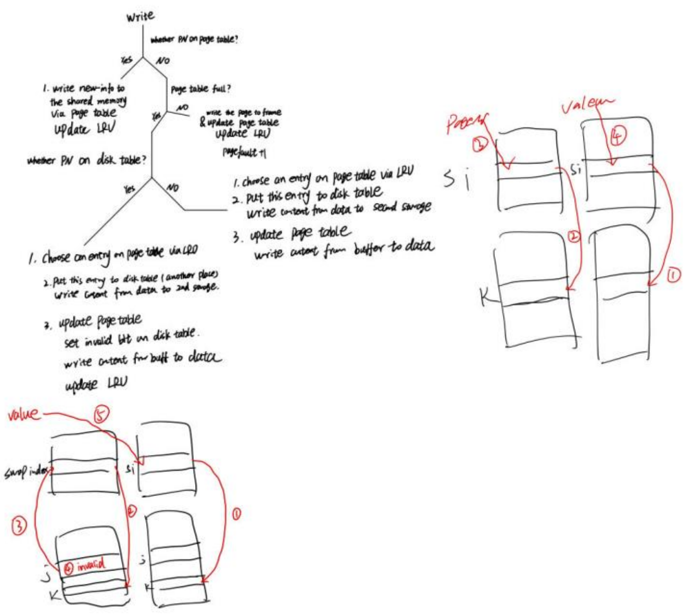
3. Inverted swap table:

In order to record the place of content swapped out from physical memory to second storage, I build a swap table. Due to the second storage's size is smaller than input buffer, I use inverted page table to store the relationship between the page number and the address of second storage. In this table, the entries are the same as I just mentioned in the inverted page table. Also, when the entries in the second storage need to be swapped back to physical memory, the related entry in the swap table need to be cleaned and move back to page table. Moreover, there are 4*1024 entries in this swap table.

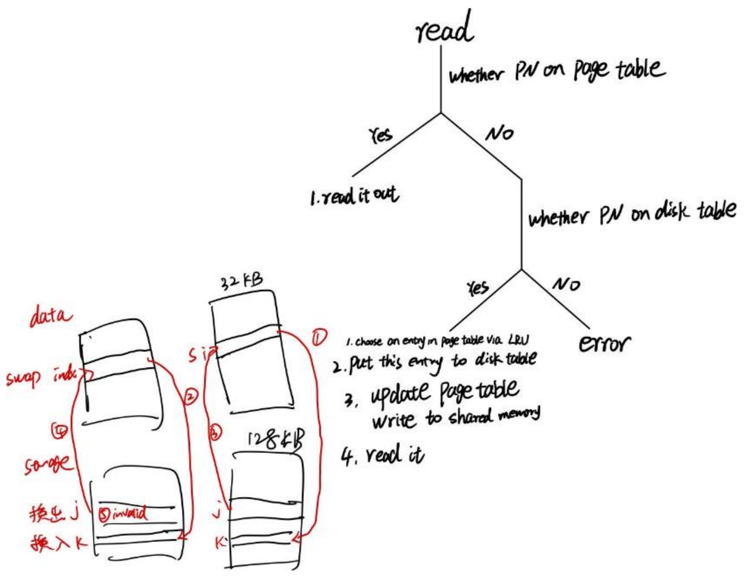
4. General design:

I use to tree struct to show all the cases in the function write and read.

vm_write:


write
whether PN on page table?
Yes
No
1. write new-info to the shared memory via page table
update LRU
page table full?
Yes
No
write the page to frame & update page table
update LRU
pagefault +1
whether PN on disk table?
Yes
No
1. choose an entry on page table via LRU
2. put this entry to disk table
write content from data to second storage
3. update page table
write content from buffer to data
1. choose an entry on page table via LRU
2. put this entry to disk table (another place)
write content from data to 2nd storage.
3. update page table
set invalid bit on disk table.
write content from buff to data
update LRU
value
swap index
si
invalid
j
k
Si
valeu
K

vm_read:



vm_snapshot:

This function will call read function I have implemented before to do reading from given offset to input_size plus offset.

Bonus (version 2 to finish: 4 threads do the same thing four times):

In bonus, I just add some codes in main.cu. In this file, I change the code call kernel function to <<1,4>> to create 4 threads. And then I use atomic and "__syncthread()"(using vm.id to store the current thread id to ask each thread to do their own work and other threads wait until the current thread ends) to ask the four threads to execute one by one by their ids. Also before every thread begin, I clean the content in page table and swap table to do overwriting. Then the task is completed.

Page fault number and analysis:

Source:

Page fault number is 8193.

Analysis:

First, write will have page fault number 4096, and the read will only have 1 page fault. The snapshot will have 4096 page fault number. Thus, in total, the page fault number is 8193.

