CS 6210 Spring 2023 Test 1

(120 min Canvas Quiz)

Table of Contents

OS Structure [35 points]

SPIN [11 points]

1.(2 points) (Answer True/False with justification) Every time there is a program discontinuity, there is border crossing in the SPIN operating system. “Border crossing” refers to going from one hardware address space to another.

False: Logical protection domains share the same hardware address space. Since we are not going from one hardware address space to another, there is no border crossing whenever program discontinuities occur.

2.(4 points) At the end of the day, SPIN is also a monolithic operating system. So, your friend argues that it offers no advantage compared to a traditional monolithic OS. Give two counterpoints to prove your friend wrong.

Answer:

- extensibility without having to take down the OS, i.e., creating new system services on the fly dynamically

- modularity for the system services using Modula-3's strong typing rules, exposing only well-defined interfaces and hiding the details of the implementation

(any other valid counter point can also be considered if they don't say the above two)

3. (3 points) Using specific examples of services provided by an OS, succinctly explain why it is a myth that an entire OS can be written in a strongly typed programming language like Modula-3.

Answer:

- Implementing system services such as a memory manager or dispatching a thread to run on a CPU, requires stepping outside the strongly typed protection model of Modula-3 to write into hardware resources such as CPU registers and device controllers.

(+1 if they mention need to step outside the protection model

+2 if they mention why such stepping outside the protection model is needed for the specific services they identify)

4. (2 points) Do you agree with the SPIN developers' decision to use Modula-3, instead of using a more widely known language such as C? Why or why not?

Modula-3 is a strongly typed language, it provides compile-time checks and run-time verification to ensure the safety of object accesses. This allows moving across protection domains within the same hardware address space. C pointers are not strongly-typed, you can type-cast them and interact with objects they point to without any restrictions.

+1 Modula-3 is strong typed/ C is not strongly typed

+1 Any explanation supporting the use of Modula-3 or against the use of C.

Exokernel [11 points]

1. (3 points)

What is the purpose of the processor environment (PE) data structure in Exokernel? Who populates the data structure? Give an example of how this data structure is used.

Answer:

(+1) Purpose: to handle program discontinuities

(+1) The library OS populates this data structure

(+1)

Upon a sys call by the currently executing process, Exokernel looks up the entry point for sys calls in the PE to make an upcall into the library OS;

OR

Upon an external interrupt destined for the OS that is currently scheduled, Exokernel looks up the interrupt context entry in the PE data structure to make an upcall to the library OS

2.

(4 points)

Consider the following situation with respect to Exokernel. A network packet arrives. How does Exokernel know the library OS that should receive this network packet?

Answer (the students have to be creative for answering this question):

- At boot time, Exokernel will assign a virtual NIC MAC address for each library OS (+2)

- The arriving network packet will have this virtual MAC address enabling Exokernel to direct it to the appropriate library OS (+2)

(above solution is very similar to how NAT works in home networks)

3. (2 points)

Exokernel switches from OS1 to OS2. The architecture does not support address-space tagging in the TLB. Therefore, the TLB must be flushed completely. The architecture treats TLB misses as a "page fault" that must be handled by the operating system. How does Exokernel ensure that OS2 becomes instantly productive even though the TLB has to be flushed while switching from OS1 to OS2?

Anwser:

(+1) S-TLB data structure for each library OS that contains the guaranteed mappings as specified in the PE data structure

(+1) S-TLB for OS2 loaded into the hardware TLB by Exokernel before transferring control to OS2

4. (2 points) (Answer True/False with justification)

When a library OS receives "revocation for a set of memory frames" from Exokernel, the library OS loses all the contents in those memory frames.

Answer:

False. The library OS is given an opportunity to "save" the contents (e.g., on disk) or "seed" Exokernel to do the saving on its behalf automatically.

Microkernel [9 points]

1.(2 points) An architecture does not support address-space tagging in the TLB. It does support segmentation. Your friend argues that implementing a microkernel-based OS on this architecture would involve frequent border crossing (i.e., hardware address space switching). Succinctly provide a counterargument to your friend's point of view.

