ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

CAPSTONE PROJECT 1
ĐÁNH GIÁ HIỆU NĂNG HỆ THỐNG (CO3007)

Thiết kế và Mô phỏng Hệ thống Lập lịch MLFQ cho CPU Đa nhân

STT	Họ và tên	MSSV
1	Hồ Đăng Khoa	2211588
2	Huỳnh Bùi Ngọc Khoa	2211589
3	Nguyễn Xuân Hy	2211409
4	Trần Tuấn Kha	2211418
5	Châu Trần Minh Khôi	2420005
6	Võ Hoàng Huy	2211299
7	Võ Tấn Việt	2433241

Thành phố Hồ Chí Minh, tháng 11 năm 2025

Mục lục

Danh sách hình vẽ

1 Phân tích hệ thống

1.1 Giới thiệu và bối cảnh

Lập lịch CPU là một trong những chức năng cốt lõi của hệ điều hành, đặc biệt là trong các hệ thống đa nhân (multi-core) hiện đại. Mục tiêu của bộ lập lịch là quản lý việc cấp phát các lõi CPU cho các tiến trình một cách hiệu quả, cân bằng giữa hai nhu cầu đối nghịch: thời gian phản hồi nhanh cho các tác vụ tương tác (I/O-bound) và thông lượng (throughput) cao cho các tác vụ tính toán chuyên sâu (CPU-bound).

Thuật toán Hàng đợi Đa cấp Phản hồi (Multilevel Feedback Queue - MLFQ) được thiết kế để giải quyết vấn đề này. Bằng cách sử dụng nhiều hàng đợi với các mức ưu tiên khác nhau và cho phép các tác vụ di chuyển giữa các hàng đợi dựa trên hành vi của chúng, MLFQ có thể tự động "học" và phân loại tác vụ, ưu tiên các tác vụ tương tác trong khi vẫn đảm bảo các tác vụ CPU-bound được thực thi.

Dự án này tập trung vào việc thiết kế và mô phỏng một hệ thống lập lịch MLFQ cho môi trường CPU đa nhân ($c \geq 4$).

1.2 Mô hình Hệ thống Hàng đợi (M/M/c)

Hệ thống của chúng ta được mô hình hóa dựa trên lý thuyết hàng đợi.

- **Customers (Khách hàng):** Các tiến trình (processes) cần xử lý CPU.
- **Server (Máy chủ):** Các lõi CPU (CPU Cores). Trong thiết kế này, chúng ta sử dụng $c \geq 4$.
- **Arrival Process (Quá trình đến):** Luồng các tiến trình đến hệ thống được giả định tuân theo phân phối Poisson. Do đó, thời gian giữa hai lần đến liên tiếp (inter-arrival time) tuân theo phân phối mũ (Exponential). Đây là đặc tính **M** (Markovian).
- **Service Process (Quá trình phục vụ):** Thời gian CPU cần thiết để hoàn thành một tiến trình (`total_cpu_time`) cũng được giả định tuân theo phân phối mũ. Đây là đặc tính **M** (Markovian).

Dựa trên các giả định trên, hệ thống CPU có thể được mô tả bằng ký hiệu Kendall là **M/M/c**.

1.3 Mục tiêu và phạm vi của hệ thống

Mục tiêu trọng tâm của dự án là thiết kế và triển khai một mô hình mô phỏng chi tiết nhằm đánh giá hiệu năng của thuật toán lập lịch Hàng đợi Đa cấp Phản hồi (MLFQ) trong môi trường đa nhân M/M/c với số lõi CPU $c \geq 4$.

Cụ thể, hệ thống sẽ được kiến trúc với một bộ lập lịch MLFQ ba cấp, bao gồm hai hàng đợi ưu tiên cao sử dụng kỷ luật Round-Robin (Q1-RR, Q2-RR) và một hàng đợi ưu tiên thấp nhất hoạt động theo cơ chế First-Come, First-Served (Q3-FCFS). Logic mô phỏng sẽ tái hiện đầy đủ các quy tắc nền tảng của MLFQ, bao gồm cơ chế tranh quyền (preemption), hạ bậc ưu tiên (demotion), và cơ chế chống đói (priority boost). Để phục vụ cho việc mô phỏng, một trình tạo tải công việc (workload generator) sẽ được xây dựng để tạo ra các tác vụ có đặc tính tuân theo mô hình M/M/c.

Một khía cạnh quan trọng khác của dự án là xây dựng một khung đánh giá hiệu năng, cho phép so sánh và đối chiếu các kết quả đo lường từ mô phỏng (ví dụ: thời gian chờ trung bình trong hàng đợi W_q) với các kết quả tính toán từ lý thuyết hàng đợi (ví dụ: Định luật Little). Về mặt kỹ thuật, thiết kế hệ thống sẽ chú trọng đến tính module hóa và khả năng tích hợp tự động, cho phép các thành viên trong nhóm phát triển độc lập thông qua các giao diện lập trình ứng dụng (API) được định nghĩa rõ ràng.

2 Thiết kế hệ thống

Phần này mô tả các dịch vụ cốt lõi mà hệ thống lập lịch MLFQ cung cấp và các chức năng mà mô hình mô phỏng phải thực hiện.

2.1 Các quy tắc MLFQ nền tảng (Dịch vụ Lập lịch)

Bốn quy tắc cơ bản định nghĩa dịch vụ của một trình lập lịch Đa hàng đợi Phản hồi (MLFQ):

Quy tắc 1 (Ưu tiên): Tác vụ ở hàng đợi ưu tiên cao hơn luôn được chọn thực thi trước tác vụ ở hàng đợi ưu tiên thấp hơn.

Quy tắc 2 (Hạ bậc): Nếu một tác vụ sử dụng hết time slice (quantum) được cấp, độ ưu tiên của nó sẽ bị giảm.

Quy tắc 3 (Giữ bậc): Nếu tác vụ tự nguyện nhả CPU trước khi hết time slice, độ ưu tiên của nó được giữ nguyên.

Quy tắc 4 (Chống đói (starvation)): Phải tồn tại một cơ chế "tăng ưu tiên" (priority boost). Định kỳ sau một khoảng thời gian S, tất cả các tác vụ trong hệ thống được di chuyển về hàng đợi ưu tiên cao nhất.

2.2 Kiến trúc và Chức năng Hệ thống (Outcome)

Ứng dụng mô phỏng được cấu trúc thành các cấu phần chức năng chính. Quá trình bắt đầu với việc **Khởi tạo Môi trường**, thiết lập c tài nguyên lõi CPU (SimPy Resources) và ba hàng đợi (Q1, Q2, Q3). Một tiến trình **Tạo Tiến trình** (Workload Generation) chạy độc lập, chịu trách nhiệm sinh ra các tác vụ mới với các thuộc tính `pid`, `arrival_time`, và `total_cpu_time`, sau đó thông báo cho bộ lập lịch.

