CS 620: Homework 3

Carmen St. Jean

September 20, 2012

1. *(6) Name 1 key OS change that occurred with each of these:*

   (a) *uni-programmed to multiprogrammed:*
   The key OS change from uni-programmed to multi-programmed systems was the ability to schedule and share resources, which included the ability to keep each program within its own space.

   (b) *batch to interactive:*
   The key OS change from batch to interactive was that an interface was required so the user could interact with the computer, because before only the operator interacted directly with the batch machine.

   (c) *stand-alone to networked:*
   The key OS change from stand-alone to networked is the ability to transfer data to other computers and communicate with other computers.

   (d) *minicomputer to personal computer:*
   The key OS change from minicomputer to personal computers was the decrease in the number of I/O devices required, since it went from many terminals for a single CPU to only one terminal.

   (e) *personal computer to hand-held:*
   The key OS change from personal computers to hand-held computers was the nature of the I/O devices involved because of the transition from mouse and keyboard to touch screen.

   (f) *networked to cloud:*
   If by "networked" you mean the brief phase of networked computers a few months ago, then the key OS change from networked to cloud is the ability to use remote virtual resources of all kinds

(storage, computing, etc...) than simply data storage. If by "networked" you mean any computer connected to the web, then the difference is that a computer connected to the cloud uses the remote resources primarily, while remote resources to a networked computer would be secondary to its own resources.

2. *(2)Why is it important for an application programmer to understand the underlying hardware and OS platform over which her software will run? Explain using an example.*

   It is important for an application programmer to understand the underlying hardware and OS platform because the programmer needs to be able to properly leverage the OS and hardware to her advantage. She also needs to be aware pf the pitfalls and bottlenecks of the hardware and OS so she does not run into problems. For example, if a programmer was trying to do bit operations with a program, but didn't realize it was little endian instead of big endian, then her program won't execute properly. Or if a programmer wrote a multithreaded program but it turns out the machine it will run on has a single processor, then her program won't be as efficient as she had hoped.

3. *(3) Many computers have two levels of CPU caches - L1 and L2 cache. In some computers, the L1 and L2 caches are non-inclusive (i.e., data in L1 and L2 are disjoint), while in other computers L1s data set is a subset of L2s data set.*

   (a) *Give one advantage of non-inclusive L1, L2 caches as opposed to inclusive.*

   Since L1 and L2 are exclusive of each other, they can store more data than inclusive caches, especially if L1 and L2 are close to each other in size. The more data you have in your caches, the less likely you are to miss, so non-inclusive caches has this advantage over inclusive caches.

   (b) *Give one disadvantage of non-inclusive L1, L2 caches as opposed to inclusive.*

   The disadvantage of non-inclusive caches is that you may have to check both caches. This problem is especially exagerated if both L1 and L2 miss. Inclusive caches are better in this sense because you only need to check a single cache, so a double-miss

in a non-inclusive cache is essentially twice as expensive as a miss in an inclusive cache.

4. *(3) There are 2 computers: computer1 has one CPU; computer2 has two CPUs that are half as fast as computer1s CPU. Under similar workloads, would you expect the performance (response time) of computer1 to be faster, slower, or identical to computer2. Justify your answer.*

   The times would be approximately identical, but computer1 would turn out to perform slightly faster than computer2. The response time for one processor would be better than the combined response time of two processors. But the nature of your work also will affect the performances and in some situations computer2 might be slightly faster.

5. *(3) Now consider the CPU scheduler for computer2. Assume that all processes in the computer are identical and have the same priority. From a performance perspective, would it be better for there to be one queue of processes (waiting to execute) or would it be better to have two queues of processes - one for each CPU. This is equivalent to having a single line (similar to the waiting line in a bank) or multiple lines (similar to the waiting lines in a grocery store).*

   A single queue is better because it makes for overall faster service time. The problem with multiple lines is that some lines may end up moving significantly slower than others, so some processes may end up waiting for a very long time and could even be serviced after processes that got into a different line sometime after. This happens often in a grocery store for example. One line may appear shorter so you join it, only for the transaction of the person in front of you to take a lot of time. Perhaps the person was only purchasing a few items so it seemed liked a good choice for a line. However, their transaction could be delayed through various unforeseeable problems that required the cashier to wait for a manager to come over. Meanwhile, people that got into different lines well after you got in line could already be leaving the grocery store. This problem does not occur in a bank, where everyone waits in a single line so it truly is first-come, first-serve.

6. *(1) What is swapping?*

   Swapping is when a medium term scheduler removes a process from memory and from active contention for the CPU to reduce the degree

of multiprogramming. The process can be reintroduced into memory later and execution can resume where it left off.

7. *1) What is the objective of swapping?*

   The objective of swapping is to improve the process mix or to free up memory if the available memory was overcommitted.

8. *(2) Compare and contrast process context switch and swapping.*

   A context switch is when an interrupt occurs to a process currently running on the CPU. The context is saved, the process is suspended, and eventually the process is resumed later on. Swapping and a context switch are similar in the sense that they have to do with suspending and restoring a process. The difference is that a context switch suspends a running process, while swapping happens to a waiting process.

9. *(2) Can a process be in more than 1 computer queue at a time? If yes, then give an example. If no, explain.*

   No, a process cannot be in more than one queue at a time. That would imply that a process is capable of having multiple states at once, which is impossible.

10. *(1) What is the disadvantage of using a table (Process Table) instead of a linked list, to keep track of all processes in the computer?*

    The disadvantage of using a table is that you must know in advance approximately how many processes will be in your table in order to create your table. A linked list can easily grow if you have a pointer to the end of the linked list. For the same reason it is also easy to add to the end of the linked list, which makes a linked list advantageous since it is constantly having items added to it.

11. *(1) What is the advantage of using a table (Process Table) instead of a linked list, to keep track of all processes in the computer?*

    The advantage of using a table is that you gain random access with a table, as compared to a linked list which has $O(n)$ traversal because it requires sequential access. Given that there may be thousands and thousands of processes in the process table, it is critical to have fast access time. Likewise, removing from a linked list will take a long time if the entire linked list needs to be traversed to find the process to be removed.