✅ 1. Thu thập dữ liệu hệ thống (Data Collection – Logging Phase)

🎯 Mục tiêu:

Ghi lại hành vi hệ thống để tạo dữ liệu huấn luyện cho mô hình offline.

📦 Dữ liệu cần thu thập:

- CPU usage
- Memory usage
- Process priority
- IO wait time
- Context switches
- Scheduling decisions (nếu có)

🛠️ Công cụ đề xuất:

- `perf`, `dstat`, `vmstat`, `iotop`, `pidstat`, `top`
- Tự ghi log bằng script Python/Bash hoặc `eBPF` tracing

🧠 Lưu ý:

- Đảm bảo các log có timestamp, tiến trình liên quan và nhãn "đúng" (nếu dùng supervised learning).
- Có thể bắt đầu từ việc chạy hệ thống Linux thật, sinh dữ liệu từ các workload benchmark (vd: `stress-ng`, `sysbench`).

✅ 2. Huấn luyện mô hình Machine Learning offline (Offline ML Training)

🎯 Mục tiêu:

Dựa trên dữ liệu log ở bước 1, huấn luyện một mô hình dự đoán chính sách hệ thống (vd: phân loại tiến trình cần ưu tiên CPU).

📌 Mô hình đề xuất:

- **Gradient Boosting** (XGBoost, LightGBM)
- Hoặc **Random Forest**, **Decision Tree**

🛠️ Công cụ:

- `scikit-learn`, `xgboost`, `lightgbm`, `pandas`, `matplotlib`

🧠 Lưu ý:

- Dùng mô hình này như **baseline policy**
- Mô hình phải nhẹ và dễ chuyển thành inference trong C/C++ nếu muốn tích hợp kernel module (hoặc convert sang ONNX/TFLite)

✅ 3. Tạo môi trường tương tác cho Reinforcement Learning (RL Environment)

🎯 Mục tiêu:

Xây dựng môi trường mô phỏng hệ thống (hoặc thật) để RL agent có thể "chơi thử", học cách ra quyết định tối ưu.

🏗️ Nội dung cần xây dựng:

- **Observation**: [CPU usage, RAM usage, Priority,...]
- **Action**: thay đổi scheduler (ưu tiên tiến trình, preempt, delay...)
- **Reward**: throughput cao, latency thấp, CPU idle ít → càng tốt

🛠️ Công cụ:

- `OpenAI Gym` (hoặc `gymnasium`)
- Viết `CPUSchedulerEnv` custom class
- Mô phỏng nhỏ nếu chưa gắn thật vào kernel

🧠 Lưu ý:

- Reward function là chìa khóa → bạn phải thiết kế nó phản ánh mục tiêu hệ thống (hiệu suất, công bằng, tiết kiệm điện...)
- Có thể khởi tạo RL agent với policy từ bước 2 (warm start)

✅ 4. Chạy RL agent và tối ưu chính sách (RL Training & Fine-tuning)

🎯 Mục tiêu:

Agent học cách tối ưu lịch trình hoặc phân bổ tài nguyên thông qua tương tác.

📌 Thuật toán RL nên dùng:

- **PPO** (Proximal Policy Optimization)
- **DQN** (Deep Q-Network)
- **A2C / A3C** (nếu cần song song hóa)

🛠️ Công cụ:

- `stable-baselines3` (Python)
- `ray[rllib]` nếu muốn huấn luyện phân tán
- `torch` hoặc `tensorflow` để build custom agent

🧠 Lưu ý:

- Có thể dùng mô hình từ bước 2 như π_0 (baseline policy), rồi RL cải tiến dần
- Mô hình sau khi huấn luyện có thể:
  - Dùng trong môi trường ảo để benchmark
  - Chuyển sang mô-đun kernel (ONNX, C/C++, hoặc TinyML)

🧩 Tổng quan pipeline

```
text
CopyEdit
[Workload or Simulated System]
        ↓
 [Log Telemetry Data (Step 1)]
        ↓
 [Train Gradient Boosting Model (Step 2)] → π₀
        ↓
 [Design RL Env (Step 3)] + π₀
        ↓
 [Train RL Agent (Step 4)]
        ↓
 [Deploy into kernel or monitor system impact]
```