CS 111 Midterm Exam

Yunjing Zheng

TOTAL POINTS

97 / 100

QUESTION 1

1 RAM speedup 9 / 10

- **0 pts** Correct
- **1 pts** It's missing pages, not missing page table entries.
- **1 pts** The main issue is paging, not swapping
- ✓ **- 1 pts** **Working set defined as the pages you need, not the page frames you are given**
- **1 pts** TLB is special hardware, not contents of RAM.
- **2 pts** Fragmentation is not the issue here, since paging causes little or no fragmentation.
- **2 pts** It's not cache hits here, it's whether you get a lot of page faults..
- **1 pts** Why does more memory make scheduling easier?
- **2 pts** Increasing amount of RAM does not increase TLB size.
- **3 pts** No necessary overhead for having large quantities of RAM.
- **3 pts** Adding CPU speed will not help with thrashing.
- **3 pts** Why does more RAM facilitate the working set better? What are the costs involved?
- **2 pts** Segmentation faults not related to amount of RAM.
- **2 pts** Lacking details of why costs are lower.
- **4 pts** Not so much that more applications can be stored in memory, but that more application pages can be, which is different.

QUESTION 2

2 API/ABI 10 / 10

- ✓ **- 0 pts** **Correct**
- **3 pts** No discussion of API X/ABI Z case
- **3 pts** No discussion of API W/ABI Z case
- **1 pts** Compiling on different machines with same API should work.
- **3 pts** If API is different, you cannot compile the program. You must rewrite it.
- **2 pts** No need to recompile if same ABI.
- **3 pts** If API the same and ABI different, must recompile.
- **1 pts** If machine has API X and ABI Z, then X can compile to Z.
- **1 pts** You generally can't install a new ABI without installing a new OS.
- **1 pts** If you rewrite to API W, ABI Y is irrelevant, since compilation will take care of differences.
- **2 pts** You can use a standard compiler for the API X/ABI Y -> API X/ABI Z case. No interpreter needed.
- **2 pts** How to account for binary-level compatibility?
- **2 pts** What issues for third case? How to resolve them?
- **2 pts** Same ABI implies compatible OS and hardware.
- **2 pts** The question states that the third and fourth machines DO support ABI Z.
- **4 pts** Same ABI implies no need to recompile.
- **10 pts** this answer has nothing to do with the question
- **4 pts** No discussion of 2nd machine with API X/ ABI Y case.
- **3 pts** You can't change a machine's ABI without changing its OS, typically. For this case, must rewrite program.
- **3 pts** Just recompile for the API X/ ABI Z case.
- **0 pts** Click here to replace this description.

QUESTION 3

3 Timer interrupts 10 / 10

✓ **- 0 pts Correct**

- 1 pts Unclear why timer interrupts make concurrency problems easier.

- 5 pts General ability to do preemptive scheduling is major reason.

- 3 pts Not really about priority boosting for MLFQ. More about general preemptive scheduling.

- 5 pts Not just about real time scheduling. More about general preemptive scheduling.

- 2 pts Major reason OS needs to regain control is to perform preemptive scheduling.

- 2 pts TImer interrupts do not help with deadlocks.

- 5 pts Why is it important for the OS to take control away from a process?

QUESTION 4

4 Limited direct execution 10 / 10

✓ **- 0 pts Correct**

- 5 pts Some form of direct execution is required for performance reasons.

- 1 pts Limits required for other reasons than taking over CPU

- 4 pts Less about virtualization than about providing good performance.

- 5 pts Direct execution is not a multiprocessor issue. A performance issue even on a single processor.

- 4 pts Limitation is to prevent processes from running dangerous instructions, not particularly about context switching or timer interrupts.

- 4 pts Direct execution has nothing to do with disks. It's about raw CPU performance.

- 4 pts Limitations on direct execution are not directly related to memory issues.

- 5 pts The limitation here is not on the OS, but on the application, to prevent it from running dangerous instructions.

- 1 pts Primary limitation is on running certain instructions.

QUESTION 5

5 Real time preemptive schedulingSe 10 / 10

✓ **- 0 pts Correct**

- 5 pts Soft real time systems are not time-sliced based.

