

CS 574
Midterm Exam
October 13, 2008

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- start each problem on a new sheet of paper
- write your Banner ID number, but not your name, on each sheet of paper you turn in

Also, please note the following:

- show your work whenever appropriate. There can be no partial credit unless you show how you derived your answers
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 2:20 to finish the exam. The questions are equally weighted.

1. A virtual machine environment typically provides virtualized versions of common devices, such as video cards and network interface cards (among others). Give an advantage of providing as simple as possible a virtualized device (for instance, providing a very generic video card even if the underlying platform has sophisticated 3D acceleration)? Give an advantage of providing a very complex virtualized device (for instance, providing a network interface card with many levels of the TCP/IP stack implemented on the card, even if the underlying interface card is a very basic NIC that requires nearly all interaction in hardware)? If it is known in advance that a guest OS will be running in a virtualized environment, why can we reduce the number of drivers shipped with the guest? Does this affect the number of drivers shipped with the host OS?

Note for future reference: the phrase “interaction in hardware” was a typo. It should have been “interaction in software”. This got corrected at the beginning of the hour.

- (a) *The main advantage to providing simple virtualized devices is simplicity in the VM software. If we just provide a simple interface, we can just interact with those parts of the real hardware. If we provide a more complex interface, we (1) need to write a driver that interacts with the entire device, and (2) need to emulate the capabilities of the complex device when only a simple device is actually present.*
 - (b) *The main advantage to providing a complex virtualized device is that it makes it possible to take full advantage of complex physical devices when they’re available, though (in general) it will need to emulate those functions when they aren’t. Note, though, that in this particular example it’s probably possible to just piggy back on the host OS to do the low-level NIC interaction, and provide a pretty complicated virtualized device without needing to do your own emulation.*
 - (c) *The guest OS only needs to ship with drivers for the virtualized hardware. Of course, this is really a special case of the more general rule that if you’re only going to have to support a small number of devices, you don’t need to ship drivers for any others.*
 - (d) *Doesn’t affect the host OS in the slightest. It still needs to support any hardware devices that might get plugged in.*
2. A fellow student comes to you with a great new idea for an MS project: journalling the LogFS filesystem, much like ext3 is a journal added to ext2. Should the student propose this idea to his or her advisor, or go back to looking for a project?

Back to the drawing board. logfs already does out-of-place updating, and makes modifications bottom-up. The end result is that it is already impossible to have the filesystem in an inconsistent state, so journalling won’t add anything (a few students were concerned about extra writes to FLASH; that would be a concern if there were any point to journalling in the first place).

3. Why is it generally a good idea for batch jobs to have a relatively long time quantum? I made the comment when presenting my solution to HW1 that it really didn't make much sense in that problem for the batch jobs to have a longer time quantum than the other tasks – why?

As a general rule of thumb, the longer a process's time quantum is the higher its cache and virtual memory hit rates get. In other words, until you have to block due to required interaction, you run more and more efficiently as you run longer. Also, of course, the scheduling itself takes CPU resources; running the scheduler less frequently leaves more for the real work. If interactivity isn't a concern (and it isn't for batch jobs, by definition), you want as long a quantum as possible.

It doesn't make much sense in that problem because the root scheduler time quantum is short. So, you don't get the benefits of the long quantum.

4. Would you expect software page coloring to be more helpful in reducing cache misses for a direct mapped cache, a four-way set-associative cache, or neither? Why is it possible software page coloring could cause more page faults (compared to a page replacement strategy that doesn't use page coloring)?

More good for direct-mapped – associativity allows the hardware flexibility in placing data, so the number of conflict misses goes down. Consequently, page coloring has less opportunity to make improvements.

Might cause more page faults because you've got less flexibility in the pages to choose for replacement. I'd be pretty surprised if there were a noticeable effect, though. One student based the answer on a pathological program that constantly changed access patterns in ways that interacted badly with the coloring strategy, which I accepted.