HW2 Dry part

Date:

Submitters:

Q1:

1.The statement is false. Since the static priorities of A and B are equal, the only thing that

differentiate them is their bonus. Since bonus varies only according to sleep\_average, and B is IO-

bound while A is CPU-bound, the sleep\_average of B is bigger than the sleep\_average of A,

therefore, the bonus of B is higher than A’s. Which means that the prio of B is lower than that of A

(since prio = static\_prio - bonus).

2. A will get a bigger timeslice than B and will be executed earlier than B in epoch (since the

processes are arranged according to their priorities in the runqueue).

3. A will get a bigger timeslice than B. A will be executed earlier than B in epoch. Moreover, A

will get a better reaction time and A can stop the execution of B when returning from waiting if it still has remaining timeslice.

4.

A. Yes, if an OTHER process yields while there are no processes with prio equal to his and his prio is the best prio in the runqueue, then he will be re-chosen to run by schedule and thus n context switch will be performed.

B. In this case, if the process that yielded is the only process in the active prio\_array then the current->prio of the process will be given the value of MAX\_PRIO and the yield syscall will try to add the process to queue[MAX\_PRIO] in the active prioarray, which is an illegal memory address for this operation since the indexes of queue run from 0 to (MAX\_PRIO -1).

C. In this case, current->prio might be given a value smaller than it priority. This is problematic since a newly woken (or started) process with a genuinely higher priority than the yielding process might be selected after it by schedule, thereby changing the intended behavior of the scheduling algorithm.

D. The son will run first. This is done to avoid unnecessary Copy On Write commands if the son execvs shortly after the fork.

Q2:

1. The difference between working with multiple CPUs and a single CPU, is that when working

with multiple CPU, a CPU’s schedule function checks whether it’s run queue is empty or not. If it

is, load\_balance is called. load\_balance looks for the busiest CPU and moves tasks from

it to the idle CPU if the busiest run queue is significantly busy.

2. pros: If each CPU has it’s own run queue, each CPU can access it’s run queue quickly and

efficiently, without worrying about receiving data contamination from other CPUs or causing

data contamination to other CPUs. cons: If each CPU has its own run queue, its possible that 1

run queue will have multiple processes in the ready mode while another CPU runs a process

from its own run queue with a lesser priority. In such cases we see that some CPUs might not

be working on the highest priority process that is technically available to the system as a whole.

3. a scheduling domain is a hierarchical system that separates CPUs into different groups and groups of groups with a structure similar to a tree. It helps speed up the balancing function by allowing a CPU to find a busy CPU that needs to get some load off without iterating through all the CPUs in the computer.

4. Once every tick, reschedule\_tick is invoked in a certain CPU, it iterates over the path from the local domain to the top level domain and checks if any of the domains need rebalancing using the load\_balance function. The load\_balance function has statistical information that represents the current average load in a CPU and in a group. It looks for the busiest group in a domain and If the group returned is not the groups containing the current CPU it looks for the busiest CPU and checks if that queue is busy enough to warrant a rebalancing.

5. If load balance found a sufficiently busy CPU to move processes from, it moves processes from the chosen CPU's runqueue to the current CPU's runqueue. Making sure that:

- a running process

-a process that is not allowed to run on the current CPU

-or a process that is "cache hot"

Are not moved. The number of processes moved is at most ((max\_load - nr\_running) / 2).

Since processes are moved only if there is a significant difference between the load of the current CPU and the busiest CPU, the CPUs are not completely balanced but rather only somewhat relieved of load when they become too busy.