Answer:

- small protection domains can be packed into the same hardware address space using segmentation

2. (2 points) (Answer True/False with justification)

L3 avoids both the implicit and explicit costs of crossing logical protection domains regardless of the memory footprint of the protection domain.

Answer:

False. A large protection domain that needs almost the entire hardware address space would suffer from both the implicit and explicit costs of protection domain switch regardless of the architectural features such as segment registers or address space tagging in the TLB.

3. (2 points) (Answer True/False with justification)

A monolithic kernel design is superior to L3 for implementing system services with large memory footprints.

Answer:

False. Even though a monolithic kernel may have all the system services packed into one hardware address space (as opposed to a L3 where each system service is in a different protection domain), the

implicit costs (due to cache pollution) will dominate due to the large memory footprint.

4. (3 points)

Consider two implementations of an OS. The code base for both the OSes are exactly the same when it comes to how the system services are implemented. One is implemented as a monolithic library OS on top of Exokernel. The second is implemented as a micro-kernel based OS on top of L3. Both execute on the same x86 hardware with support for segmentation and no support for address-space tagging.

Answer the following question:

A process makes a system call that results in traversing three system services to complete the call. All three system services have small memory footprints. Will the system call take about the same time in both OSes? Justify your answer.

Answer:

Yes. In the library OS, traversing the three system services will be like simple procedure calls. In the L3 implementation, while they will cross protection domains, they can all be packaged in the same hardware space using segmentation. So while there may be a small explicit cost for loading the segment registers for facilitating the cross-domain calls, the service time in each protection domain will be the same since the code base is the same. Thus overall the system call will take about the same time in both OSes.

OS Structures [4 points]

1. (4 points)

An OS uses non-preemptive shortest job first scheduling for the CPU. Using this exemplar system service, identify mechanisms you might need in the OS, and how the scheduling policy would use these mechanisms.

Answer:

Mechanisms:

- Saving CPU registers in PCB of currently running process
- Load the CPU registers from the PCB of a process chosen to run

- Gathering stats on the amount of time a process runs before making a system call

Policy:

- use the save CPU registers mechanism to save the volatile processor state into the PCB of the currently running process that has made a system call

- use the gathered stats for the currently runnable processes to pick a "winner" for running on the CPU that has the shortest execution time

- use the load CPU registers mechanism to dispatch the chosen winner

Virtualization [35 points]

Paravirtualization [14 points]

1.

a) [3 points] An application process starts up in a guest OS running on top of Xen. List the steps taken by the guest OS and Xen to set up the page table for this process. You can assume any reasonable hypervisor calls without worrying about the exact syntax. State your assumptions.

[+1] Guest OS allocates a physical page frame from its own reserve as the page table (PT) data structure for the newly created process
[+1] Guest OS registers this page frame with Xen as a page table using a hypercall.

[+1] Guest OS updates batched VPN to PPN mappings into this page table data structure via the hypervisor by using a single hypercall.

b) [4 points]

Give two reasons with justification in support of para virtualization.

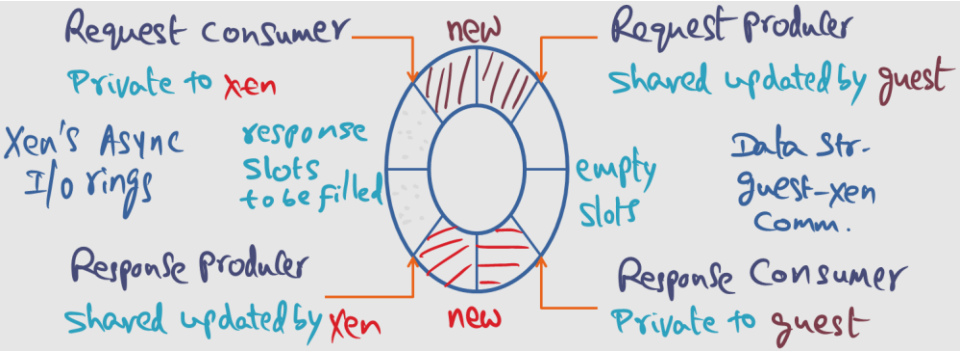
Answer:

- More opportunity for a guest OS to be in control. (+1)

For example, a guest OS may turn off notifications from the hypervisor for a period of time (e.g., when it is inside a critical section). (+1)

- More opportunity to innovate how I/O is handled. (+1)

For example, it would allow a guest OS to reduce memory buffer copying overhead during data transfer.