- 5 pts Meeting deadlines is the primary issue for soft real time systems.

- 1 pts MLPQ not used for real time, other than perhaps having one special real time queue.

- 4 pts Without preemption, we often can't meet deadlines in a soft real time system. No way to ensure a new event with a near deadline gets scheduled.

- 3 pts Soft real time systems are not usually tightly prescheduled, but instead rely on event deadlines for scheduling.

- 4 pts For real time system preemption, deadlines are the criteria for preempting things. Near deadline events aren't preempted by far deadline events.

- 4 pts Insufficient explanation, not hitting key points.

- 2 pts Fairness and general response time are not the main issue in soft real time systems.

- 7 pts Soft real time isn't hard real time.

QUESTION 6

6 Federation frameworks 10 / 10

✓ **- 0 pts Correct**

- 10 pts No answer.

- 10 pts Completely wrong.

- 3 pts More details required.

- 5 pts Not really. It's not about OS portability, it's about providing a common interface to a service offered in many different forms.

- 3 pts Used not just for new devices, but for whatever is running in your system. Also used for some software modules, like file systems.

- 5 pts Backward compatibility is provided by a federation framework, but that isn't the point. It provides a uniform interface to many different, but similar things providing a type of service.

- 7 pts Used to provide a uniform interface to many

different, but similar things providing a type of service.

- 1 pts Your definition not very clear.

- 8 pts A few details are correct, but basically your answer is vague and misses important points.

- 0 pts Click here to replace this description.

QUESTION 7

7 TLB and ASIDs 10 / 10

✓ **- 0 pts Correct**

- 9 pts Used to identify whose address space it is. Particularly valuable for context switches.

- 6 pts The ASID field does indeed indicate the process identity, but the main reason is to avoid having to flush the TLB.

- 1 pts TLBs are not per process. They're shared hardware.

- 5 pts Why?

- 3 pts Why is this useful?

- 2 pts Why not just flush TLB on context switch?

- 10 pts No answer here.

QUESTION 8

8 Shared memory IPC 10 / 10

✓ **- 0 pts Correct**

- 3 pts Speed is a main advantage

- 1 pts Only applications that work together to share memory have access to it.

- 3 pts Difficulty in synchronization is main disadvantage.

- 2 pts Doesn't necessarily save space, since data to be moved between processes lives somewhere, often multiple places, in any mechanism.

- 4 pts Incorrect understanding of when and how shared memory can be updated.

- 5 pts No disadvantages discussed.

- 2 pts Not a high overhead mechanism.

- 2 pts What kind of cost?

- 1 pts Actually quite easy to implement in OS.

- 2 pts Bus is not the issue. Being on the same machine is.

QUESTION 9

9 Semaphore counts 9 / 10

- 0 pts Correct

- 4 pts Count should be 3.

- 1 pts Semaphore operations might put the requesting thread to sleep, but never another thread.

- 2 pts The question asked about access to a resource, not producer/consumer.

✓ **- 1 pts Post operation will wake a waiting thread, if one exists.**

- 1 pts Only one waiting thread awakened.

- 2 pts Wait on one doesn't block.

- 2 pts Wait on 0 blocks.

- 2 pts This solution might work, but isn't how semaphore behavior here is defined.

- 2 pts Wait on zero decrements to -1. Normally blocks, shouldn't for your solution.

- 2 pts Post operation increments count.

- 3 pts Initializing count to 2 will block third requester.

- 1 pts Wait and post are typically implemented as atomic operations, so no concurrency issue on the post operation.

- 2 pts Your version of post won't work with your initialization of count.

- 4 pts You didn't specify exactly what happens on the wait operations. Semaphore operations are highly specific and being vague leads to trouble, which may be why you set the count to 2 initially.

QUESTION 10

10 Synchronization example 9 / 10

- 0 pts Correct

- 1 pts Return in bitMapGet not in critical section.

- 1 pts for() in bitMapGet not in critical section

- 3 pts Describes update, but doesn't show how to do it.

- 2 pts Critical section in bitMapGet starts when mask is set.

