Nachos Assignment
Project 3 - Memory Management

Qi Zheng, Lin.

Advisor: Farn, Wang.

Outline

- Memory Management
- Report
- Policy

Memory Management

About the virtual memory manager, swap, page fault handler and etc.

Goal of Memory Management

- Both the test cases require lots of memory.
  - /test/matmult.c
  - /test/sort.c
  - Larger than the physical memory
- Goal:
  - Run this two programs concurrently, and get the correct result.
  - Remember to specify the scheduling algorithm you used in the report.
  - **DO NOT** pass by simply modify the memory size in “machine”.

Hints

- File system – swap space.
- Maintain three tables to record the information.
  - PageTable
  - FrameTable
  - SwapTable
- Think how to “load” the page correctly and deal with the PageFaultException
- Design your own “Virtual Memory Manager”
- Some of you have figure out some point in project 1. Remember to write them down in report

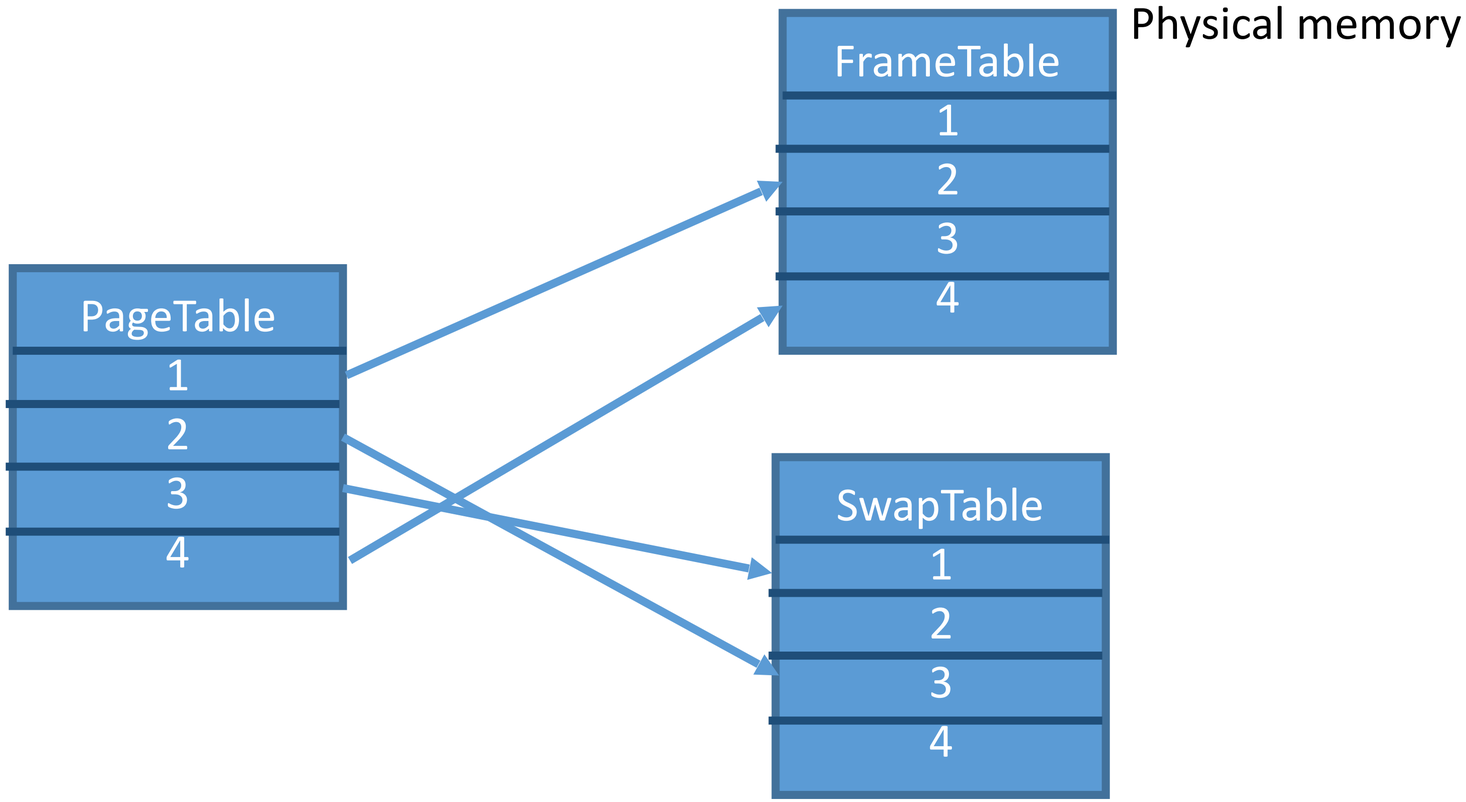
Swap Space

- swap = new SynchDisk in your kernel.
- Use the disk as the swap space.
- Access the virtual memory in the disk by…
  - kernel->swap->WriteSector
  - kernel->swap->ReadSector
- See “synchdisk.cc” and other files in /filesys for details!
- Read the header in those files first.