(b) [7 points]

Above picture shows the generic I/O ring data structure used in Xen to facilitate communication between the guest OS and Xen. Guest-OS places a request in the I/O ring using the "Request Producer" pointer. Xen places a response in the I/O ring using the "Response Producer" pointer.

(i) (1 point) How does this data structure ensure memory isolation properties across guest OSes running on top of Xen?

Answer: I/O ring is not shared across guest OSes. It is private between a given guest OS and Xen.

(ii) (1 point) What is the condition that tells the guest OS that it cannot enqueue any more requests into the I/O ring?

Answer: The "request producer" pointer equals the "response consumer" pointer.

(iii) (1 point) What is the condition that tells Xen that it cannot enqueue any more responses into the I/O ring?

Answer: This will never happen since Xen puts its response in the same slot that it dequeued a request from the guest OS.

(iv) (2 points) You are implementing the network communication in a guest OS. You choose to use two I/O ring data structures, one for transmit and one for receive. What will you do to ensure "zero copy semantics" on the transmit side?

Answer:

- (+1) In the descriptor that is enqueued in the I/O ring, place a pointer to the memory buffer in the guest OS.

- (+1) Ensuring that the page is pinned until the packet has been successfully transmitted.

(v) (2 points) You are implementing the network communication in a guest OS. You choose to use two I/O ring data structures, one for transmit and one for receive. What will you do to ensure "zero copy semantics" on the receive side?

Answer:

- (+1) Upon packet reception, Xen uses one of its memory buffers (i.e., page frame) to receive the packet.

- (+1) Xen exchanges the memory frame containing the packet with free page frame from the guest OS.

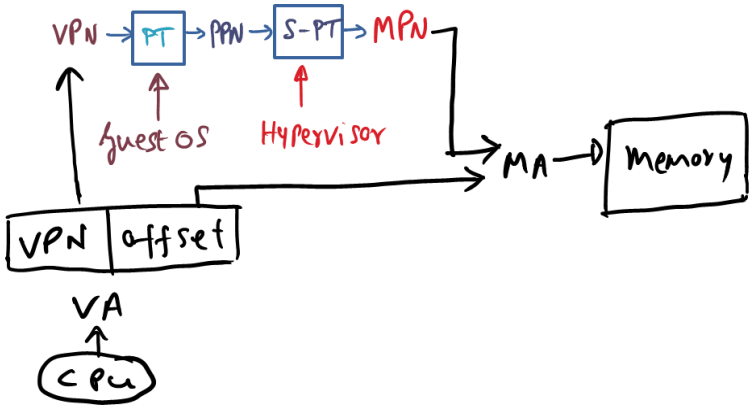
-

Full virtualization [7 points] [@Hemang]

a) [2 points] Consider a fully virtualized operating system. The OS maintains a page table (PT) data structure for each process that it launches to provide memory protection. A process launched by this OS is currently being executed.
The processor architecture uses a hardware page table for address translation. We will call the page frames in the DRAM (i.e., the real memory in the computer system) as "machine pages".
(Answer True/False with justification) (No credit without justification) The page table data structure of the currently executing process is the same as the hardware page table.

False, the page table data structure for the currently executing process is present in the guest OS. The hardware page table is present in the processor and contains the mappings from VPN (of the process) to MPN.
[+1] for false
[+1] for justification

b) [3 points]



Kira took CS 6210 while at GT. She graduated and started work with a cloud start-up. She is in the hypervisor development team that is responsible for designing the memory virtualization for hosting "fully virtualized" OS on top of the hypervisor. She could only remember the above diagram from the CS 6210 course that shows the steps by which a process's virtual address is translated to machine address.