✓ **- 1 pts Critical section in bitMapFree doesn't include first two lines.**

- 3 pts No critical section in getblock or freeblock.

Serious overlocking. Lock only in bitMapGet and bitMapFree.

- 2 pts Missing critical section in bitMapFree

- 3 pts You didn't use mutual exclusion on anything.

- 0 pts Not quite right (think about that continue()), but close enough.

- 0 pts Didn't perfectly identify the critical section, but using an atomic instruction is indeed better here.

- 1 pts In bitMapGet, need to hold lock until map is set.

- 2 pts Are you locking anywhere except in bitMapGet?

- 3 pts Missing critical section in bitMapGet

- 1 pts Won't execute the lock release in bitMapGet if it returns first.

- 1 pts Need to hold lock in bitMapGet throughout.

- 0 pts Actually, the critical section in bitMapGet covers the code between those sections. But you locked the whole range anyway, so no points off.

- 2 pts No critical section in getblock.

- 1 pts Lines in between two marked sections in bitMapGet also critical.

Midterm Exam
CS 111, Principles of Operating Systems
Summer 2017

Name: Yunjing Zheng

Student ID Number: 304806663

This is a closed book, closed note test. Answer all questions.

Each question should be answered in 2-5 sentences. DO NOT simply write everything you remember about the topic of the question. Answer the question that was asked. Extraneous information not related to the answer to the question will not improve your grade and may make it difficult to determine if the pertinent part of your answer is correct. Confine your answers to the space directly below each question. Only text in this space will be graded. No question requires a longer answer than the space provided.

1. Adding more RAM to a typical computer will often make applications on the machine run faster, even though the CPU speed has not changed. Why?

If there is more memory, processes can have larger working sets. They will run into the issue of pages they need being on disk instead of in RAM less often. Pages from disk will need to be swapped with pages in RAM less. Since swapping to/from disk is slow, doing so less allows applications to run faster.

2. You write a program that compiles and runs properly on a machine with API X and ABI Y. What must you do to run that program on a different machine with API X and ABI Y? On a different machine with API X and ABI Z? On a machine with API W and ABI Z?

X & Y: If the API and ABI are the same, you can either copy the compiled program to the other machine or copy the source code and then compile it on the other machine to get the compiled program. Both ways work.

X & Z: Only the API is the same. This means you must copy the source code and compile on the machine to obtain the compiled program.

W & Z: You can't accomplish this easily. If the APIs are similar, you can try modifying the source code to satisfy this machine's API, and then compile it on this machine. If ABI is similar, you can try modifying the compiled program. Otherwise you cannot run the program.

3. Why are timer interrupts particularly important for a general purpose operating system?

Timer interrupts allow preemptive scheduling. This means that the operating system can run one program for a fixed amount of time, interrupt the program, and then run a different program. The benefits of this approach are that

a) Programs with bugs or greedy programs won't freeze the cpu by never passing control back to OS

b) Programs can be run round-robin style (in a multilevel scheduling) allowing short response times to I/O queue

4. Why do modern operating system use the limited direct execution model for processes? Why any form of direct execution? Why "limited," and limited in what ways?

Modern operating systems use limited direct execution as a way to balance security and speed.

The direct execution part allows programs to run mostly in user-level, which is much faster than running in kernel level because it eliminates much of the overhead from the kernel supervision.

The limited part makes sure programs don't break the OS, access another program's memory, take up all the CPU time, or do anything else it's not supposed to. A process's run time is limited by timer interrupts, described in the question above. Furthermore, a process is only allowed to do certain things directly. It cannot call privileged instructions. If it ever wants to perform I/O, open files, or do anything else that might be dangerous, it must make a system call. The process will go into kernel mode, and the kernel will judge whether the call is legal. If it is, the kernel will perform the operation and then return the process to user mode.

