Santa Clara University
Department of Computer Engineering
Advanced Operating Systems (COEN383, Section-1)
Final (35 points)
Time: 75 minutes

Name: Benita Rego
ECampus ID: W1628656

1. 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]?

Answer: One way to use this power is to parallelize typical desktop programs like word processors and web browsers. The only desktop application that potentially benefits from extensive parallelization is gaming, as many game elements demand intensive (parallel) computation. Despite the fact that some desktop apps have several threads, these threads are not often used for computation-intensive tasks but instead are used to assist I/O programming.

Based on the two techniques described above, it appears that parallelizing the Operating Systems' services is a preferable choice. All applications can use the OS after it has been parallelized. As an illustration, modern network cards are equipped with on-board parallel processors (network processors) that accelerate packet processing and make higher-level network services possible at line speeds (e.g., encryption, intrusion detection).
We can consider the potent processors found in video cards, which are created to offload video rendering from the primary CPU and provide programs with higher-level graphics APIs (e.g., OpenGL). Single-chip multi core CPUs might take the place of these specialized cards. Therefore, it seems like a preferable option to parallelize the OS's services.

- OS offers a standard layer of interface for all of its programs.
- OS would synchronize tasks using message passing and inter-process communication.
- The OS manages memory and files quite well.

Even while some desktop programs are multi-threaded, these threads are often used to simplify I/O programming and are not thus computationally costly. The parallelization of operating system and library functions is a more viable strategy. Modern operating systems and hardware

architectures reflect this. As the number of cores rises, different operating systems and common library functions may be parallelized using the same underlying strategy.
Additionally, operating system-level parallelization is done just once and may be used by any other program. Each application must have its own implementation if parallelization is done at the level of the application.
Because of the previous functions, operating systems are therefore preferable to desktop apps.

2. Please answer the following:
a) Affinity scheduling reduces cache misses. Does it also reduce TLB misses [1.5 pts]? What about page faults [1.5 pts]?
Answer: Affinity Scheduling helps lower the number of TLB misses:

- The term "process" refers to any ongoing program.
- A process is divided into threads when it grows too large to manage.
- The appropriate thread is thus run on the appropriate CPU affinity schedule.
- The Translation Look Aside Buffer (TLB) will experience fewer misses.
- TLB will be processed by the appropriate CPU as a result of the fact that TLBs are located within the CPU.
- The page fault will not be affected by affinity scheduling.
- This is so that every CPU's memory contains every page that is stored in the memory of a single CPU.

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 [3pts]?
Answer: Migration refers to the process of moving an operational virtual machine from one server to another. Any ongoing sessions should be as little as possible disturbed by it.
Computer firmware and software known as hypervisors create the illusion of several virtual computers. Another name for a hypervisor is a virtual machine monitor (VMM). Thanks to hypervisors, virtual machines (VMs) are no longer dependent on actual hardware. As a result, the migration process is a bit simpler.

Moving a virtual machine from one server to another while it's still in use in a production environment is known as migration. There shouldn't be much impact to any ongoing sessions. A piece of computer software and firmware known as a hypervisor gives the appearance of having several virtual computers. Another name for a hypervisor is a virtual machine monitor (VMM). Virtual machines (VMs) are no longer dependent on real hardware thanks to hypervisors. This somewhat simplifies the migrating procedure.

Virtual machine migration may be simpler than process migration, but it still presents certain technical challenges. The following difficulties may occur during virtual machine migration:

- The migration procedure itself can take a while since the whole virtual machine, including the guest operating system (OS) and all running processes, must be transferred to the new computer.
- Problems arise when physical I/O devices do not move with the virtual computer.
- Think about the following:
    - Such devices' registers could store a state that is essential to the system's smooth operation.
    - For instance, a read or write operation to the disk that has been started but not yet finished.
- Network 1/O can also be a problem because other machines may keep sending packets to the hypervisor oblivious to the virtual machine's migration.
- Think about the following:
    - During the protracted migration time, the virtual machine won't function even if packets are diverted to a different hypervisor.
    - As a result, if the hypervisor buffers overflow, packets may incur lengthy delays or possibly be lost.

Consequently, the following procedures should be taken if problems occur while migrating a virtual machine:

- Verify the migration settings.
- Connect the host servers once again.
- Check for device and server hardware compatibility.
- Check the servers' network connection.
- Check the target server's computational capacity.

3. Can disabling interrupts handle concurrency correctly [2 pts]? What are mutexes and condition variables and how they are used and also if a mutex can perform both functions [3 pts]?

Answer: Assume we are using a system with just one CPU. We guarantee that the code inside the critical area won't be interrupted and will thus run as if it were atomic by disabling interrupts (using some form of special hardware instruction) prior to entering the critical region. When we're done, we turn interrupts back on (again using a hardware command), and the program continues as usual. The simplicity of this strategy is its greatest advantage. The drawbacks, sadly, are numerous. In order to utilize this method, we must first trust that no one would misuse the privilege of allowing any calling thread to do a privileged action (turning on and off interrupts).

Second, multiprocessors cannot be used with the method. Regardless of whether interrupts are disabled, threads will still be able to execute on other processors and may do so if they attempt to access the same critical region while operating on other CPUs. Multiprocessors are

increasingly widely used, thus our generic solution will need to perform better. Third, and perhaps least significant, this strategy may not be effective. Code that masks or unmasks interruptions typically executes more slowly on current CPUs compared to code that executes instructions normally. For these reasons, disabling interrupts as a mutual-exclusion primitive is only employed in a few situations.