(i) (1 point) What is the pitfall if Kira implements the figure as is for address translation?

Answer: The page table (PT) for the process is a data structure inside the fully virtualized guest OS not visible to the processor architecture. So every memory access by the processor is no longer happening in hardware.

(ii) (2 points) Her colleague Hemang offers a clever solution that allows the processor to directly translate the VPN to MPN. What is his solution?

Answer:
- The S-PT which is a hypervisor data structure is the hardware page table for the currently executing process.
- When the guest OS tries to put the (VPN, PPN) mapping into the PT it is trapped (since it is a privileged operation) and the hypervisor directly puts the (VPN, MPN) mapping into the S-PT since it knows the mapping between PPN and MPN.

c) [2 points] A guest-OS has started 4 processes in a fully virtualized setting. How many page tables are maintained in the Shadow Page Table (S-PT)?

There are four page tables for the four processes and 1 page table for the guest-OS itself (totally 5 page tables).
[+1] for identifying 4 page tables for the 4 processes
[+1] for identifying 1 page table for the guest-OS itself

Memory Management [14 points]

a) (8 points) Ballooning

A data center is running multiple VMs all fully virtualized. Each VM at bootup time knows the amount of physical memory it is born with which does not change during its lifetime. The hypervisor maps those physical memory frames to machine memory frames at boot time. Each VM has a balloon driver installed.

(i) (2 points) How does the balloon driver "steal" machine memory from the guest OS and give it to the hypervisor?

Answer:

- When the driver receives a command to "inflate", it mallocs memory for the requested amount of inflation from the guest OS.

- The driver returns the machine pages associated with this allocation via a covert channel to the hypervisor unbeknownst to the guest OS.

(ii) (2 points) The hypervisor decides to steal memory from VM1 and give it to VM2 which is experiencing memory pressure. What command is given to the balloon driver in VM1? What changes in the S-PT for VM1?

Answer:

- Hypervisor to balloon: "Inflate" which results in some number of PPNs being allocated to the driver from VM1 using "malloc"

- PPN-MPN mappings for these pages are removed the S-PT of VM1

(iii) (2 points) The hypervisor decides to steal memory from VM1 and give it to VM2 which is experiencing memory pressure. What command is given to the balloon driver in VM2? What changes in the S-PT for VM2?

Answer:

- Hypervisor to balloon: "Deflate" which results in some number of PPNs being released by the driver to VM2 using "free"

- PPN-MPN mappings for these pages are installed into the S-PT of VM2

(iv) (2 points) The hypervisor decides to steal memory from VM1 and give it to VM2 which is experiencing memory pressure. What happens if VM1 has no free memory currently?

Answer:

- The balloon driver in VM1 will "inflate" using malloc.

- VM1 will page out the requested amount of memory from other processes to make that much of memory available for the driver

(b) (6 points) Memory Sharing

(i) (2 points) (Answer True/False with justification)
VM-oblivious page sharing is possible only if VMs are hosting identical versions of the same OS.

Answer: False. This technique does a "full" comparison to see if the contents of pages of disparate VMs happen to be identical then there is a potential for sharing the same machine page across the VMs even if they are hosting different OSes.

(ii) (4 points) The hypervisor is sharing a machine page across two VMs. One of them wants to write to the page. How is this handled by VM-oblivious page sharing?

Answer:
- The PPN-MPN mapping in the S-PT of both VMs will be marked copy-on-write
- The attempt to write will result in an access violation trap.
- The hypervisor will make a copy of the machine page so that the two S-PT entries are pointing to different machine pages.
- The entries will be made r/w in both the S-PTs allowing the VMs to write to their respective pages.

Parallel systems [32 points]

Shared Memory Machines [6 points]

1. (2 points) (Answer True/False with justification)

To implement a sequential consistency memory model, the underlying architecture should provide an atomic read-modify-write instruction.

Answer:

False. Atomic read and atomic write instructions are sufficient.