Dựa trên các cơ chế của hệ thống, một tiến trình khi được xử lý sẽ rơi vào một trong các kết quả sau:

1. **Được xếp hàng (Queued):** Một tiến trình mới đến được xếp vào hàng đợi ưu tiên cao nhất (Q1).

2. **Được thực thi (Executed):** Tiến trình được một lõi xử lý (PE) rảnh chọn để thực thi (khi bộ lập lịch quét từ Q1 → Q2 → Q3).
3. **Hoàn thành (Completed):** Tiến trình chạy xong toàn bộ `total_cpu_time` và rời khỏi hệ thống.
4. **Bị hạ cấp (Demoted):** Tiến trình chạy hết quantum (thời gian cho phép) tại Q1 hoặc Q2 nhưng chưa hoàn thành, và bị di chuyển xuống hàng đợi ưu tiên thấp hơn (Q2 hoặc Q3).
5. **Bị ngắt (Preempted):** Một tiến trình đang chạy ở Q2 hoặc Q3 bị một tiến trình Q1 mới đến ngắt (giành quyền) và phải quay trở lại hàng đợi của mình.
6. **Bị chặn (Blocked / I/O Wait):** Tiến trình tạm dừng thực thi để chờ I/O (theo logic "Thực thi Tiến trình" báo cáo lại).
7. **Được thăng cấp (Boosted):** Một tiến trình đang chờ ở hàng đợi thấp (Q2, Q3) được cơ chế "Chống đói" (Priority Boost) định kỳ di chuyển trở lại Q1.
8. **Tiếp tục chờ (Waits):** Tiến trình vẫn nằm trong hàng đợi (Q1, Q2, hoặc Q3) vì không có lõi PE nào rảnh hoặc tất cả các lõi đang bận thực thi các tiến trình có ưu tiên bằng hoặc cao hơn.

3 Các chỉ số Hiệu năng (Metrics)

Phần này định nghĩa các chỉ số đo lường (metrics) được sử dụng để đánh giá hiệu năng của hệ thống lập lịch, tương ứng với các kết quả (outcomes) mong đợi.

3.1 Bộ thu thập Thống kê (StatisticsCollector)

Một lớp `StatisticsCollector` chuyên dụng chịu trách nhiệm thu thập và tính toán các chỉ số hiệu năng chính của hệ thống.

1. `wait_times`: Danh sách lưu thời gian chờ của các tiến trình (Waiting Time).
2. `turnaround_times`: Danh sách lưu tổng thời gian trong hệ thống của các tiến trình (Turnaround Time).
3. `response_times`: (Mở rộng) Danh sách lưu thời gian phản hồi (Thời điểm chạy lần đầu – Thời điểm đến).
4. `record_wait_time(time)`: Ghi nhận một giá trị thời gian chờ mới.

5. `record_turnaround_time(time)`: Ghi nhận một giá trị turnaround time mới.
6. `calculate_averages()`: Tính toán và trả về các giá trị trung bình (W_q – thời gian chờ trung bình, và W_s – turnaround time trung bình).

4 Tham số và Phương thức

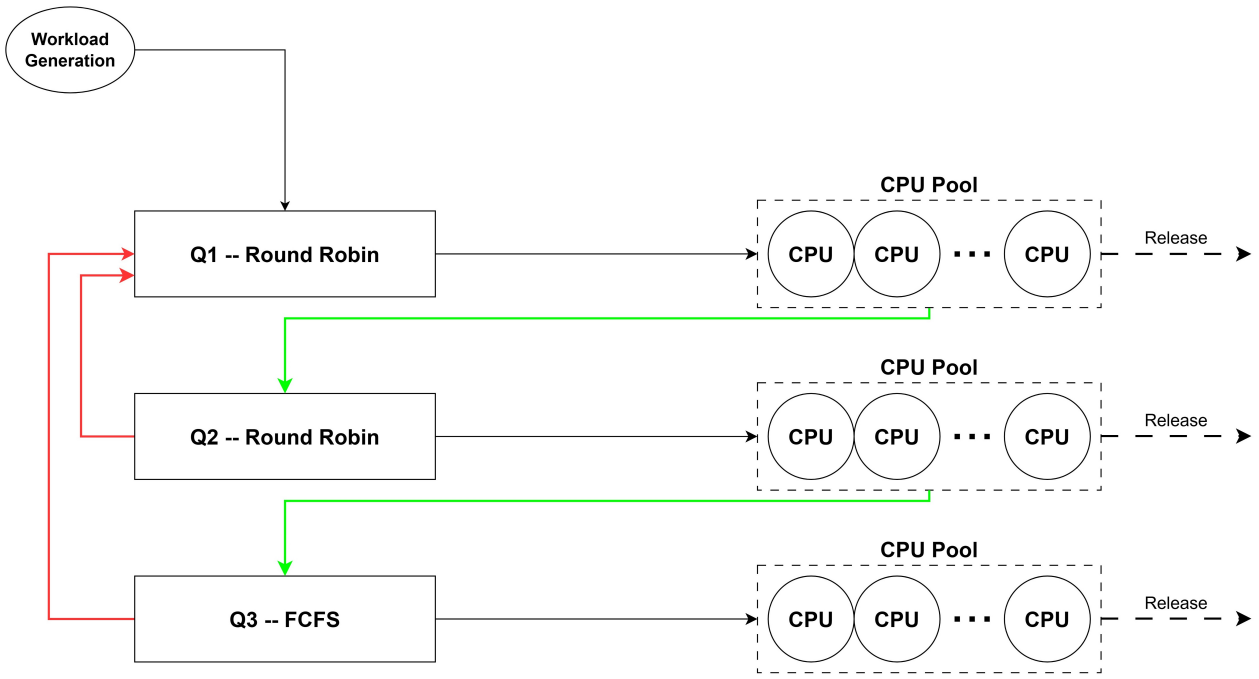
Phần này mô tả phương thức mô phỏng được sử dụng và các tham số (parameters) của hệ thống và của tải công việc (workload).

4.1 Phương thức Mô hình hóa và Điều phối

Phương thức được sử dụng là Mô phỏng Sự kiện Rời rạc (Discrete-Event Simulation) sử dụng thư viện SimPy. Logic hệ thống được trực quan hóa và định nghĩa thông qua các mô hình sau.

4.1.1 Mô hình Kiến trúc Tổng quan

Mô hình kiến trúc tổng quan (Hình 1) mô tả luồng di chuyển của một Tác vụ (Job) qua hệ thống. Tác vụ được tạo ra, đi vào một trong ba hàng đợi (Q1, Q2, Q3) và chờ được cấp phát cho một CPU trong Nhóm CPU (CPU Pool). Nếu một tác vụ được cấp CPU, nó sẽ thực thi cho đến khi hoàn thành, hết quantum, hoặc bị tranh quyền. Sau đó, nó có thể được chuyển đổi giữa các hàng đợi hoặc hoàn tất và rời khỏi hệ thống.