Maintain Three Tables

- PageTable
  - One page table per process.
  - Decide your virtual page number.
- FrameTable
  - Record every physical page’s information.
  - Each frame represent one physical page.
- SwapTable
  - Record every sector’s information in swap.
  - The number of entries in SwapTable is the same as the swap sectors.
  - Each entry represent one frame in the disk.

Maintain Three Tables Cont.



PageTable

- For each entry in PageTable
- TranslationEntry {

```
  unsigned int virtualPage;  //virtual memory page
  unsigned int physicalPage; //if in physical memory
  bool valid;                //if in physical memory
  bool use;                  //been used(read or write)
  bool dirty;                //been modified
};
```

FrameTable

- For each entry in FrameTable
- FrameInfoEntry {

```
    Bool valid;             //if being used
    Bool lock;
    AddrSpace *addrSpace; //which process is using this page
    Unsigned int vpn;       //which virtual page of the process is stored in
                            //this page
};
```

SwapTable

- For each entry in SwapTable
- ```
  FrameInfoEntry {          //exactly same as FrameTable
      Bool valid;           //if being used
      Bool lock;
      AddrSpace *addrSpace; //which process is using this page
      Unsigned int vpn;     //which virtual page of the process is stored in
                            //this page
  };
  ```
  ```

Addrspace::Load

- Note: Load 1 page a time.
- First ask for 1 page, if the FrameTable is full, select 1 frame and put it into SwapTable.
  - Design your own page replacement method to get the frame.
  - Specify the method in your report.
- Mapping virtual address to physical address.
- Invoke “executable->ReadAt(&(kernel->machine->mainMemory[physical address]), sizeToLoadNow, inFileAddr)”

Address Mapping

- Map the Virtual Address to Physical Address.
- Like the way you used in project 1.
- Physical Address =

  pageTable[(virtual address / PageSize)].physicalPage * PageSize + (virtual address % PageSize)

Maybe Create a New Class “Memory Manager”

- Class MemoryManager {

```
Int TransAddr(AddrSpace *space, int virtAddr);
  //return phyAddr (translated from virtAddr)
Bool AcquirePage(AddrSpace *space, int vpn);
  //ask a page (frame) for vpn
Bool ReleasePage(AddrSpace *space, int vpn);
  //free a page
Void PageFaultHandler();
  //will be called when manager want to swap a page from SwapTable
  //to FrameTable and the FrameTable is full.
};
```

Page-Fault Handler

- Put the pages in SwapTable into FrameTable.
- When all physical memory frames are occupied, design a page replacement method to get a frame.
  - Remember to specify it in your report!
- You can do this work in your own way.

Some File May Be Useful

- For the disk usage details, see:
  - /filesys/synchdisk.h
  - /filesys/synchdisk.cc
- Other files in /filesys (Optional).
- For the swap space initialization:
  - /userprog/userkernel.h
  - /userprog/userkernel.cc
- For the table maintaining, see:
  - /machine/machine.h and /machine/machine.cc
  - /machine/translate.h and /machine/translate.cc

Some File May Be Useful Cont

- For the table maintaining, see:
  - /machine/machine.h and /machine/machine.cc
  - /machine/translate.h and /machine/translate.cc
- For the loading of pages.
  - userprog/addrspace.h
  - userprog/addrspace.cc
- Always **see the header and comments** first.
- Based on your implementation, there might be different files that your need to see and modify.

Report & Policy

Report contents, grading policy

Report

- Report
  - Motivation and the problem analysis
  - What’s your plan to deal with the problem (high-level)
  - You can including some important code segments and comments
  - Experiment result and some discussion
  - Tell me **why and how** it happened
  - Remember there are two parts in project3
- Please saved as [Student ID]_report.pdf
  - E.g. r04921119_report.pdf

Code and Report

- Upload to CEIBA
  - Do not mail me the homework please
- Source code and report BOTH
  - create a folder and follow the structure below

    /r04921119_Nachos3

    |___ /nachos-4.0

    |___ r04921119_report.pdf
  - tar zcvf r04921119_Nachos2.tar.gz ./r04921119_Nachos2

Policy

- Nachos source code: (40%)
- Report: (60%)
  - Important !!! Tell as detail as you can
  - **why you choose, what you do, and why it works**
- Bonus
  - The replacement method you design.
  - Extra observation or modification on nachos.
  - Tell me how to switch and show in report