+0, Only False
+1, False with incorrect justification
+2, False with correct justification

2. (2 points) (Answer True/False with justification)

Sequential consistency memory model cannot be achieved on an NCC-NUMA machine.Answer: False. It becomes the responsibility of the system

software (i.e., language runtime, OS) to enforce the requirements of the SC model if the hardware does not support cache coherence.

+0, Only False
+1, False with incorrect justification
+2, False with correct justification

3. (2 points) (Answer True/False with justification)

In an NCC-NUMA machine, the private cache on each CPU only caches memory locations fetched from its local memory.Answer: True. A read/write to a remote NUMA piece always goes across the network. Such accesses are never cached locally on the processor.+0, Only True
+1, True with incorrect justification
+2, True with correct justification

M.E.Lock [10 points]

1. (2 points) (Answer True/False with justification)

It is possible to implement mutual exclusion lock algorithm in an architecture that supports atomic read and atomic write operation.

Answer: False. Need an atomic RMW instruction.

+0, Only False
+1, False with incorrect justification
+2, False with correct justification

2. [4 points] Consider the following lock algorithm using T&S:

```
while ((L == locked) or (T&S(L) == locked))
{
while (L == locked); // spin
delay (d[Pi]); // different delays for different processors
}
// success if we are here
```

(i) (Answer True/False with justification) (No credit without justification) This algorithm does not rely on hardware cache coherence [2 marks].

False.
Processors spin on cached value (While (L==locked)).The only way for them to come out of the spin loop is if the cached value changes which necessitates hardware cache coherence.
+0, Only False
+1, False with incorrect justification
+2, False with correct justification

(ii) (Answer True/False with justification) (No credit without justification) The algorithm performs especially well under high lock contention[2 marks].
True.
Upon lock release, different processors wait for different amount of delay times thus reducing the contention on the bus.
+0, Only True
+1, True with incorrect justification
+2, True with correct justification

3. (2 points) Mention two main drawbacks of Anderson's queue-based lock algorithm.

Answer:

- space complexity for each lock proportional to the number of processors

- false sharing due to large cache block size

4. (2 points) In the MCS queue based lock algorithm, if the current holder of the lock finds their next pointer is NIL, can they simply set the head node to NIL?

Answer:

No. There could be a new request that is just forming so they have to worry about that corner case.

Barriers [8 points]

---

1. (4 points) Why does MCS tree barrier work even on an NCC-NUMA machine?

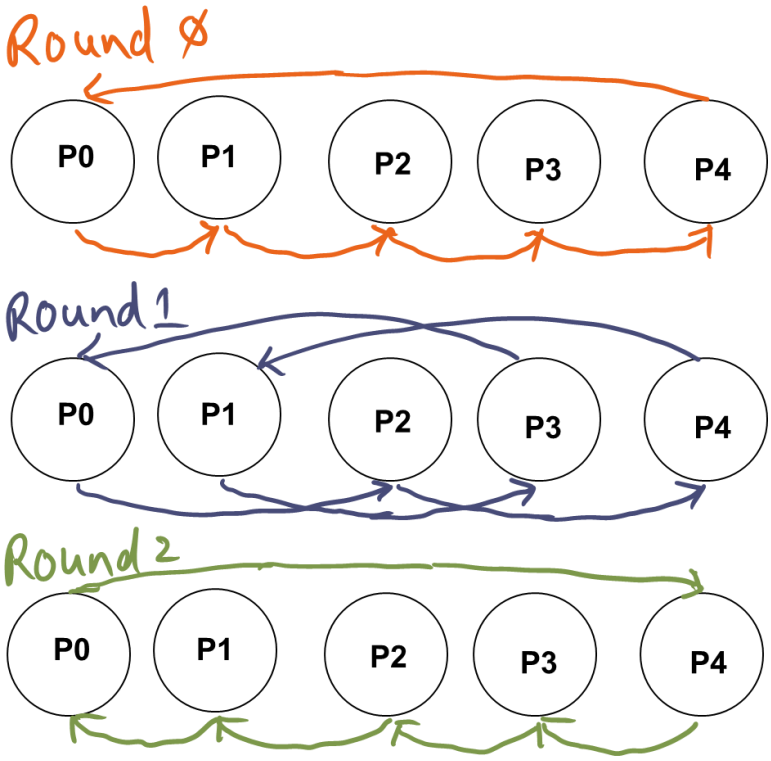
Answer:

- The shared variables that parent spin on for the arrival tree are statically allocated on the parents' NUMA pieces. When the children arrive they reach across the network to write to their respective locations in the data structure that the parent is spinning on.