Hình 1: Mô hình kiến trúc tổng quan của hệ thống MLFQ đa nhân.

4.1.2 Máy Trạng thái Tác vụ (FSM)

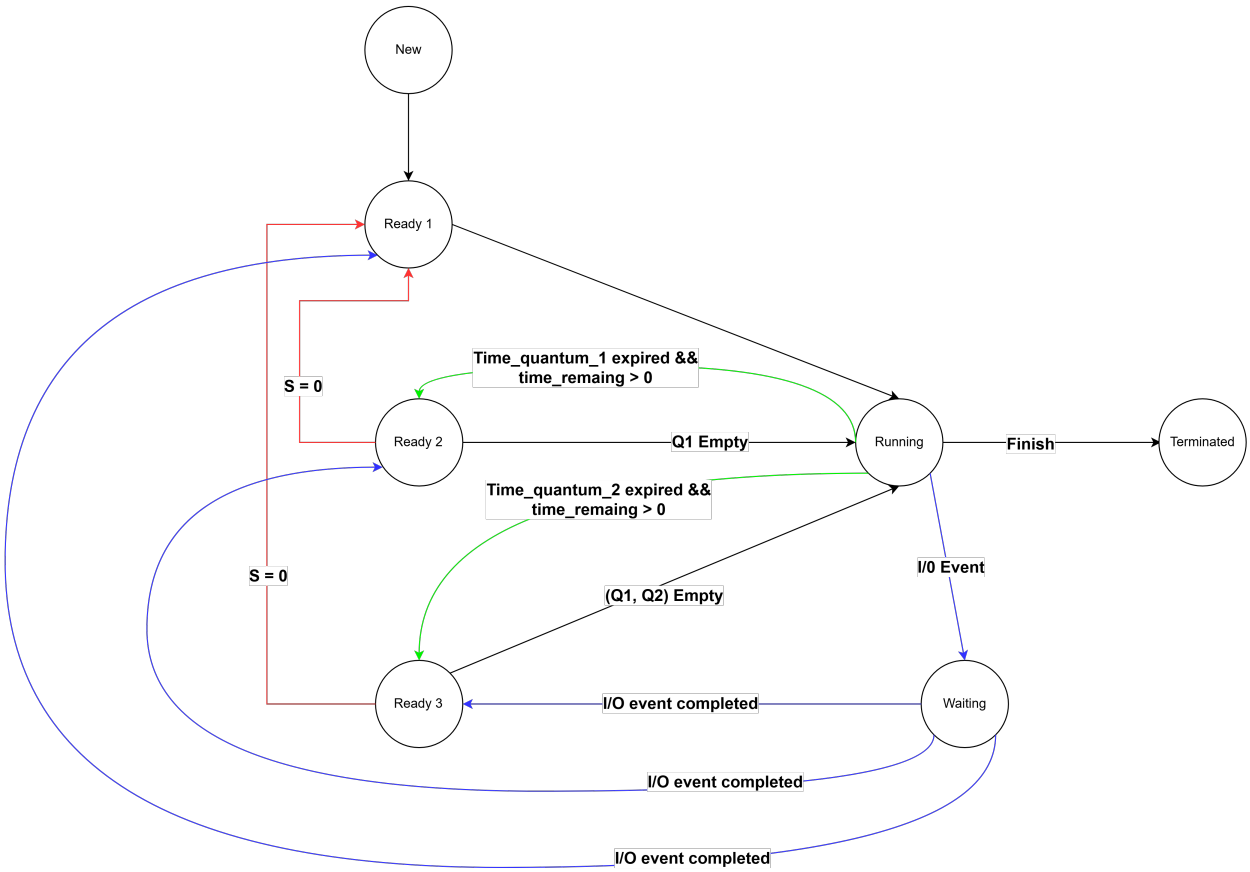
Sơ đồ máy trạng thái (Hình 2) mô tả các trạng thái mà một Tác vụ (Task) có thể trải qua trong hệ thống. Bắt đầu từ trạng thái `New`, một tác vụ được khởi tạo và ngay lập tức đưa vào hàng đợi ưu tiên cao nhất là `Ready 1`.

Từ một trong ba hàng đợi sẵn sàng, tác vụ có thể được chọn để chuyển sang trạng thái `Running`. Việc lựa chọn này tuân theo cơ chế ưu tiên: một tác vụ ở `Ready 2` chỉ được chạy nếu `Ready 1` rỗng (Q1 Empty), và một tác vụ ở `Ready 3` chỉ được chạy khi cả `Ready 1` và `Ready 2` đều rỗng ((Q1, Q2) Empty).

Khi đang ở trạng thái `Running`, có ba khả năng xảy ra:

1. Nếu tác vụ hoàn thành (Finish), nó chuyển sang trạng thái `Terminated`.
2. Nếu tác vụ chạy hết quantum (Time_quantum expired) nhưng vẫn còn thời gian (time remaining > 0), nó sẽ bị hạ cấp ưu tiên và chuyển xuống hàng đợi thấp hơn (hoặc `Ready 2` hoặc `Ready 3`).

Ngoài ra, sơ đồ mô tả một cơ chế "boost" (Priority Boost), biểu thị bằng các đường màu đỏ ($S = 0$). Khi điều kiện này được kích hoạt, mọi tác vụ đang chờ ở `Ready 2` và `Ready 3` sẽ được đưa trở lại hàng đợi `Ready 1` để tránh bị "đói" (starvation).



Hình 2: Máy trạng thái (FSM) mô tả vòng đời của một Tác vụ trong hệ thống.

4.1.3 Biểu đồ Tuần tự (Sequence Diagrams)

Các biểu đồ tuần tự sau đây mô tả chi tiết hai kịch bản quan trọng nhất của Bộ lập lịch.