A shared variable known as a mutex has two possible states: unlocked and locked. In order to represent it, just 1 bit is needed; nevertheless, in reality, an integer is frequently used, with 0 denoting unlocked and all other values denoting locked.

A variety of synchronization functions are available in Pthreads. Each crucial zone is protected by a mutex variable in the basic mechanism, which may be locked or freed. Pthreads provides condition variables as a second synchronization method in addition to mutexes. Mutexes are useful for permitting or preventing access to a crucial area. Threads can become stuck if a condition is not met thanks to condition variables. Almost always, the two techniques are combined.

4. 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]?
Answer: Using all three would yield incredibly solid results for authentication. However, authentication solutions should also make the system simple to use for the user as a whole, rather than just concentrating on the security component.

This approach has one flaw, in my opinion: you would have to check someone out using a variety of factors when just one of them would be sufficient. If, for instance, you only intended to use fingerprints for user authentication, it would be sufficient for you to validate the user solely on the basis of those prints because neither anyone else can have exactly the same set of prints on his hands nor is it simple to obtain someone else's fingerprints. With regard to magnetic strips, ATM cards have been using this technology for several decades to verify users and account holders around the globe; thus, if used, they are self-sufficient for performing authentication. The only element that could not be self-sufficient is a login ID and password, since we have seen both in the past and in the present that login IDs and passwords can be targets of hacker assaults.

Second, the implementation of fingerprint scanning would require specialized equipment as well as the user's physical presence, which might pose logistical and financial problems. Using specialized equipment would be necessary to scan magnetic strips. Additionally, a complicated

verification procedure would draw fewer users. A straightforward, painless approach may be both effective and secure in terms of user authentication.

5. Please answer the following:

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

grep nd xyz | wc -l

Answer: Taking into consideration the Linux shell pipeline (grep nd xyz | wc -1) above:

- The Grep command returns the lines in a file or standard input that match a pattern.
- In this case, it only returns lines that match the string "nd" in the file xyz.
- The | operator is used to send this output to wc -1.
- wc -1 displays the file's line count.

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

a | b | c &
d | e | f &

How many processes are running [1.5 pt]?

Answer: Taking an instance, from the above commands:

- There are six processes running: {a,b,c,d,e,f}.
- The symbol "|" sends the output of one operation to another.
- The "&" at the end suggests that the procedure is running in the background.
- Therefore, there are six processes running: {a,b,c,d,e,f}.

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

Answer: Zombie process: When resources are not distributed across many processes, the zombie state is created.

In a shared memory multiprocessor system, processes want to enter the vital region, which might be problematic if several processes want to do so at once. The code segment that is shown is utilized as a preventative measure to stop several processes from entering a critical area. A process is no longer being used when it is in a zombie state. As a consequence, it makes sense to clean your memory.

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]?

Answer: Shared Locks: Read locks are another term for shared locks, which let many processes read the resource simultaneously. A read lock's main function is to stop another process from obtaining a write lock.

Exclusive Locks: Because they are primarily used to write data, exclusive locks are sometimes referred to as "write locks." Exclusive locks allow for data updating. Exclusive locks have a drawback in that, once one is obtained on a resource, no more locks may be obtained on that resource. As a result, other processes are forced to wait for the lock to be released, which results in the request failing. In order to prevent concurrent problems, only one process can use the resource at once.

A process that demands an exclusive lock may be stalled indefinitely if other processes request shared locks instead of it. We may also add that authors might go hungry if readers always come first.

6. Why is Hadoop 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]?
Answer: Hadoop is regarded as a platform for batch processing because:

- Batch processing is the automated, computer-driven completion of a number of tasks without involving any human involvement (non-interactive).
- In the strictest sense, it is a processing mode since it requires running a series of programs, each on a collection of inputs known as a "batch" as opposed to a single input.
- Hadoop is a massive data processing system based on batch processing, which means that the data is saved over time before being processed using Hadoop.
- Hadoop MapReduce is the finest framework for batch processing.

YARN: The open source Hadoop distributed processing system uses the Apache Hadoop YARN technology for resource management and task scheduling. One of the fundamental parts of Apache Hadoop, YARN is in charge of assigning system resources to the many applications operating inside a Hadoop cluster and planning the execution of tasks across several cluster nodes.

Apache Hadoop YARN is positioned between HDFS and the application-processing engines in a cluster design. Containers, application coordinators, and node-level agents that keep track of processing activity on specific cluster nodes are combined with a central resource management. When opposed to MapReduce's more static allocation method, YARN may dynamically distribute resources to applications as needed, which is intended to enhance resource consumption and application performance.

Furthermore, YARN offers a variety of scheduling options, all of which rely on a queue style for processing task submission. As implied by its name, the default FIFO Scheduler executes programs on a first-in-first-out basis. For clusters that are shared by several individuals, though, it might not be the best option. Instead, depending on a weighting criteria that the scheduler

determines, Apache Hadoop's pluggable Fair Scheduler utility allots each task executing at the same moment its "fair share" of cluster resources.

The open source Hadoop distributed processing architecture uses Apache Hadoop YARN for resource management and job scheduling.

- One of Apache Hadoop's key components, YARN, is in charge of scheduling jobs to execute on various cluster nodes as well as allocating system resources to the several applications running in a Hadoop cluster.
- YARN can dynamically assign resources to applications as needed, which improves resource usage and application performance when compared to MapReduce's more static allocation technique.
- HDFS and the processing engines needed to run applications in a cluster architecture are separated by Apache Hadoop YARN.
- Processing processes in individual cluster nodes are supervised by containers, application coordinators, and node-level agents.