Operating Systems and Networks

Mini Projects / Mini Project 3

Mini Project 3: PBS, CoW (Memory management) and Concurrency.

Deadline: 11:59pm 10 November 2023

GitHub Classroom

Welcome to the final mini-project of the course! Follow this link to accept the mini-project. You will then be assigned a private repository on GitHub. This is where you will be working on the mini project. All relevant instructions regarding this project can be found below.

xv-6 Revisited:

1. Modified Priority Based Scheduler in xv-6. [30 Marks]

Implement a preemptive priority-based scheduler that selects the process with the highest priority for execution. In case two or more processes have the same priority, we use the number of times the process has been scheduled to break the tie. If the tie remains, use the start-time of the process to break the tie(processes with lower start times should be scheduled further).

There are two types of priorities.

- The **Static Priority of a process** (SP) can be in the range $[0, 100]$, a smaller value will represent higher priority. Set the default priority of a process as 50.
- The **Dynamic Priority** (DP) of a process depends on its Static Priority and RBI (recent behaviour index).

The **RBI** (Recent Behaviour Index) of a process measures its recent behavior and is used to adjust its dynamic priority. It is a weighted sum of three factors: *Running Time (RTime), Sleeping Time (STime), and Waiting Time (WTime)*. The default value of **RBI** is 25.

- Definition of the variables:

- $RTime$: The total time the process has been running since it was last scheduled.
- $STime$: The total time the process has spent sleeping (i.e., blocked and not using CPU time) since it was last scheduled.
- $WTime$: The total time the process has spent in the ready queue waiting to be scheduled.
- $$RBI = max\left(Int\left(\frac{3 * RTime - STime - WTime}{RTime + WTime + STime + 1} * 50\right), 0\right)$$
- $$DP = \min\left(SP + RBI, 100\right)$$

- Use *Dynamic Priority (DP)* to schedule processes.
- To change the Static Priority add a new system call `set_priority()` .

  ```
  int set_priority(int pid, int new_priority)
  ```

  The system call returns the old Static Priority of the process. In case the priority of the process increases(the value is lower than before), then rescheduling should be done. *Note that calling this system call will also reset the Recent Behaviour Index (RBI) of the process to 25 as well.*

  Also make sure to implement a user program `setpriority` , which uses the above system call to change the priority. And takes the syscall arguments as command-line arguments.

  ```
  setpriority pid priority
  ```

- *Along with implementing the scheduler, analyze and observe the effectiveness of SP and RBI (thus RTime, WTime, STime variables) in updating the DP values and assigning priorities to multiple processes. Submit your brief analysis along with the report. Feel free to add plots/proofs to support your analysis and observations.*

2. Copy on Write Fork in xv-6 [20 Marks]

In xv6 the `fork` system call creates a duplicate process of the parent process and also copies the memory content of the parent process into the child process. This results in inefficient usage of memory since the child may only read from memory.

The idea behind a copy-on-write is that when a parent process creates a child process then both of these processes initially will share the same pages in memory and these shared pages will be marked as copy-on-write which means that if any of these processes will try to modify

the shared pages then only a copy of these pages will be created and the modifications will be done on the copy of pages by that process and thus not affecting the other process.

The basic plan in COW fork is for the parent and child to initially share all physical pages, but to map them read-only. Thus, when the child or parent executes a store instruction, the RISC-V CPU raises a page-fault exception. In response to this exception, the kernel makes a copy of the page that contains the faulted address. It maps one copy read/write in the child's address space and the other copy read/write in the parent's address space. After updating the page tables, the kernel resumes the faulting process at the instruction that caused the fault. Because the kernel has updated the relevant PTE to allow writes, the faulting instruction will now execute without a fault.

Implement copy-on-write (COW) fork in xv6.

Concurrency

1. Cafe Sim [20 marks]

You have been given the task to simulate the operation of a small cafe. The cafe owner wants to ensure that customers can get their coffee efficiently. In your report, you will provide insights based on the simulation results.

DURING THE SIMULATION:

- There are `K` coffee types.
- Each coffee type `c` takes some time `t_c` to prepare.
- The cafe has `B` baristas.
- There are `N` customers waiting to get their coffee.
- Each customer `i` arrives at some time `t_arr_i`, and orders only one coffee `x`.
- Each customer `i` arrives at some tolerance time `tol_i`, after which their patience runs out and they will instantly leave the cafe (bad).

ASSUMPTIONS:

- Simulation begins from 0 seconds.
- The cafe has unlimited ingredients.
- If a customer arrives at time `t`, they place the order at time `t`, and the coffee starts getting prepared at time `t+1`.
- If a customer arrives at time `t`, and they have tolerance time `tol` => they collect their order only if it gets done on or before `t + tol`.