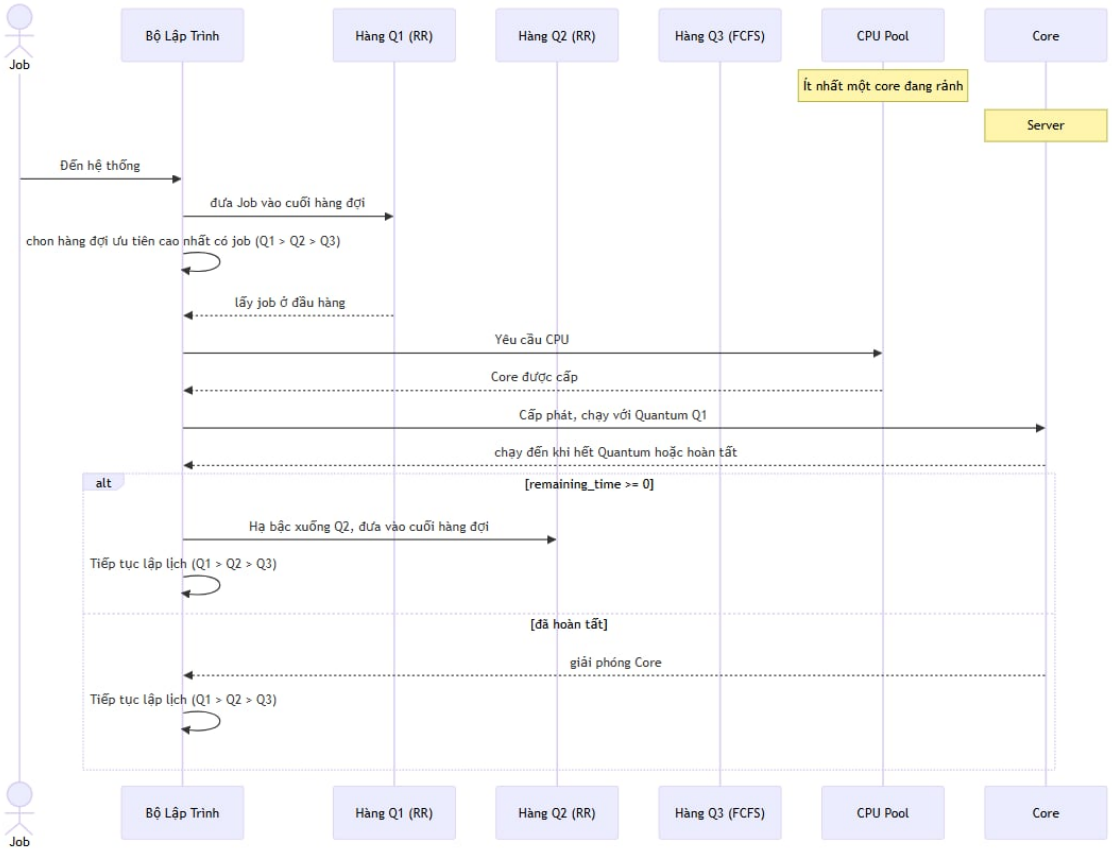
4.1.3.1 Luồng cơ bản (Luôn có Core rảnh)

Biểu đồ tuần tự (Hình 3) mô tả kịch bản lý tưởng khi luôn có ít nhất một core CPU rảnh.

1. Một Job mới đến hệ thống và được đưa vào cuối Hàng đợi 1.
2. Bộ lập lịch (Scheduler) tìm Job ưu tiên cao nhất (Job vừa đến) và yêu cầu CPU từ CPU Pool.
3. CPU Pool xác nhận còn core rảnh và cấp phát cho Job.
4. Job chạy cho đến khi hết Quantum 1 hoặc hoàn tất.
5. Trong kịch bản này (`alt`), Job chạy hết Quantum 1 mà chưa xong, do đó bị hạ

bậc xuống cuối Hàng đợi 2.

6. Quá trình lặp lại, nếu Job hoàn tất, nó sẽ giải phóng Core.



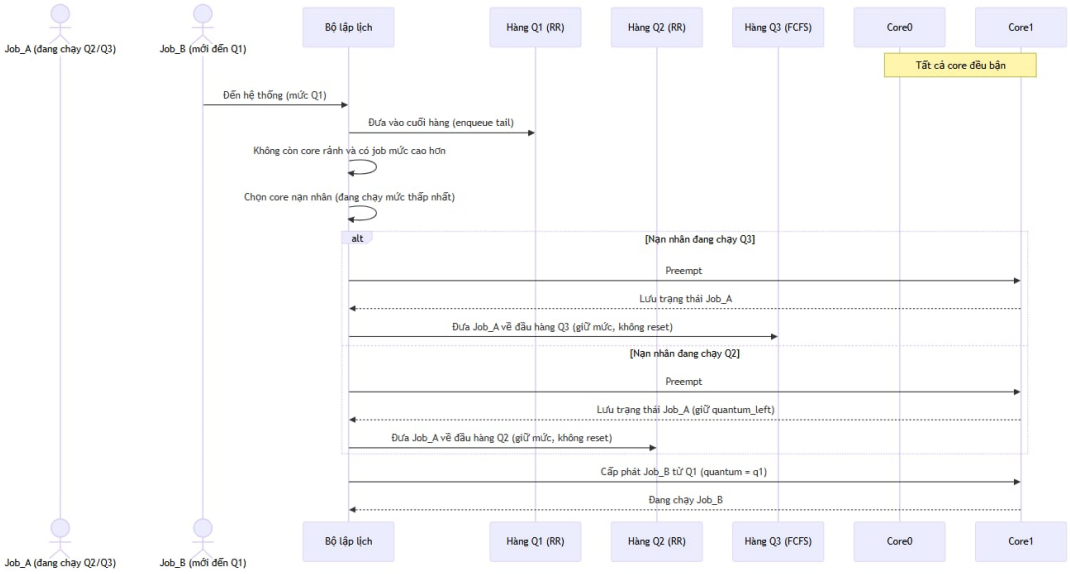
Hình 3: Biểu đồ tuần tự: Vòng đời Tác vụ với ít nhất một core rảnh.

4.1.3.2 Luồng tranh quyền (Hết Core)

Biểu đồ tuần tự (Hình 4) mô tả kịch bản tranh quyền (preemption) phức tạp khi một Job B (mới, ưu tiên cao Q1) đến và tất cả các core đều bận.

1. Một Job B (mới) đến và được đưa vào Hàng đợi 1.
2. Bộ lập lịch phát hiện không còn core rảnh và có một Job A (ưu tiên thấp, Q2 hoặc Q3) đang chạy.
3. Bộ lập lịch chọn "nạn nhân" là core đang chạy Job A (ưu tiên thấp nhất) và gửi tín hiệu `Preempt`.
4. Trạng thái của Job A được lưu lại và được đưa về **đầu** hàng đợi cũ của nó (Q2 hoặc Q3) theo quy tắc 5.

5. Core (hiện đã rảnh) được cấp phát cho Job B để chạy.



Hình 4: Biểu đồ tuần tự: Kịch bản tranh quyền khi tất cả core đều bận.

4.1.4 Quy tắc Điều phối Chi tiết

Logic điều phối chi tiết của hệ thống được triển khai tuân thủ 7 quy tắc vận hành sau:

1. **Cấp phát lõi trống:** Khi có lõi CPU khả dụng, Bộ lập lịch quét (Q1 → Q2 → Q3) và gán lõi cho process đầu tiên được tìm thấy.
2. **Tranh quyền (Preemption):** Khi process Q1 sẵn sàng nhưng tất cả lõi đều bận, hệ thống tranh quyền lõi đang chạy process có ưu tiên thấp nhất (Q3 → Q2).
3. **Hạ bậc ưu tiên (Demotion):** Process sử dụng hết quantum mà chưa hoàn thành sẽ bị hạ xuống hàng đợi thấp hơn (Q1 → Q2; Q2 → Q3) và đặt ở **cuối** hàng đợi mới.
4. **Giữ nguyên ưu tiên (Yielding):** Process tự nguyện nhả CPU (I/O) sẽ giữ nguyên mức ưu tiên và được đặt ở **cuối** hàng đợi hiện tại khi sẵn sàng trở lại.
5. **Xử lý process bị tranh quyền:** Process (Q2, Q3) bị tranh quyền sẽ giữ nguyên trạng thái và được đặt lại ở **đầu** hàng đợi của nó.

