Santa Clara University

Department of Computer Engineering

Advanced Operating Systems (COEN383) – FINALS

Prashant Sharma (W1609993)

Q1. Multicore CPUs are beginning to appear in conventional desktop machines and laptop computers. Desktops and laptops with tens or hundreds of cores are not far off. One possible way to harness this power is to parallelize the application. Mention one application that is attractive to parallelize [3 pts]? Another approach is to parallelize the services offered by the OS (e.g., TCP processing), etc. Which approach is more promising between these two approaches [3 pts]?

Solution:

- The one of the many possible way is to harness the power in order so as to parallelize the standard desktop applications such as the word processor and the web browser.
- While another possible way is to harness the power and to parallelize the services that could be easily offered by the operating system for example the TCP processing and the most commonly used services of the library, for example, the functions of secure HTTP library.

Parallelize the OS's services is the more promising of the two approaches outlined. The reason for this is that each multicore CPU has its own operating system. As a result, a master OS assigns duties or tasks to the other remaining operating systems.

- Message forwarding and Inter Process Communication (IPC) are features of the operating system that allow tasks to be synchronized.
- The operating system allows for better memory and file management.
- OS provides a common interface layer for all of its programs.

Because of the aforementioned features, the OS appears to be more promising than traditional desktop apps.

Q2. Please answer the following:

a) Affinity scheduling reduces cache misses. Does it also reduces TLB misses [1.5 pts]? What about page faults [1.5 pts]?

Solution:

TLB misses can be reduced using Affinity Scheduling.

- Any program in progress is referred to as a process.
- When a process becomes too huge to handle, it is separated into threads.
- As a result, the right thread is executed on the right CPU affinity schedule.
- Misses from the Translation Look Aside Buffer (TLB) will also be reduced.

- This is due to the fact that TLBs are housed within the CPU.
- As a result, TLB will be executed by the appropriate CPU.
- Affinity scheduling will have no effect on the page fault.
- This is because when a page is stored in the memory of a single CPU, it is likewise stored in the memory of all other CPUs.

b) Migrating virtual machines may be easier than migrating processes, but migration can still be difficult. What problems can arise when migrating a virtual machine [3 pts]?

Solution:

Although migrating virtual computers is less complex than migrating processes, migration can still be challenging. When moving a virtual computer, there are a number of issues that can develop.

Migration:

The process of migrating a virtual machine from one server to another while it is still in production is known as migration. It should cause the least amount of disturbance to any ongoing sessions.

- Hypervisors are computer firmware and software that give the appearance of several virtual machines.
- A hypervisor is also called a virtual machine monitor (VMM).
- Virtual machines (VMs) are no longer reliant on real hardware thanks to hypervisors. This makes the migration procedure a little easier.
- Aside from migrating the complete virtual computer, special attention should be paid to moving:
  - VM configuration file.
  - Memory pages used by the virtual machine.
  - Memory bitmap.
  - Transfer of storage and network connectivity.
  - The IP addresses.
  - Virtual disk file.
- Although moving virtual machines is less complicated than migrating processes, it still presents significant challenges during implementation.
- The following concerns may emerge while transferring virtual machines:
  - Because the entire virtual machine, including the guest operating system (OS) and any processes running on it, must be relocated to the new machine, the migration period might be lengthy.
  - Problems arise because physical I/O devices do not move with the virtual machine.

Consider the following:

- The registers of such devices may store a state that is crucial to the system's operation.
- For instance, read or write operations to disk that have been started but not completed.
- Other machines will continue to transmit packets to the hypervisor, blissfully oblivious that the virtual machine has migrated, which can cause network I/O issues.
- The virtual machine will be unresponsive during the protracted migration period, even if packets are diverted to a different hypervisor.

- As a result, if the hypervisor buffers overflow, packets may incur significant delays or may be lost.

As a result, when challenges develop while migrating a virtual machine, the actions to take are as follows:

- Make sure your migration settings are correct.
- Re-establish a connection between the host servers.
- Check for server hardware compatibility and device requirements.
- Examine the servers' network connectivity.
- Examine the destination server's computational resources.

Q3. Can disabling interrupts handle concurrency correctly [2 pts]? What are mutexes and condition variables and how they are used [3 pts]?

Solution:

- Let's pretend we're using a single-processor system.
- We assure that the code inside the vital region will not be interrupted by turning off interrupts (using some form of special hardware instruction) before entering it, and thus will execute as if it were atomic.
- We re-enable interrupts (again, through a hardware command) when we're done, and the program continues as usual.
- The simplicity of this strategy is its key advantage. Unfortunately, there are other drawbacks.
- First, we must trust that this facility will not be misused by allowing any calling thread to conduct a privileged operation (turning interrupts on and off).
- Second, the method is ineffective on multiprocessors.
- It doesn't matter if interrupts are disabled if many threads on various CPUs try to enter the same critical area; threads will be able to execute on other processors and therefore enter the critical region.
- As multiprocessors become more popular, our generic solution will need to improve.
- Finally, and perhaps most importantly, this strategy can be inefficient.
- Code that masks or unmasks interrupts is typically performed slowly by current CPUs when compared to normal instruction execution.
- As a result, shutting off interrupts as a mutual-exclusion primitive is only utilized in specific situations.

