Grading rubrics:

Scheduler Activations

Answer
For I/O:

1. When a user-level thread blocks in the kernel (e.g., on I/O or page fault), the kernel creates a new scheduler activation
2. This new activation is used to notify the user-level thread system via an upcall that the original thread has blocked
3. The user-level thread scheduler can then use this new activation to:
    - Remove the blocked thread from its previous scheduler activation context
    - Return the old activation to the kernel for reuse
    - Schedule another ready thread to run on that processor

Maintaining the Processor-Activation Invariant:

1. The kernel will make sure there is exactly one scheduler activation per processor assigned to the address space
2. When a thread blocks, the kernel creates a new activation before notifying the user level, maintaining the count
3. When a blocked thread becomes ready (e.g., I/O completes), the kernel:
    - Preempts one of the address space's processors
    - Creates a new activation to notify the user level of both the completion and preemption
    - Allows the user level to reschedule both the previously blocked and preempted threads
4. The user level can recycle old activations by returning them to the kernel once their threads are removed from their context

Rubric:

I/O and Page Fault Handling (10)

- Creation of new activation for notification (3 pts)
- Upcall to user-level thread system (3 pts)
- Handling of blocked thread state/context (2 pts)
- Ability to schedule new thread on processor (2 pts)

Maintaining Invariant: (10)

- One activation per processor rule (2 pts)
- New activation creation (2 pts)
- Handling of thread unblocking (3 pts)
- Processor preemption process (2 pts)

- Activation recycling mechanism (1 pts)

ghOSt

Isolation is provided through enclaves, and dedicated ghOSt agents managing CPUs belonging to them Explain what enclaves are and how agents map to them. (4).
Guarantees provided:

1. Containing faults: If an agent of an enclave fails, only the enclave is affected, without affecting the rest of the CPUs or workloads. (3)
2. Interference protection: An agent can only schedule threads on CPUs belonging to a specific enclave, ensuring independence and CPU core isolation. (3)

Use discretion to award marks if the vocabulary for guarantees doesn't match the rubric, but conveys the same meaning.

3.

```
void scheduler (struct task_struct* last_process, size_t runtime) {
        // Check if the last process is finished
        if (last_process->state == FINISHED) {
        // If it is, delete the process from the run queue
        delete_process(last_process);
        } else {
        // Otherwise, update the virtual runtime of the process
        last_process->vruntime += (runtime * last_process->weight);
        }
        // Sort the run queue
        sort_run_queue();
        // Get the next process in the run queue
        struct task_struct* next_process = get_next_process();
        // Calculate the timeslice to be assigned for the process
        size_t next_timeslice = next_process->weight / get_total_weight();
        // Schedule the process
        run_process(next_process, next_timeslice);
}
```

+4 Check if the process is finished
    -2 If checking if the process is finished after getting the next process in the queue
+2 Delete the process if it is finished
    -1 If deleting the process after getting the next process in the queue
+4 Otherwise, update the virtual runtime of the process:
    -2 If calculation is wrong
    -1 If updating the virtual runtime of the process after getting the next process in the queue
+2 Sort the run queue
    -1 If the queue is sorted before updating the virtual runtime of the last process or after getting the next process
+2 Get next process in the run queue
+4 Calculate the timeslice assigned for the process

-2 If calculation is wrong
+2 Schedule the process
-1 For each additional incorrect line of code

5. (a) write amplification is 2 because only half of the base pages in a super page are dirty, and the rest half is clean.

5. (b)
- They calculate the hash value for each individual base page in a superpage when the superpage is first loaded into the DRAM. (50% of how many points you wanna assign to question 5.b)
- Later when swapping them. by re-calculating the hash value for each individual base page, and compare with the previous one, it knows if a base page is dirty or not, and only issue I/O to write the dirty base page (50% of how many points you wanna assign to question 5.b)

TMO
(a)
HDD Cheapest, slowest, can offload the least but good for application which has a large very cold memory region
nvmeSSD faster can offload more
Both HDD and nvmeSSD have to consider ware leveling issue

Memory is the fastest but the most expansive, also workload dependent

Rubric:
HDD slowest random access SSD medium random access Memory fastest random access (3)
HDD cheapest SSD medium Memory most expansive (3)
HDD nvmeSSD ware leveling issue (2)
Memory workload dependent (2)

(b)
Motivation:
The motivation of using PSI is because PSI provides a more comprehensive and direct way by reporting on the exact amount of time that the processes are stalled because of memory, CPU and I/O so the system can strike a balance between memory saving and the impact on application

Major faults is not as accurate as PSI because (1) application can perform differently to a major faults (2)
Major faults takes different amount of time for different medium (3) major faults can because of app phase change or cold start up not because of lack of memory

Rubric:
What is PSi (2)
Strike a balance (2)

2 out of 3
Different impact on APP (3)
Different medium different fault time (3)
Major faults that anyway happens (3)