Module 7063 - Operating Systems - FS-2010
Exercise #12: Virtual Memory Management

Problems from OSC

Problem 1 (OSC 9.2)
Discuss the hardware support required to support demand paging. When do page faults occur ? Describe the actions taken by the operating system when a page fault occurs.

Answer
For every memory access operation, the page table needs to be consulted to check whether the corresponding page is reisdent or not and whether the program has the read or write privileges for accessing the page. These checks have to be performed in hardware. A TLB normally serves as a cache and improves the performance of the lookup operation.
Actions for a page fault: see Figure 8.11 of the book.

Problem 2 (OSC 9.4)
A certain computer provides its users with a virtual-memory space of 2**32 bytes. The computer has 2**18 bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

Answer
The virtual address in binary form is

```
0001 0001 0001 0010 0011 0100 0101 0110
```

Since the page size is 2**12 (four kilo bytes), the page table has 2**20 elements (slots). Therefore, the low-order 12 bits "0100 0101 0110" are used as displacement into the page, while the remaining 20 bits "0001 0001 0001 0010 0011" are used as the displacement into the page table.

Problem 3
Which of the following programming techniques and structures are "good" for a demand paging environment. Which are "not good" ?
Explain your answers.

- Stack
- Hashed Symbol Table
- Sequential Search
- Binary Search
- Pure Code
- Vector Operations
- Indirections

Answer

- Stack -- good, stack operations are local
- Hashed Symbol Table -- not good, operations are not local
- Sequential Search -- good
- Binary Search -- not good, unless the table fits in few pages
- Pure Code -- good, sequential access
- Vector Operations -- good, sequential access
- Indirections -- not good, contains jumps

Problem 4
Suppose we have a demand paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty frame is available or if a replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds.
Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page fault rate for an effective access time of no more than 200 nanoseconds ?

Answer

```
0.2 x 10**-6 sec >= (1-p) x 0.1 x 10**-6 sec + 0.3 x p x 8 x 10 ** -3 sec

                  + 0.7 x p x 20 x 10 **-3 sec

0.1 >= p (-0.1 + 2400 + 14000)

p <= 0.000006
```

Problem 5 (OSC 9.10)
Consider a demand paging system with the following time measured utilizations:

```
CPU:              20 %
Paging Disk:      97,7 %
Other I/O Devices: 5 %
```

Which of the following will probably improve CPU utilitzation ? Explain your answers.

1. Install a faster CPU.
2. Install a bigger paging disk.
3. Increase the number of processes
4. Decrease the number of processes
5. Install more main memory.
6. Install a faster paging disk, or multiple controllers with multiple hard disks.
7. Add prepaging to the page fetch algorithms.
8. Increase the page size.

Answer

1. Install a faster CPU -- No, CPU is waiting most of the time
2. Install a bigger paging disk -- No, it is not the problem
3. Increase the number of processes -- Never, will increases thrashing
4. Decrease the number of processes -- Best Idea
5. Install more main memory -- Why not? can hold more pages in memory and thus, obtain less page faults

6. Install a faster paging disk, or multiple controllers with multiple hard disk -- Yes, as the disk is the bottleneck, the CPU gets data more quickly.
7. Add prepaging to the page fetch algorithms -- Could help, above all, if programs follow the locality principle.
8. Increase the page size -- Will reduce the number of page faults if programs follow the locality principle. If not, it could result in higher paging activity because fewer pages can be kept in main memory and more data needs to be transferred per page fault.

Problem 6
Consider the two-dimensional array A:

```
int A [100] [100];
```

where A[0][0] is at location 200, in a paged memory system with pages of size 200. A small process is in page zero (locations 0..199) for manipulating the matrix; thus, every instruction fetch will be grom page zero.
For three page frames, how many page faults are generated by the following array initialization loops, using LRU replacement, and assuming page frame 1 has the process in it, and the two others are initially emtpy:

```
Solution a:

for (int j = 0; j < 100; j++)
    for (int i = 0; i < 100; i++)
        A[i][j] = 0;


Solution b:

for (int i = 0; i < 100; i++)
    for (int j = 0; j < 100; j++)
        A[i][j] = 0;
```

Answer
The array is stored row wise (row-major), that is, the first data page contains the elements A[1,1],A[1,2],....,A[1,100],A[2,1],A[2,2],....,A[2,100], the second page contains A[3,1],A[3,2],....,A[3,100],A[4,1],A[4,2],....,A[4,100], and so on.

```
Solution a:

The page reference string is

             0,1,0,2,0,3,0,4,....,0,49,0,50,0,1,0,2,0,3,.....0,50,.....

There will be 50 x 100 = 5000 page faults.

Solution b:

The page reference string is
```

```
                0,1,0,2,0,3,0,4,....,0,49,0,50

    There will be 50 x 1 = 50 page faults.
```

Problem 7
Consider the following page reference string:

```
1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6
```

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames ?
Remember that all frames are initially empty, so your first unique pages will all cost one page fault each.