- Once an order is completed, the customer picks it up and leaves instantaneously.
- If a customer was already waiting, once a barista finishes their previous order, say at time `t`, they can start making the order of the waiting customer at time `t+1`.
- The cafe has infinite seating capacity.

Your simulation should utilize multi-threading concepts and avoid potential issues like deadlocks and busy waiting. Implement the problem using semaphores and mutex locks to ensure thread safety. In your report, provide implementation details and address the given follow-up questions. Make sure to colour code your output using the specified colours.

INPUT FORMAT (ALL SPACE SEPARATED):

The first line contains `B K N` The next `K lines` contain `c t_c` The next `N lines` contain `i x t_arr_i tol_i`

OUTPUT FORMAT:

- When a customer `c` arrives, print [white colour] `Customer c arrives at t second(s)`
- When a customer `c` places their order, print [yellow colour] `Customer c orders an espresso`
- When a barista `b` begins preparing the order of customer `c`, print [cyan colour] `Barista b begins preparing the order of customer c at t second(s)`
- When a barista `b` successfully complete the order of customer `c`, print [blue colour] `Barista b successfully completes the order of customer c`
- If a customer successfully gets their order, print [green colour] `Customer c leaves with their order at t second(s)`
- If a customer's patience runs out, print [red colour] `Customer c leaves without their order at t second(s)`

EXAMPLE:

Input:

```
2 2 3
Espresso 3
Cappuccino 10
1 Cappuccino 0 15
2 Espresso 3 6
3 Espresso 3 5
```

Output:

```
Customer 1 arrives at 0 second(s)
Customer 1 orders a Cappuccino
Barista 1 begins preparing the order of customer 1 at 1 second(s)
Customer 2 arrives at 3 second(s)
Customer 2 orders an Espresso
Customer 3 arrives at 3 second(s)
Customer 3 orders an Espresso
Barista 2 begins preparing the order of customer 2 at 4 second(s)
Barista 2 completes the order of customer 2 at 7 second(s)
Customer 2 leaves with their order at 7 second(s)
Barista 2 begins preparing the order of customer 3 at 8 second(s)
Customer 3 leaves without their order at 9 second(s)
Barista 1 completes the order of customer 1 at 11 second(s)
Barista 2 completes the order of customer 2 at 11 second(s)
Customer 1 leaves with their order at 11 second(s)

1 coffee wasted
```

QUESTIONS

1. **Waiting Time:** Calculate the average time a customer spends waiting for their coffee. Determine if this is more than (and how much) if the cafe had infinite baristas. Coffee prep time is not part of waiting time.
2. **Coffee Wastage:** Determine how many coffees are wasted. Print the number of coffees wasted after the simulation ends.

2. Ice Cream Parlor Sim [30 marks]

Sunny's Ice Cream Parlor is renowned for its delectable ice cream creations and has garnered fame through media coverage. To meet the growing demand, Sunny has expanded the offerings, employing skilled ice cream machines. Your task is to simulate the operation of Sunny's Ice Cream Parlor.

DURING THE SIMULATION:

- There are `N` ice cream machines.
- Every ice cream machine `m` works from time `tm_start` to time `tm_stop`.
- There are `F` ice cream flavours
- Every ice cream flavour `f` takes unique time `t_f` to prepare.

- There are `T` toppings.
- Every topping `t` takes constant time `t_t` to put.
- A topping `t` is present in limited quantity, `q_t`
- There is a capacity of `K` customers in the parlour.
- A customer `c` arrives at time `t_arr`

ASSUMPTIONS:

- Simulation begins from 0 seconds.
- If a customer arrives at time `t`, they place the order at time `t`, and the order starts getting prepared at time `t+1`.
- If a customer's order can only be partially completed, they must be rejected completely, no part of their order should waste ingredients.
- If a customer leaves due to ingredient shortages, they leave the second they're informed (at `t`), and their spot becomes free from the next second i.e. `t+1`.
- Once an entire order is completed, the customer picks it up and leaves instantaneously.
- A machine cannot start preparing an order if it will have to stop working in the middle of that order.
- If a machine starts working at time `t`, and a customer was already waiting since some time `<t`, the machine can instantly start preparing their order.

Here's a structured breakdown of the scenario:

1. Ice Cream Machines:
   - Sunny's parlor has N ice cream machines, and they have individual work shifts.
   - Ice machines unfortunately work at different times during the day.
2. Ice Cream Varieties:
   - Sunny offers an assortment of ice cream flavors, each with its own preparation time and ingredient requirements.
   - Some ingredients, like vanilla and chocolate syrup, are available in abundance, while others, such as fresh fruits and whipped cream, are limited for the day.
   - If all limited ingredients are depleted, the parlor closes for the day.
3. Customers:
   - The parlour can accommodate a maximum of `K` customers at a time. Customers can place orders for multiple ice creams with diverse flavors and toppings.
   - Customers submit their orders immediately upon entering. If ingredient shortages prevent fulfillment, they receive immediate notification (after order is taken) and can choose to exit the parlour.