Mutex:

- A mutex is a shared variable that can be unlocked or locked in one of two states.
- As a result, only one bit is necessary to represent it; but, in reality, an integer is frequently used, with 0 indicating unlocked and all other values indicating locked.
- Pthreads has a number of functions for synchronizing threads.
- To protect each vital region, the basic technique employs a mutex variable that can be locked or unlocked.
- Pthreads provides a second synchronization technique in addition to mutexes: condition variables.

- Mutexes are useful for authorizing or disallowing access to a sensitive area.
- Threads can be blocked by condition variables if a condition isn't met.
- The two strategies are almost typically used in tandem.

Q4. Authentication mechanisms are divided into three categories: Something the user knows, something the user has, and something the user is. Imagine an authentication system that uses a combination of these three categories. For example, it first asks the user to enter login name and password, then insert a plastic card (has magnetic strip) and enter a PIN, and finally provide fingerprints. Can you think of two drawbacks of this design [6 pts]?

Solution:

The combination of different authentication mechanisms will provide stronger authentication.

However, there are two drawbacks.

- First, the cost involved in implementing this system is high.
- The system incurs the cost of three different authentication mechanisms.
- Implementing fingerprint scanning would require special devices and the physical presence of the user as well.
- Also scanning magnetic strips would also require the use of special devices.
- Second, this authentication mechanism puts extra burden on the user.
- The user has to remember his login/password, carry his plastic card and remember its PIN, and has to go through the process of fingerprint matching.
- The key issue is that all this will increase the time it takes to authenticate a user, resulting in increased user dissatisfaction.
- Also, you would be using different varieties of elements to verify a person when even one of them can do the perfect job for you.
- For example, you just plan to implement user authentication based only on fingerprints, then it would be sufficient for you to validate the user based only on that as neither can someone possess that same set of prints on his hands nor is it easy to obtain someone else's fingerprints.
- Coming to magnetic strips, ATM cards have over a couple of decades been working with this technology, authenticating users/ account holders worldwide, so if implemented, they are self-sufficient to carry out authentication.
- A login id and password are the only aspect that might not be self-sufficient as we have seen in the past as well as present that login id and passwords can be vulnerable to attacks by hackers.

Q5. Please answer the following:

a) What does the following Linux shell pipeline do [1.5 pt]?

grep nd xyz | wc -l

b) A user at a terminal types the following commands:

a | b | c &

d | e | f &

How many processes are running [1.5 pt]?

c) Does it makes sense to take away a process' memory when it enters zombie state? Why or why not [1.5 pt]?

d) Explain under what situation a process may request a shared lock or an exclusive lock. What problem may a process requesting an exclusive lock suffer from [1.5 pts]?

Solution (a): Considering the above Linux shell pipeline (grep nd xyz | wc -1):

- Grep command returns the lines where a pattern exists in a file or std input.
- Here, it finds matches for "nd" in the file named xyz and returns only those lines.
- This output is supplied to wc -1 via | operator.
- wc -1 gives the number of lines in file.

Solution (b): Considering the above commands,

- There are 6 processes running – {a,b,c,d,e,f}.
- '|' redirects the output of the first process to the next process.
- The trailing '&' implies the process is run in the background.
- So, the processes that run are {a,b,c,d,e,f} – 6 processes.

Solution (c): Zombie process:

- The condition known as zombie state occurs when resources are not shared across distinct processes.
- In a shared memory multiprocessor system, processes desire to enter the crucial section, which might be an issue if multiple processes seek to enter the critical area at the same time.
- The code segment displayed is used as a solution to avoid a situation when numerous processes enter a vital region.
- When a process is in a zombie state, it means it is no longer being run.
- As a result, clearing the memory is a good idea.

Solution (d):

Shared Locks:

- Read locks are another name for shared locks.
- These locks allow many processes to read the resource at once.
- The purpose of having a read lock is to prevent another process from acquiring a write lock.

Exclusive Locks:

- Exclusive locks are often referred to as write locks because they are primarily used to write data.
- Exclusive locks enable data writing and updating.

The issue with exclusive locks is that once one is acquired on a resource, no other lock can be acquired on that resource, so other processes must wait for the lock to be released, which means the request will fail. Additionally, only one process can access the resource at a time to avoid concurrency issues.
If multiple processes request shared locks, a process that requests an exclusive lock may be blocked indefinitely. We may also remark that if readers always go before the writers, the writers may go hungry.

Q6. Why Hadoop is considered a batch processing platform and not an interactive platform [3 pts]? What is YARN, why needed, and how it can be used [3 pts]?
Solution:
Hadoop is considered a batch processing platform because:

- Batch processing is the automatic execution of a set of jobs in a computer program without the need for human interaction (non-interactive).
- It is a processing mode in the strictest sense: it involves the execution of a succession of programs, each on a set or "batch" of inputs rather than a single input.
- Hadoop MapReduce is the best framework for batch processing.
- Hadoop is a huge data processing system based on batch processing.
- This means that the data is saved over time before being processed with Hadoop.

YARN:

- The resource management and task scheduling technology in the open source Hadoop distributed processing framework is Apache Hadoop YARN.
- YARN is one of Apache Hadoop's main components, and it's in charge of assigning system resources to the many applications operating in a Hadoop cluster, as well as scheduling tasks to run on different cluster nodes.
- When compared to MapReduce's more static allocation strategy, YARN can dynamically allocate resources to applications as needed, improving resource consumption and application performance.
- Apache Hadoop YARN lies between HDFS and the processing engines required to run applications in a cluster architecture.
- Containers, application coordinators, and node-level agents supervise processing operations in individual cluster nodes.