University of Wisconsin: Department of Computer Sciences

Operating Systems Qualifying Exam: Spring 2012

Instructions: There are six questions on this exam; answer all six questions.

Question 1: Interprocessor Communication and Scheduling

Suppose you want to build a system that incorporates both scheduler activations and lightweight remote procedure calls (LRPC).

(a) For scheduler activations, explain the information passed by the user-level scheduler to the kernel, and by the kernel to the user-level scheduler.

(b) On a system with MxN threads **without** scheduler activations, what happens when a user-level thread makes an LRPC?

(c) On an MxN thread system **with** scheduler activations, how should the user/kernel interface change to accommodate LRPC?

Question 2: Software Fault Isolation

Suppose you are building the next-generation web browser and want to make it secure, so you decide to run extensions and plugins using software fault isolation.

(a) Explain what faults software fault isolation can handle, and how?

(b) Software fault isolation, while relatively fast, is not free.

> (i) What are the major performance overheads from SFI? Be specific.
>
> (ii) How does the performance of SFI differ from using virtual memory for isolation, as in Nooks?

(c) In your system, programmers will be writing code knowing it will be executed with SFI. Explain what can be done during the code preparation process (coding, compiling, linking) to improve the performance of the code with SFI.

Question 3: Reliability and Synchronization

(a) There are different styles of signal-wait synchronization among threads, such as Hoare style and Mesa style. What is the difference between these two styles? Why have existing systems all implemented the Mesa style?

(b) The following program encounters non-deterministic crashes at run time. Please describe the root cause of this crash and explain how the root cause leads to a crash.

(c) Change the program source code to fix the problem, with minimal change to the semantics.

Systems such as Rx can help a program survive crashes by changing the execution environment, instead of changing the program source code itself.

(d) Without changing the program source code, can you change OS scheduler to make the crash disappear? What is the negative impact of your change?

(e) Without changing the program source code or the OS scheduler, can you change memory management to make the program less likely to crash? What is the negative impact of your change?

```
char*buffer;
int count = 0;
pthread_mutex_t  L = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t     C = PTHREAD_COND_INITIALIZER;

char produce(); //the detail of this function is irrelevant to this question

void main(){
   buffer = (char*)malloc(100*sizeof(char));
   if(!buffer) return;

   pthread_create(NULL,NULL,child,NULL);
   pthread_create(NULL,NULL,child,NULL);

   while(1){
      if (count > 0){
         pthread_mutex_lock(&L);
         printf(" %c ", buffer[--count]);
         pthread_cond_signal(&C);
         pthread_mutex_unlock(&L);
      }
   }
}//The program does not exit by itself

void child(){
   while(1){
      pthread_mutex_lock(&L);
      if (count==100)
         pthread_cond_wait(&C,&L);
      buffer[count++] = produce();
      pthread_mutex_unlock(&L);
   }
}
```

Question 4: Virtual Machines

In a virtual machine monitor (VMM), virtualizing memory beneath a guest operating system can be quite tricky. In this question, we will explore some of the difficulties of VMM memory virtualization.

(a) To virtualize memory underneath the OS, the VMM must keep track of "physical"-to-machine mappings, where the "physical" page numbers are what the OS thinks it is using, whereas the machine mappings are the actual page locations in hardware. What kind of structure does a VMM use to track these mappings? How big are these structures? Are they similar to any structures within the OS itself?

(b) When running on a system with a software-managed TLB (such as MIPS), the OS installs virtual-to-physical mappings directly into the TLB. However, when running on a VMM, the VMM must be involved upon a TLB miss. Describe what happens in a VMM-based system when a TLB miss takes place (you can assume that the page resides in physical memory). How does the virtualization affect performance, and what can be done about it?

(c) Some systems (e.g., x86) have a hardware-managed TLB; a register (e.g., cr3) points to the root of the page table for the currently-running process, and all TLB misses are directly serviced by the hardware. Describe how a VMM interacts with the hardware and OS to virtualize memory underneath the OS on such a hardware-managed TLB system.

Question 5: Network File System

Sun's Network File System (NFS) was one of the first client/server distributed file systems. In this question, we'll explore some aspects of the protocol and its impact on client-side and server-side issues.

(a) One critical aspect of server performance is responding to "Get Attribute" (GETATTR) requests; for most servers, a high fraction of all requests are GETATTRs. What aspect(s) of the NFS protocol induces this large number of GETATTR requests?

(b) High-performance NFS servers usually have large, battery-backed memories, which they use to improve performance. What aspect of the NFS protocol mandates that servers have such hardware? How much does it help performance?

Both clients and servers are usually provisioned with a great deal of memory. We'll now explore how to best utilize these memories for caching.

(c) Client caching behavior is critical for performance. Describe how NFS client caching improves the performance for both reads and writes of files.

(d) As client caches grow, the utility of server-side caching comes into question. What should be cached in server memory, and what shouldn't be? Explain how to best design the caching algorithms of the server so as to improve performance.

Question 6: Storage Mashup

The Google File System targets large-scale workloads requiring mostly sequential access. The HP AutoRAID system automatically balances data between capacity and performance tiers.

(a) Explain the design elements of GFS that optimize for mostly sequential access.

(b) Explain how AutoRAID determines (i) how much and (ii) which data should be stored using mirroring instead of RAID 5.

(c) Suppose you have a GFS cluster and upgrade a portion of the machines with new disks that are twice as fast as the existing disks. Explain how you would apply AutoRAID-style techniques to automatically figure out which data should be stored on these faster disks.