6. **Giải quyết xung đột (Conflict Resolution):** Khi nhiều process cùng ưu tiên, thứ tự được quyết định theo thời gian vào hàng đợi.
7. **Chống đói (Priority Boost):** Sau khoảng thời gian S, tất cả process ở Q2 và Q3 được đưa về **cuối** Q1.

4.2 Tham số Hệ thống và Cấu trúc Dữ liệu

Hệ thống mô phỏng MLFQ được xây dựng dựa trên các lớp đối tượng và cấu trúc dữ liệu chính sau đây.

4.2.1 Lớp Process (Biểu diễn Tác vụ)

Lớp `Process` đại diện cho một tiến trình trong hệ thống với các thuộc tính được chia thành hai nhóm:

4.2.1.1 Thuộc tính Tĩnh (Workload Parameters)

Các thuộc tính này được khởi tạo khi tác vụ được tạo ra và không thay đổi:

- `pid`: Định danh duy nhất của tiến trình.
- `arrival_time`: Thời điểm đến hệ thống (tuân theo phân phối Poisson với tham số λ).
- `total_cpu_time`: Tổng thời gian CPU yêu cầu (tuân theo phân phối mũ với tham số μ).

4.2.1.2 Thuộc tính Động (Runtime State)

Các thuộc tính này thay đổi trong quá trình thực thi:

- `current_level`: Mức ưu tiên hiện tại (1, 2, hoặc 3).
- `cpu_left`: Thời gian CPU còn lại cần xử lý.
- `quantum_left`: Thời gian quantum còn lại trong mức ưu tiên hiện tại.
- `time_entered_current_queue`: Thời điểm vào hàng đợi hiện tại (dùng cho Quy tắc 6 – giải quyết xung đột).

4.2.2 Lớp ProcessQueue (Quản lý Hàng đợi)

Lớp `ProcessQueue` cung cấp cấu trúc dữ liệu cho mỗi hàng đợi ưu tiên (Q1, Q2, Q3) với các phương thức chính:

- `add_to_end(p)`: Thêm tiến trình `p` vào **cuối** hàng đợi (áp dụng cho Quy tắc 3, 4, 7).
- `add_to_head(p)`: Thêm tiến trình `p` vào **đầu** hàng đợi (áp dụng cho Quy tắc 5 – xử lý tranh quyền).
- `remove_from_head()`: Lấy và loại bỏ tiến trình ở đầu hàng đợi (áp dụng cho Quy tắc 1 – cấp phát CPU).
- `is_empty()`: Kiểm tra hàng đợi có rỗng hay không.

4.2.3 Lớp Core (Biểu diễn Lõi CPU)

Lớp `Core` đại diện cho một lõi CPU vật lý với các thành phần:

- `scheduler`: Tham chiếu đến bộ lập lịch trung tâm (`CPUScheduler`).
- `current_process`: Tiến trình hiện đang được thực thi trên lõi này.
- `run_loop()`: Vòng lặp mô phỏng quá trình thực thi tiến trình.
- `preempt()`: Phương thức ngắt lõi để tranh quyền (áp dụng cho Quy tắc 2).

4.2.4 Lớp CPUScheduler (Bộ Lập lịch Trung tâm)

Lớp `CPUScheduler` là thành phần cốt lõi của hệ thống, chịu trách nhiệm điều phối và quản lý tất cả các hoạt động lập lịch.

4.2.4.1 Tham số Hệ thống

- `NUM_CPUS` (hoặc c): Số lượng lõi CPU trong hệ thống.
- `S_PERIOD`: Chu kỳ Priority Boost (tham số S trong Quy tắc 7).
- q_1, q_2: Thời gian quantum cho hàng đợi Q1 và Q2 (Quy tắc 3).

4.2.4.2 Thuộc tính Nội bộ

- `env`: Môi trường mô phỏng SimPy.
- `cpu`: Tài nguyên CPU SimPy với dung lượng c.

- `queues`: Mảng 3 đối tượng `ProcessQueue` (Q1, Q2, Q3).
- `stats`: Đối tượng `StatisticsCollector` để thu thập metrics.
- `all_processes`: Danh sách theo dõi tất cả tiến trình trong hệ thống (sử dụng cho Quy tắc 7).

4.2.4.3 Phương thức Xử lý Sự kiện

Bộ lập lịch hoạt động theo mô hình hướng sự kiện (event-driven) với các phương thức chính:

- `add_process(process)`: Thêm tiến trình mới vào Q1 và đăng ký vào `all_processes`.
- `run_scheduler()`: Vòng lặp điều phối chính, quyết định tiến trình nào được thực thi tiếp theo.
- `execute_process(process, queue_level)`: Thực thi tiến trình và xử lý logic hạ bậc. Xóa tiến trình khỏi `all_processes` khi hoàn thành.
- `priority_booster()`: Tiến trình song song kích hoạt mỗi `S_PERIOD` giây để đưa tất cả tiến trình trong `all_processes` về Q1.
- `on_process_arrival(p)`: Xử lý sự kiện tiến trình mới đến hệ thống.
- `on_process_yield(p)`: Xử lý sự kiện tiến trình tự nguyện nhả CPU (Quy tắc 4).
- `on_process_complete(p)`: Xử lý sự kiện tiến trình hoàn thành.
- `on_quantum_expired(p)`: Xử lý sự kiện tiến trình hết quantum (Quy tắc 3).
- `request_next_process(core)`: Được gọi bởi lõi CPU rảnh để yêu cầu tiến trình tiếp theo (Quy tắc 1).

5 Mô hình hóa và Phân tích Hiệu năng

Chương này trình bày nền tảng lý thuyết và các công thức toán học được sử dụng để phân tích và xác thực hiệu năng của hệ thống MLFQ đa nhân. Mục tiêu là liên kết các tham số của mô hình M/M/c với các số liệu hiệu năng đo lường được từ mô phỏng, đặc biệt là thông qua Định luật Little.

5.1 Mô hình Hóa Hệ thống (M/M/c)

Để thiết lập một nền tảng phân tích định lượng, hệ thống lập lịch CPU đa nhân được mô hình hóa như một hệ thống hàng đợi kinh điển M/M/c:

- **M (Markovian Arrivals):** Luồng các tiến trình đến hệ thống tuân theo phân phối Poisson với tốc độ đến trung bình là λ. Thời gian giữa hai lần đến liên tiếp tuân theo phân phối mũ.
- **M (Markovian Services):** Thời gian CPU cần thiết để hoàn thành một tiến trình tuân theo phân phối mũ. Tốc độ phục vụ trung bình của mỗi lõi CPU là μ, do đó, thời gian phục vụ trung bình là $1/\mu$.
- **c (Servers):** Hệ thống bao gồm c lõi CPU hoạt động song song ($c \geq 2$).