- If an order can be fulfilled, customers must wait for their ice creams to be prepared and brought to the pickup spot.

Your simulation should utilize multi-threading concepts and avoid potential issues like deadlocks and busy waiting. In your report, provide implementation details and address the given follow-up questions. Make sure to colour code your output using the specified colours.

INPUT FORMAT (EVERYTHING IS SPACE SEPARATED):
The first line contains `N K F T` The next `N` lines contain machine timings `tm_start tm_stop` The next `F` lines contain `f t_f` The next `T` lines contain `t q_t` [Note: `q_t` will be given -1 for unlimited quantity]

The rest of the lines will have the structure `c t_arr number of ice creams they want to order flavour of ice cream 1 topping 1 topping 2 ... topping n ... flavour of ice cream n topping 1 topping 2 ... topping n`

OUTPUT FORMAT:

- When a customer `c` enters at time `t`, print [white colour] `Customer c enters at t second(s)`
- Print their order [yellow colour]:

```
Customer c orders 2 ice creams
Ice cream 1: vanilla caramel
Ice cream 2: chocolate candy
```

- When a machine `m` is starts preparing the order of customer `c`, print [cyan colour] `Machine m starts preparing ice cream 1 of customer c at t seconds(s)`
- When a machine `m` is completes preparing the order of customer `c`, print [blue colour] `Machine m completes preparing ice cream 1 of customer c at t seconds(s)`
- When a customer's order is complete, as they will pick it up immidiately, print [green colour] `Customer c has collected their order(s) and left at t second(s)`
- If a customer was rejected due to ingredient shortage, print [red colour] `Customer c left at t second(s) with an unfulfilled order`
- If a customer `c` was not serviced even when the parlour is closing (last machine has stopped working), print [red colour] `Customer c was not serviced due to unavailability of machines`
- When a machine `m` starts working, print [orange colour] `Machine m has started working at t second(s)`
- When a machine `m` stops working, print [orange colour] `Machine m has stopped working at t second(s)`

EXAMPLE:

Input:

```
2 3 2 3
0 7
4 10
vanilla 3
chocolate 4
caramel -1
brownie 4
strawberry 4
1 1 2
vanilla caramel
chocolate brownie strawberry
2 2 1
vanilla strawberry caramel
```

Output:

```
Machine 1 has started working at 0 second(s)
Customer 1 enters at 1 second(s)
Customer 1 orders 2 ice creams
Ice cream 1: vanilla caramel
Ice cream 2: chocolate brownie strawberry
Machine 1 starts preparing ice cream 1 of customer 1 at 2 second(s)
Customer 2 enters at 2 second(s)
Customer 2 orders 1 ice cream(s)
Ice cream 1: vanilla strawberry caramel
Machine 2 has started working at 4 second(s)
Machine 2 starts preparing ice cream 2 of customer 1 at 4 second(s)
Machine 1 completes preparing ice cream 1 of customer 1 at 5 seconds(s)
Machine 1 has stopped working at 7 second(s)
Machine 2 completes preparing ice cream 2 of customer 1 at 8 seconds(s)
Customer 1 has collected their order(s) and left at 8 second(s)
Machine 2 has stopped working at 10 second(s)
Customer 2 was not serviced due to unavailability of machines
Parlour Closed
```

QUESTIONS:

1. **Minimizing Incomplete Orders:** Describe your approach to redesign the simulation to minimize incomplete orders, given that incomplete orders can impact the parlor's reputation. Note that the parlor's reputation is unaffected if orders are instantly rejected due to topping shortages.
2. **Ingredient Replenishment:** Assuming ingredients can be replenished by contacting the nearest supplier, outline how you would adjust the parlour order acceptance/rejection process based on ingredient availability.
3. **Unserviced Orders:** Suggest ways by which we can avoid/minimise the number of unserviced orders or customers having to wait until the parlor closes.

Report [10 marks]

- Submit a report consisting of concise answers to all the questions described in each specification.

Guidelines:

1. Do not change the basic file structure given on Github Classroom.
2. No deadline extensions will be granted.

Do NOT take codes from seniors or your batch mates, by any chance. We will extensively evaluate cheating scenarios along with the previous few year's submissions.

Viva will be conducted during the evaluations, related to your code and also the logic/concept involved. If you're unable to answer them, you'll get no marks for that feature/topic that you've implemented.

Copyright © 2023 Karthik Vaidhyanathan. Distributed by an MIT license.