- The shared variables that children spin on for the wakeup tree are statically allocated on the childrens' NUMA pieces. The parents reach across the network to write to their childrens' spin locations.

2. (4 points)

Given the following picture of the dissemination algorithm, focusing on node P3, what information does P3 know at the end of round 1? Explain your answer.



Answer:

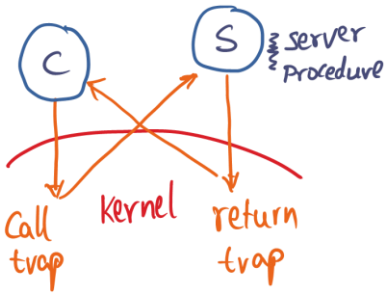
- (+1) At the end of round 0, P3 knows it is done and P2 is done

- (+2) In round 1, P3 receives a message from P1. For P1 to send this message it must have completed round 0, in which it would have received a message from P0.

- (+1) Thus at the end of round 1, P3 knows that {P2, P1, P0 and P3} have arrived at the barrier

Potpourri [8 points]

I) (2 points) In the picture shown below for a client to execute a server procedure, why are there 4 message copies incurred without any optimization as suggested in the LRPC paper?



Answer:

- client stack marshalled into RPC message by client stub
- RPC message copied into the kernel buffer
- Kernel buffer copied into the server domain buffer
- Server domain buffer copied onto the stack of the server process by the server stub

+0.5 Mentioning client stack marshalls RPC message

+0.5 Mentioning copy of msg into kernel buffer

+0.5 Mentioning kernel buffer copy into server domain buffer.

+0.5 Server domain buffer copy into server stack.

II) [4 marks] Consider a multiprocessor system consisting of 3 processors P1, P2, P3.

Consider the history of threads that ran on each of the processors below

P1: T8-> T14 -> T15 -> T7 -> T10

P2: T7 -> T11 -> T8 -> T13

P3: T8 -> T15 -> T14 -> T10

Consider the queues for these processors

P1: T1 -> T3 -> T6

P2: T2 -> T5

P3: T4

T7 and T8 just completed their I/O and are ready to be scheduled again.

Considering the state of the system, on which processors will thread T7, T8 be scheduled again?

(Assume minimum intervening + queue policy for scheduling).

Ans.

T7's Affinities P1 = 1, P2 = 3, P3 = (infinity?) never ran on P3?

Queue sizes for P1 = 3, P2 = 2, P3 = 1

I + Q values for P1 = 4, P2 = 5, P3 = infinity?

Hence T7 will run on P1

Similarly,

T8's Affinities P1 = 4, P2 = 1, P3 = 3

Queue sizes for P1 = 3, P2 = 2, P3 = 1

I + Q values for P1 = 7, P2 = 3, P3 = 4

Hence T8 will run on P2.

+1 If Affinities for T7 are correct.

+1 If I + Q values, hence processor chosen for T7 is correct.

+1 If Affinities for T8 are correct.

+1 If I + Q values, hence processor chosen for T8 is correct.

III) (2 points) Tornado suggests replicating an object to increase the concurrency for accessing the same object from different threads. Whose responsibility is it to ensure the consistency of the replicated objects? Explain.

Answer: It is the system software's responsibility (i.e., OS). The replicas are all different objects (living in different memory locations).

Hardware cache coherence will not work for keeping the replicas consistent.

(+1) Mentioning it is system software's responsibility.
(+1) Mentioning that hardware cache coherence will not work.