5.1.1 Ký hiệu và Định nghĩa

Từ các định nghĩa này, tham số cơ bản để đánh giá là **cường độ lưu thông** (ρ), đại diện cho tải trung bình trên mỗi bộ xử lý:

$$\rho = \frac{\lambda}{c\mu}$$

Điều kiện để hệ thống hoạt động ổn định là $\lambda < c\mu \Rightarrow \rho < 1$.

Bảng dưới đây tóm tắt các ký hiệu và định nghĩa quan trọng được sử dụng:

Ký hiệu	Mô tả	Đơn vị
λ	Tốc độ đến trung bình của các tiến trình vào hệ thống	tiến trình/giây
μ	Tốc độ phục vụ trung bình của một lõi CPU	tiến trình/giây
c	Số lượng lõi CPU trong hệ thống	-
ρ	Cường độ lưu thông (hệ số sử dụng trung bình của mỗi lõi)	-
P_n	Xác suất ổn định của việc có n tiến trình trong hệ thống	-
L_s	Số lượng tiến trình trung bình trong toàn bộ hệ thống	tiến trình
L_q	Số lượng tiến trình trung bình đang chờ trong hàng đợi	tiến trình
W_s	Thời gian trung bình một tiến trình ở trong hệ thống	giây
W_q	Thời gian trung bình một tiến trình chờ trong hàng đợi	giây
Q_i	Hàng đợi ưu tiên cấp i ($i = 1, 2, 3$)	-
q_i	Lượng tử thời gian (time quantum) của hàng đợi Q_i	giây
S	Chu kỳ tăng ưu tiên (priority boost period)	giây
n_i	Số lượng tiến trình trong hàng đợi Q_i	tiến trình
λ_i^{eff}	Tốc độ đến hiệu dụng vào hàng đợi Q_i	tiến trình/giây

Bảng 1: Bảng ký hiệu và định nghĩa

5.2 Các Tiêu chuẩn Đánh giá Hiệu năng Hàng đợi

Khi hệ thống đạt **trạng thái ổn định**, chúng ta có thể tính toán các chỉ số hiệu năng trung bình.

5.2.1 Xác suất Trạng thái

Xác suất trạng thái (P_n) là xác suất có n tiến trình trong hệ thống.

Xác suất hệ thống hoàn toàn rỗi ($n = 0$):

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^c}{c!} \frac{1}{1-\rho} \right]^{-1}$$

Thành phần đầu tiên $\sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!}$ tính tổng xác suất tương đối của các trạng thái có từ 0 đến $c-1$ tiến trình trong hệ thống, tức là khi chưa có hàng đợi chờ.

$$P_n = \frac{(\lambda/\mu)^n}{n!} P_0$$

Thành phần thứ hai $\frac{(\lambda/\mu)^c}{c!}\frac{1}{1-\rho}$ đại diện cho tổng xác suất tương đối khi có c hoặc nhiều hơn c tiến trình trong hệ thống. Ta có:

$$\sum_{n=c}^{\infty} \frac{P_n}{P_0} = \sum_{n=c}^{\infty} \frac{(\lambda/\mu)^n}{c! \cdot c^{n-c}} = \frac{(\lambda/\mu)^c}{c!} \sum_{n=c}^{\infty} \left(\frac{\lambda/\mu}{c}\right)^{n-c} = \frac{(\lambda/\mu)^c}{c!} \cdot \frac{1}{1-\rho}$$

5.2.2 Số lượng Tiến trình Trung bình

Số lượng tiến trình trung bình đang chờ trong hàng đợi (L_q) tại trạng thái có n tiến trình trong hệ thống (với $n \geq c$) là $(n-c)$. Do đó:

$$L_q = \sum_{n=c}^{\infty} (n-c) \cdot P_n$$

Từ lý thuyết hàng đợi M/M/c, khi $n \geq c$, xác suất trạng thái có thể biểu diễn bằng $P_n = P_c \cdot \rho^{(n-c)}$,trong đó $P_c = \frac{P_0(\lambda/\mu)^c}{c!}$ là xác suất có đúng c tiến trình trong hệ thống.

Thay thế vào công thức L_q:

$$L_q = \sum_{n=c}^{\infty} (n-c) \cdot P_c \cdot \rho^{(n-c)}$$

Đặt $k = n - c$, khi đó $n = k + c$ và khi n chạy từ c đến ∞ thì k chạy từ 0 đến ∞:

$$L_q = P_c \sum_{k=0}^{\infty} k \cdot \rho^k$$

Sử dụng công thức tổng chuỗi hình học: $\sum_{k=0}^{\infty} k\rho^k = \frac{\rho}{(1-\rho)^2}$ (với $\rho < 1$), ta có:

$$L_q = P_c \cdot \frac{\rho}{(1-\rho)^2} = \left[\frac{P_0(\lambda/\mu)^c}{c!}\right] \cdot \frac{\rho}{(1-\rho)^2} = \frac{P_0(\lambda/\mu)^c \rho}{c!(1-\rho)^2}$$

Số lượng tiến trình trung bình trong toàn bộ hệ thống (L_s):

$$L_s = L_q + \frac{\lambda}{\mu}$$

5.2.3 Thời gian Trung bình

Áp dụng Định luật Little ($L = \lambda W$):

Thời gian chờ trung bình trong hàng đợi (W_q):

$$W_q = \frac{L_q}{\lambda}$$

Thời gian trung bình một tiến trình ở trong hệ thống (W_s):

$$W_s = \frac{L_s}{\lambda} = \frac{L_q}{\lambda} + \frac{1}{\mu} = W_q + \frac{1}{\mu}$$

5.2.4 Xác thực Tính nhất quán của Mô hình (Sử dụng Định luật Little)

Tóm lại, việc mô hình hóa hệ thống dưới dạng hàng đợi M/M/c đã cung cấp một nền tảng phân tích định lượng hiệu quả. Định luật Little ($L = \lambda W$) đóng vai trò then chốt, tạo ra cầu nối toán học thiết yếu, cho phép suy ra các chỉ số hiệu năng dựa trên thời gian (W_q, W_s) từ các chỉ số dựa trên số lượng (L_q, L_s). Mối quan hệ $W_s = W_q + 1/\mu$, được suy ra trực tiếp từ định luật này, đã xác thực tính nhất quán nội tại của mô hình. Khuôn khổ này không chỉ giúp diễn giải kết quả mô phỏng mà còn cung cấp một công cụ mạnh mẽ để dự đoán thời gian phản hồi trung bình của hệ thống.