- LRU replacement
- FIFO replacement
- Optimal replacement

Consider the modified page reference string:

```
1112234444221115556222211122233777766333211122233366
```

How many page faults would occur for the working set algorithm if we use a window of (a) 6 references and (b) 10 references ?

Answer

Number of frames	LRU	FIFO	OPT
1	20	20	20
2	18	18	15
3	15	16	11
4	10	14	8
5	8	10	7
6	7	10	7
7	7	7	7

Working Set Algorithm

Window = 6

```
1112234444221115556222211122233777766333211122233366

1  1 11 2  21  1 1112  11    1 2  3363  211  1 1 22
   2 22 3  42  2 5525  22    2 3  7676  322  2 2 33
     33 4   4  5  656  6     3 7   7 7  633    3  6
      4            6                      6

*  * **     *  *  **   *     * *   * *  **     *  *
```

17 page faults

Windows = 10

```
1112234444221115556222211122233777766333211122233366
```

```
1  1 11        1  11       111 1   2  3 21  1 1   1
   2 22        2  22       222 2   3  6 32  2 2   2
     33        4  45       6 3 3   6  7 63  3 3   3
      4        5  56           7   7    76  6     6
                  6                      7

*  * **        *  *          * *   *    **        *
```

12 page faults

Problem 8 (OSC 9.14)
Consider a demand paging system with a paging disk that has an average disk access time and transfer time of 20 milliseconds. Addresses are translated through a page table in main memory, with an access time of one microsecond per memory access. Thus, each memory reference through the page table takes two accesses. To improve this time, we have added an associative memory that reduces access time to one memory reference, if the page table entry is in the associative memory.
Assume that 80 percent of the accesses are in the associative memory, and that of the remaining, 10 percent cause page faults. What is the effective memory access time ?

Answer

```
effective access time =
   0.8 1[us] +  (associative hit ok)
   0.2 (0.9 (1[us]  + 1[us]) (associative miss)
        0.1 (20[ms] + 1[us] + 1 [us]) (page transfer,memory access)
                      = 0.4 [ms]
```

Problem 9: Page Replacement Simulator (Programming Problem)
Write a program that implements the FIFO and LRU page replacement algorithms. presented in Chapter 9 of the text book.
Sample Solution. The files `FIFO.java` and `LRU.java` contain sample solutions for the classes to write.

Problem 10: Memory Mapped File with Linux
To work with a file you use the system calls `open(), read(), write(),` and `close()`. Virtual memory techniques allows you to map the file into the pages in the virtual address space of a process (memory mapping). If you access a page the first time it will be loaded (demand paging). Note that write operation to the file mapped in memory are not necessarily immediate writes to the file on disk.
To map a file in memory in Linux you use the `mmap()` system call.
Write a program that opens a text file that simply contains the following text:
`The quick brown fox jumps over the lazy dog.`
Establish two mappings at different addresses with different permissions. First display the file using the first mapping, Second modify the file using the second mapping.
Use the `getchar()` function to stop execution of the program so that you can observe the address mappings of the process in another window. The mappings can be found in the pseudo file `/proc/<PID>/maps/.`

Answer

There is a C program file in mmapfile1.c.

Problem 11: Copy a file using Memory Mapping
Map two files in two portions of the address space of a process. Then use the `memcpy()` utility to copy the first file in the second one. Again, use the `getchar()` function to stop execution of the program so that you can observe the files and their address mappings in the process in another window. The mappings can be found in the pseudo file `/proc/<PID>/maps/`.

Answer

There is a C program file in copywithmmap.c.

Problem 12: Memory Mapped File with Java
In Figure 9.25 of Section 9.2.7 in the text book there is an example of how to use the Java API for memory mapping files. Study the code and read the explanations given in the text. Notice that in the example you use a read only mapping. It means that the file must exist.
Modify the program to first create a large file by using the memory mapping technique. Second, write a program to compare the performance of sequential write operations with memory mapped write operations and compare the performance of sequential read operations with memory mapped read operations.

Answer

There is java program in MappedIoPerformance1.java that runs tests to compare stream I/O with memory mapped I/O and random access read operations.

Franz Meyer
Last modified: Mon May/25/2010