Bonus:

Page fault number is 32772.

Analysis:

The four threads do the same thing in source four times, and every new thread come to do work, it will overwrite previous content. Thus, the page fault number is 4*8193 = 32772.

Problem I met in the assignment:

1. I did not know the difference between physical memory and so-called virtual memory.

   Solution: I read the ppt and ask TA to figure out.

2. I need to find the oldest entry to swap in page table.

   Solution: I use the array to store how many times the entries were used and swap the count number highest one.

3. I need a struct to tell me where the swapped content is placing now.

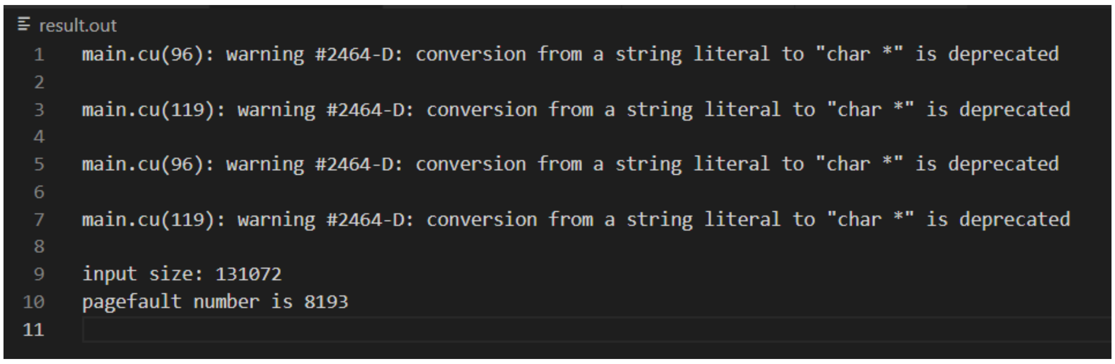
   Solution: I directly build a swap table connect the page number and the address of second storage. In this way, I can directly use page number to find the address of the page in the second storage.

4. I am not so clear about the cases happening in those function.

   Solution: I print the decision tree to list all cases that will happen in this program. After that, my mind is more clearer.

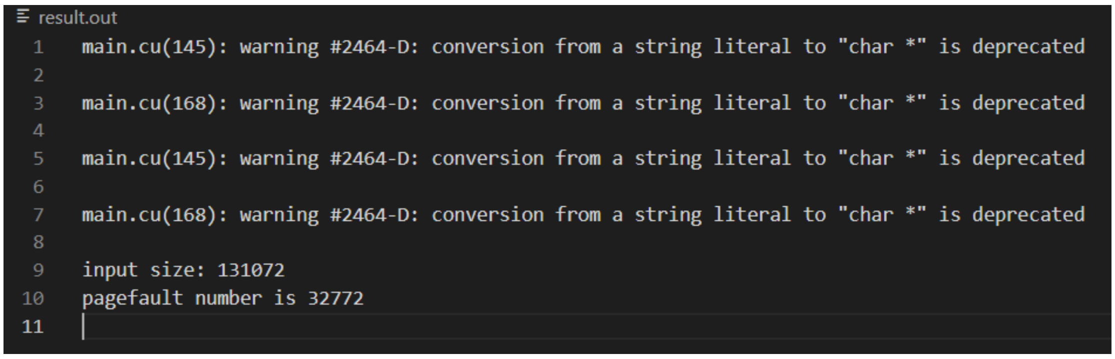
Screenshot:

Source:



```
result.out
main.cu(96): warning #2464-D: conversion from a string literal to "char *" is deprecated

main.cu(119): warning #2464-D: conversion from a string literal to "char *" is deprecated

main.cu(96): warning #2464-D: conversion from a string literal to "char *" is deprecated

main.cu(119): warning #2464-D: conversion from a string literal to "char *" is deprecated

input size: 131072
pagefault number is 8193

```

Bonus:



```
result.out
main.cu(145): warning #2464-D: conversion from a string literal to "char *" is deprecated

main.cu(168): warning #2464-D: conversion from a string literal to "char *" is deprecated

main.cu(145): warning #2464-D: conversion from a string literal to "char *" is deprecated

main.cu(168): warning #2464-D: conversion from a string literal to "char *" is deprecated

input size: 131072
pagefault number is 32772

```

What I learn in the assignment:

1. I learn how to implement LRU using array to count.
2. I learn the switch between virtual address and the real address.
3. I learn how CPU write things using virtual address.
4. I learn how CPU read things using virtual address.
5. I learn how to count page fault number.
6. I learn when should we use inverted page table instead of normal page table to save space.