Tuy nhiên, do cơ chế ưu tiên tuyệt đối của Q_1, nếu có một luồng tiến trình ngắn hoặc các tiến trình được "boost" lên Q_1 với tốc độ hiệu dụng lớn hơn khả năng xử lý của hệ thống ($\lambda_1^{eff} > c\mu$), thì các lõi CPU sẽ chỉ bận rộn xử lý các tiến trình Q_1.

Điều này sẽ khiến các tiến trình ở Q_2 và Q_3 bị **đói** tài nguyên (starvation), và hàng đợi của chúng sẽ tăng lên ngay cả khi hệ thống tổng thể chưa bị quá tải. Do đó, một phân tích ổn định hoàn chỉnh cho MLFQ phải xem xét cả:

1. Sự ổn định toàn cục của hệ thống
2. Sự ổn định "cục bộ" của các hàng đợi ưu tiên cao

5.3 Mô Hình Hóa Trạng Thái Hệ Thống MLFQ bằng Chuỗi Markov

Để phân tích sâu hơn các động lực bên trong của bộ lập lịch MLFQ, chúng ta sử dụng mô hình Chuỗi Markov thời gian liên tục (CTMC). Mô hình này cho phép định

lượng tác động của các tham số như lượng tử thời gian (q_1, q_2) và chu kỳ tăng ưu tiên (S).

Không gian Trạng thái: Một trạng thái của hệ thống được định nghĩa bằng vector $S = (n_1, n_2, n_3)$, trong đó n_i là số lượng tiến trình trong hàng đợi Q_i.

Hệ phương trình Cân bằng: Ở trạng thái ổn định, xác suất $P(S)$ của mỗi trạng thái là không đổi. Nguyên lý cân bằng phát biểu rằng: tổng tốc độ dòng xác suất đi vào một trạng thái phải bằng tổng tốc độ dòng xác suất đi ra khỏi trạng thái đó ($\text{Rate}_{\text{in}} = \text{Rate}_{\text{out}}$).

Do chính sách ưu tiên của MLFQ, c lõi CPU được phân bổ tuần tự. Số lõi k_i hoạt động phục vụ mỗi hàng đợi tại trạng thái S là $k_1 = \min(n_1, c)$, $k_2 = \min(n_2, c - k_1)$, $k_3 = \min(n_3, c - k_1 - k_2)$.

Các sự kiện hệ thống làm thay đổi trạng thái $S = (n_1, n_2, n_3)$ với các tốc độ (rates) sau:

1. **Đến (Arrival):** Một tiến trình mới đến hệ thống với tốc độ λ được đặt vào Q_1. Điều này gây ra sự chuyển trạng thái thành $(n_1 + 1, n_2, n_3)$.
2. **Hoàn thành (Completion):** Một tiến trình hoàn thành và rời hệ thống.
   - Từ Q_1: Tốc độ $k_1 \cdot \mu$. Chuyển đến $(n_1 - 1, n_2, n_3)$ (nếu $n_1 > 0$).
   - Từ Q_2: Tốc độ $k_2 \cdot \mu$. Chuyển đến $(n_1, n_2 - 1, n_3)$ (nếu $n_1 = 0, n_2 > 0$).
   - Từ Q_3: Tốc độ $k_3 \cdot \mu$. Chuyển đến $(n_1, n_2, n_3 - 1)$ (nếu $n_1 = 0, n_2 = 0, n_3 > 0$).
3. **Hạ bậc (Demotion):** Một tiến trình chạy hết quantum q_i và bị hạ cấp.
   - Từ $Q_1 \to Q_2$: Tốc độ $k_1 \cdot (1/q_1)$. Chuyển đến $(n_1 - 1, n_2 + 1, n_3)$ (nếu $n_1 > 0$).
   - Từ $Q_2 \to Q_3$: Tốc độ $k_2 \cdot (1/q_2)$. Chuyển đến $(n_1, n_2 - 1, n_3 + 1)$ (nếu $n_1 = 0, n_2 > 0$).
4. **Tăng ưu tiên (Priority Boost):** Tất cả tiến trình ở Q_2, Q_3 được chuyển về Q_1 sau chu kỳ S.
   - Tốc độ: $1/S$
   - Chuyển trạng thái: $(n_1 + n_2 + n_3, 0, 0)$ (nếu $n_2 > 0$ hoặc $n_3 > 0$).

Tổng tốc độ dòng chảy ra khỏi trạng thái $S = (n_1, n_2, n_3)$:

$$\begin{aligned}\text{Rate}_{\text{out}}(S) = \lambda \quad &\text{(Arrival)} \\ &+ k_1\mu \cdot \mathbf{1}_{n_1>0} \quad \text{(Completion from } Q_1\text{)} \\ &+ k_2\mu \cdot \mathbf{1}_{n_1=0,n_2>0} \quad \text{(Completion from } Q_2\text{)} \\ &+ k_3\mu \cdot \mathbf{1}_{n_1=0,n_2=0,n_3>0} \quad \text{(Completion from } Q_3\text{)} \\ &+ k_1 \cdot \frac{1}{q_1} \cdot \mathbf{1}_{n_1>0} \quad \text{(Demotion from } Q_1\text{)} \\ &+ k_2 \cdot \frac{1}{q_2} \cdot \mathbf{1}_{n_1=0,n_2>0} \quad \text{(Demotion from } Q_2\text{)} \\ &+ \frac{1}{S} \cdot \mathbf{1}_{n_2>0 \vee n_3>0} \quad \text{(Priority Boost)} \end{aligned} \tag{1}$$

(trong đó $\mathbf{1}_{\text{condition}}$ là hàm chỉ thị, bằng 1 nếu điều kiện đúng và 0 nếu sai)

Phương trình cân bằng tổng quát ($\mathbf{Rate_{out}} \cdot P(S) = \mathbf{Rate_{in}}$):

$$\begin{aligned}\text{Rate}_{\text{out}}(S) \cdot P(S) = \lambda \cdot P(n_1 - 1, n_2, n_3) \cdot \mathbf{1}_{n_1>0} \\ &+ \mu \cdot \min(n_1 + 1, c) \cdot P(n_1 + 1, n_2, n_3) \\ &+ \mu \cdot \min(n_2 + 1, c) \cdot P(n_1, n_2 + 1, n_3) \cdot \mathbf{1}_{n_1=0} \\ &+ \mu \cdot \min(n_3 + 1, c) \cdot P(n_1, n_2, n_3 + 1) \cdot \mathbf{1}_{n_1=0,\, n_2=0} \\ &+ \frac{1}{q_1} \cdot \min(n_1 + 1, c) \cdot P(n_1 + 1, n_2 - 1, n_3) \cdot \mathbf{1}_{n_2>0} \\ &+ \frac{1}{q_2} \cdot \min(n_2 + 1, c) \cdot P(n_1, n_2 + 1, n_3 - 1) \cdot \mathbf{1}_{n_1=0,\, n_3>0} \\ &+ \frac{1}{S} \sum_{\substack{n_1'+n_2'+n_3'=n_1 \\ n_2'+n_3'>0}} P(n_1', n_2', n_3') \cdot \mathbf{1}_{n_2=0,\, n_3=0}\end{aligned} \tag{2}$$

Việc giải hệ phương trình tuyến tính này cùng với điều kiện chuẩn hóa $\sum_S P(S) = 1$ sẽ cho ra phân phối xác suất ổn định $P(S)$.

