CSE 121: Operating Systems - Architecture and Implementation
Homework 2
Fall 2003

Due: Thursday, November 06

1 Journaling and Soft Updates [10 points]

Imagine a news server that needs to store new articles and remove old ones. These operations impose a lot of meta-data updates on the file system. We will analyze the performance of Soft Updates and Journaling FFS under this type of workload. Suppose the machine has a 512 MB file cache. Further, assume that:

- an inode is 128 bytes, indirect and directory blocks are 8 KB
- the journal is sufficiently large (6 GB)
- don't worry about the periodic checkpoints for either file system
- both file systems are based on FFS with 8 KB blocks.

Given a benchmark that does the following:

- traverses 100 different directories, creating a new 8 KB file in each
- removes an old file from each of the the 100 directories

There is no exact answer to this question. It is meant to illustrate larger properties of both file systems, so don't worry about the exact paging properties of the system and the exact numbers, just write down your (reasonable) assumptions and the steps to your answers in detail.

a) For each of the Journaling File System with a synchronous log, a Journaling File System with an asynchronous log, and Soft Updates, how many disk writes will occur?

b) Now suppose the News server deals with larger files. Repeat the benchmark described above assuming files of size 20 MB. Also assume that each create and delete is interleaved. For each of the Journaling File System and Soft Updates, how many times will a block need to be written to disk on the series of creates? Deletes?

2 Rio File System Cache [10 points]

a) The performance of Soft Updates decreases when the file cache fills up. In particular, it must evict dirty buffer cache pages with dependencies, forcing roll backs. The can result in synchronous writes and even an increase in the number of writes to disk. Given an infinite-sized file cache (i.e., all files and their associated meta-data could fit in memory), where writes (both data and meta-data) will always be guaranteed to perform asynchronously, would there be any motivation for using a scheme like the *Rio File Cache*? Explain why or why not.

b) A memory-based file system, sometimes known as a RAM disk, offers the low latency of memory accesses, but provides the interface of a disk. Assuming battery-backed memory (e.g., NVRAM or UPS), and accesses to the memory-based file system had to go through the same interface as disk accesses, would this file system provide the same guarantees as *Rio*?

3 Virtual Memory and Page Replacement [10 points]

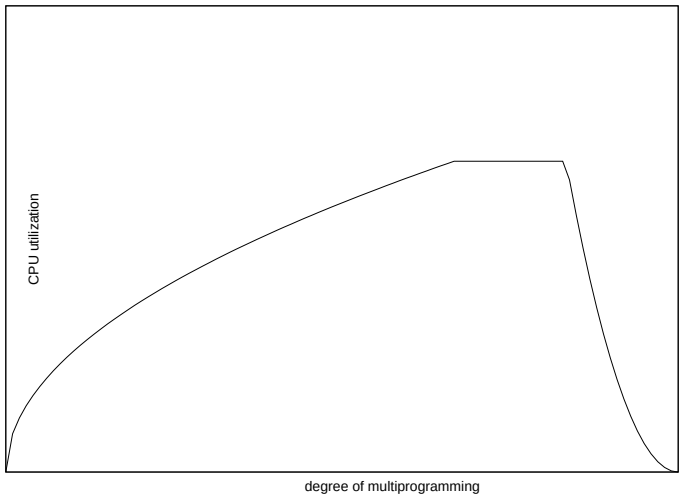

Figure 1: CPU utilization refers to *effective* CPU utilization.

a) Explain the shape of the graph in Figure 1. Be specific in explaining what is happening. Assume CPU Utilization refers to *effective* CPU utilization, which means the execution of program instructions.

b) It is desirable for an Operating System to minimize the page-fault rate of processes in the system. FIFO and LRU are two different types of global page replacement policies. FIFO replaces the oldest page in the system, and LRU replaces the page that has least recently been used. Suppose we have 3 page frames, F1, F2, and F3, and four virtual pages, A, B, C, D. Show the sequence of page faults for the reference string A B C A D C B D A C.‘

FIFO: __ page faults

physical frames	A	B	C	A	D	C	B	D	A	C
F1										
F2										
F3										

LRU: __ page faults

physical frames	A	B	C	A	D	C	B	D	A	C
F1										
F2										
F3										

c) Indicate whether any of the following *always* reduce (or at least never increase) the page-fault rate (support your answer):

(a) Increasing memory block size.

(b) Choice of page-replacement policy.

(c) Increasing number of page frames in system.

4 Interprocess Communication [10 points]

a) Suppose somebody has written an application where processes communicate using IPC. Unfortunately the program performs poorly when dealing with large data transfers. Would having the processes communicate through using shared memory improve the performance? Why or why not?

b) Suppose the signal handler below was written to catch SIGSTP. What is the problem with it?

```
catchsigstp()
{
            //do some preparation before stopping.
            signal(SIGSTP, SIG_DFL);
            kill(my_pid, SIGSTP);
            signal(SIGSTP, catchsigstp);
}
```

c) Cooperating processes often need to share resources. Access to these resources need to be guarded and fair. Because these resources can be accessed concurrently, we need some sort of synchronization mechanism to ensure that resources can be only used by one process (or other entity) at a time.

Below are two separate algorithms designed to facilitate resource access between two processes. Assume that two processes, *foo* and *bar*, both run the same algorithm. Indicate whether each algorithm works; in other words, whether they ensure both safety and liveness.

```
// foo is current process; bar is the other process
```

```
Algorithm 1
for (;;)
{
    while (turn != foo);
    //enter critical section
    turn = bar;
}

Algorithm 2
for (;;)
{
    interested[foo] = yes;
    while (interested[bar] == yes);
    // enter critical section
    interested[foo] = no;
}
```