5. Should a soft real time system use a pre-emptive scheduling mechanism? Why?

Usually yes. A soft real time system is one in which it's highly preferable that events happen at exactly the right time (like when playing a massive multiplayer game, it's nice to shoot the enemy while he's still in front of you), but missing an event or running it at a slightly incorrect time won't result in system failure. In soft real time systems in which the OS knows each process well (knowing how long they'll run, which order to run them in, and assuming they're relatively short), preemptive scheduling is unnecessary and will only cause overhead that slows things down. But if the OS doesn't know the processes well and can't guarantee a set run time, preemptive scheduling is needed to ensure processes and run fairly and no particular process freezes the OS

6. What is the purpose of a federation framework in an operating system? In what circumstances is one likely to be used?

A federation framework sets a standard interface for a variety of programs of the same type. For example, it ensures that numerous different printers are all compatible with your computer because the device drivers can recognize and work with all of them. Federation frameworks are generally used when compatibility is a major concern and the programs/device drivers all have similar features and functionality. They are less useful when the programs each have unique features.

7. Why do some modern TLBs have an address space identifier (ASID) field in each entry?

In old TLBs, once a new process began all TLB entries of the old process became invalid. The TLB had to be flushed so that the invalid bit was set in each of the entries. This increased the time context switching cost, because all relevant VA had to be looked up in page tables after a context switch. Modern TLBs often have an ASID field so that new processes don't have to flush the TLB. They can simply check the ASID to see whether the entry is theirs or not theirs. This reduces context switching times.

8. What are the advantages of a shared memory IPC mechanism over other IPC mechanisms? What are its disadvantages?

Adv: Very fast and flexible. Since processes are writing and reading directly into their memory, it's over a thousand times faster than general network communication. Especially if they're communicating a large amount of data, shared memory is an effective way to quickly send it to another process. This is a good way to communicate to graphic drivers, for example.

Disadv: Complicated. The processes themselves must work out how they'll communicate. This is complicated especially because they are running asyncronously. If two processes write to an area at the same time, they'll accidentally overwrite each other. Only works on one machine. Of course, the two programs must be on the same machine to utilize the same physical address space.

9. If you use a semaphore to control access to a resource with three simultaneously usable copies, what do you initialize the semaphore's count to? Describe the effects of a wait() operation on the semaphore when the count is at 1. Describe the effects of a wait() operation when the count is at zero. Describe the effects of a post() operation when the count is at 2.

Count should be initialized to 3 for three copies.

A wait at count = 1 will decrement the count to 0

A wait at count = 0 will decrement the count to -1 and then put the thread to sleep. It'll only be woken by another thread who increments count back to 0.

A post when count is 2 will increment count to 3.

10. Study the following code, taken from heavily used parts of a multi-processor operating system, and circle critical section(s):

```
Insert → int spinlock=0;
int bitMapGet( unsigned char *map, int len ) {
        int byte, bit, mask;
        for( byte = 0; byte < len/8; byte++ ) { while(test_and_set(&spinlock,1)==1){}← Insert;
                if ((mask = map[byte]) == 0)
                        continue;
                for( bit = 0; bit < 8; bit++ ) {
                        if ((mask & (1 << bit)) == 0)
                                continue;
                        map[byte] = mask & ~(1 << bit);
        → spinlock=0; return( (8*byte) + bit );
Insert → }
        } → spinlock=0;

        return( -1 );
}

void bitMapFree( unsigned char *map, int entry ) {
        int bit, byte;
Insert → while(test_and_set(&spinlock,1)==1) {}
        byte = entry/8;
        bit = entry&7;
        map[byte] |= (1<<bit);
    } → spinlock=0;
...
int getblock(struct bsdfs *fs, int inode) {
...
        cylgrp = inode / fs->inodes_per_group;
        cgstruct = fs->cylgroups[cylgrp];
        blkno = bitMapGet( cgstruct->free_list, cgstruct->num_blocks );
        return (blkno);
}

void freeblock(struct bsdfs *fs, int block) {
...
        cylgrp = block / fs->blocks_per_group;
        cgstruct = fs->cylgroups[cylgrp];
        bitMapFree( cgstruct->free_list, block );
}
```

// A mutex is more efficient here
// but unfortunately I forgot
// the syntax for mutex
// locking and unlocking

Update the code in part (a) to show how you would eliminate the possibility of disastrous conflicts within the critical section(s) you identified. You may either indicate lines to insert above on this page, or write the entire revised code on the following blank page.