5.4 Phân tích Hiệu năng và Đánh đổi Tham số

Khi đã có phân phối $P(S)$, các chỉ số hiệu năng có thể được tính toán để định lượng tác động của các tham số q_1, q_2, S.

5.4.1 Các chỉ số hiệu năng chính

Số lượng chờ trung bình (L_{q_i}):

$$L_{q_i} = \sum_{S} n_i \cdot P(n_1, n_2, n_3) \quad (\text{với } i = 1, 2, 3)$$

Tốc độ đến hiệu dụng (λ_i^{eff}): Tốc độ thực tế các tiến trình đi vào mỗi hàng đợi, bao gồm cả luồng nội bộ (hạ bậc, tăng ưu tiên).

$$\lambda_1^{\text{eff}} = \lambda + \frac{1}{S} \sum_{S: n_2 > 0 \vee n_3 > 0} (n_2 + n_3) \cdot P(S)$$

$$\lambda_2^{\text{eff}} = \sum_{S: n_1 > 0} \left(k_1 \cdot \frac{1}{q_1} \right) \cdot P(S)$$

$$\lambda_3^{\text{eff}} = \sum_{S: n_1 = 0, n_2 > 0} \left(k_2 \cdot \frac{1}{q_2} \right) \cdot P(S)$$

Thời gian chờ trung bình (W_{q_i}): Áp dụng Định luật Little cho từng hàng đợi con:

$$W_{q_i} = \frac{L_{q_i}}{\lambda_i^{\text{eff}}}$$

5.4.2 Phân tích tác động tham số

Phân tích W_{q_1} (Hàng đợi Ưu tiên Cao nhất):

Thời gian chờ W_{q_1} được giữ ở mức thấp do hai yếu tố thể hiện trong mô hình:

- **Ưu tiên tuyệt đối:** Tốc độ phục vụ $k_1 \cdot \mu$ của Q_1 (với $k_1 = \min(n_1, c)$) chỉ phụ thuộc vào số lượng tiến trình trong Q_1, độc lập với n_2 và n_3. Q_1 luôn được ưu tiên cấp phát lõi CPU trước.
- **Lọc tiến trình CPU-bound:** Các tiến trình CPU-bound sẽ nhanh chóng bị đẩy ra khỏi Q_1 thông qua cơ chế hạ bậc (tốc độ k_1/q_1), giữ cho L_{q_1} ở mức thấp.

Phân tích W_{q_2} (Hàng đợi Trung bình):

Thời gian chờ W_{q_2} phức tạp hơn đáng kể. Từ công thức $W_{q_2} = L_{q_2}/\lambda_2^{\text{eff}}$, ta thấy:

- **Tốc độ phục vụ bị ảnh hưởng:** Tốc độ phục vụ $k_2 \cdot \mu$ (với $k_2 = \min(n_2, c - k_1)$) bị giới hạn bởi số lõi còn lại sau khi Q_1 đã lấy. Nếu Q_1 bận ($n_1 \geq c$), k_2 sẽ bằng 0, khiến Q_2 bị "đóng băng".

- **Tốc độ đến không ổn định:** λ_2^{eff} phụ thuộc trực tiếp vào q_1 (tốc độ hạ bậc) và $P(S)$ (tải của Q_1).

Đánh đổi Tham số S (Priority Boost):

Tham số S kiểm soát tần suất của việc tăng ưu tiên, ảnh hưởng đến cả W_{q_1} và W_{q_2}, W_{q_3}.

Khi S giảm (tăng tần suất boost):

- **Tác động lên Q_1:** Nhìn vào phương trình λ_1^{eff}, hạng tử $\frac{1}{S}\sum(n_2+n_3)P(S)$ tăng lên. Điều này làm tăng tổng tốc độ đến hiệu dụng của Q_1, dẫn đến tăng L_{q_1} và có khả năng làm tăng W_{q_1}.
- **Tác động lên Q_2, Q_3:** Việc "làm sạch" Q_2, Q_3 thường xuyên hơn làm giảm xác suất $P(S)$ của các trạng thái có n_2, n_3 lớn. Điều này trực tiếp làm giảm L_{q_2} và L_{q_3}, giúp giảm W_{q_2}, W_{q_3} và chống đói hiệu quả.

Đánh đổi Tham số q_1 (Quantum Q_1):

Tham số q_1 kiểm soát sự cân bằng giữa W_{q_1} và W_{q_2}.

Khi q_1 giảm (quantum ngắn):

- **Tác động lên Q_2:** Tốc độ hạ bậc $(1/q_1)$ tăng lên. Nhìn vào phương trình λ_2^{eff}, tốc độ đến Q_2 tăng lên. Điều này làm tăng L_{q_2} và do đó làm tăng W_{q_2}.
- **Tác động lên Q_1:** Lợi ích là các tác vụ CPU-bound bị lọc ra khỏi Q_1 nhanh hơn, giúp L_{q_1} giữ ở mức thấp và duy trì W_{q_1} thấp (tốt cho tác vụ tương tác).

Khi q_1 tăng (quantum dài): Tác động ngược lại xảy ra. λ_2^{eff} giảm, giúp giảm W_{q_2}, nhưng các tác vụ CPU-bound ở lại Q_1 lâu hơn, làm tăng L_{q_1} và làm tăng W_{q_1}.

5.5 Giới hạn Lý thuyết và Điều kiện Ổn định

Mô hình này cũng cho thấy các giới hạn của hệ thống:

Điều kiện Ổn định Phức hợp: Để hệ thống ổn định, tải trên hàng đợi ưu tiên cao nhất phải nhỏ hơn khả năng phục vụ tổng:

$$\lambda_1^{\text{eff}} < c\mu$$

Nếu $\lambda_1^{\text{eff}} \geq c\mu$, Q_1 sẽ bão hòa, dẫn đến đói tài nguyên cho Q_2 và Q_3.

Kiểm tra Thông lượng (Throughput): Ở trạng thái ổn định, thông lượng đầu ra X (tổng tốc độ hoàn thành) phải bằng tốc độ đầu vào λ.

$$X = \sum_{S} (k_1\mu \cdot \mathbf{1}_{n_1>0} + k_2\mu \cdot \mathbf{1}_{n_1=0,n_2>0} + k_3\mu \cdot \mathbf{1}_{n_1=0,n_2=0,n_3>0}) \cdot P(S) = \lambda \tag{3}$$