DATA PATH IN COMPUTER SYSTEMS

YUZHE TANG

CF: OSPP-11.4

PAGE FAULT HANDLING (3.4)

DISK READ

CALLER OF DISK READ

- syscall `read`
  0. os pauses caller process of `read()` to a wait queue
  1. os calls the code procedure of **disk read**
  2. os's interrupt handler then copies the data from kernel's buffer into the process's (virtual) address space.
  3. remove `p1` from wait queue; when `p1` gets scheduled next time, it's returned from `read()`
- page-fault handler
  0. os pauses caller process of `read()` to a wait queue
  1. os calls the code procedure of **disk read**
  2. os update page-table and return page-fault handler.

DISK READ (OS PERSPECTIVE)

- d1. mmio: os uses memory-mapped IO to:
  - tell the disk to read the requested data
  - to set up DMA
- d2. DMA: disk then transfer the data to its internal memory before DMAing it to main memory (kernel's buffer).
- d3. notify DMA completion.

1. MMIO MEMORY-MAPPED IO

1. IO bus:
   - one end connected to control registers in device (disk)
     - device control registers used to transmit data/commands to device
   - another end connected to system's memory bus
2. mmio: map device reg to memory addr space, allowing CPU to access it like regular memory blocks
   - e.g. `MOV rax, 0xC0003000`: move value of `rax` to disk control reg.
   - versus port-mapped io: pmio uses dedicated instructions `IN/OUT` (x86), now lessly used.

- physical address space is interpreted by hardware; diff. regions assigned to diff. devices.
  - in 32-bit addr. space, a little over 2GB are regions assigned to IO devices.

```
                VAddr Space     Page table      PAddr Space
              +-------------+ +-------------+  +----------+ 0X0000000
              |  kernel     | |    |        |  |          |
           +  +-------------+ +-------------+  |  DRAM    |
           |  |  .text      | |    |        |  |          |
fixed size |  +-------------+ +-------------+  +----------+ 0X7FFFF
           |  |  data       | |    |        |  |          |
           +  +-------------+ +-------------+  +----------+ 0XC00010
           |  |  heap       | |    |        |  |AUDIO CTRL|
dyn size   |  +-------------+ +-------------+  +----------+ 0XC00020
           |  +-------------+ |    |        |  |KEYBOARD C|
           |  |  stack      | +-------------+  +----------+ 0XC00030
           +  +-------------+                  |DISK CTRL |
                                               +----------+
```

D2.DMA

- disk io is suuuuuper slow, can't afford to go to CPU for every word transfer
  - instead, disk io transfer words in batch, and w.o. CPU to intervene
- DMA transfers data block from disk's internal memory to system's main memory (target page)
- DMA requires os to "pin" target pages in physical memory; can't be swapped out (e.g. by paging)

D3. INTERRUPT VS POLLING

- when DMA is done, need a mechanism to notify os:
    - "hey CPU! your requested data is now ready in kenerl buffer"
- CPU polling the control reg is slow (causing bus IO!)
- raising an interrupt is faster