Chương 4:

Câu 1:

+ **Định nghĩa và phân biệt: Phối hợp vs. Đồng bộ trong hệ thống phân tán**

Phối hợp (Coordination):

Phối hợp trong hệ thống phân tán là quá trình các tiến trình hoặc nút (node) trong hệ thống tương tác, điều khiển hoặc sắp xếp hành động với nhau để đạt được một mục tiêu chung. Ví dụ: phân quyền truy cập tài nguyên, phân phối công việc, đảm bảo không có hai tiến trình cùng truy cập vào vùng nhớ chia sẻ.

Đồng bộ (Synchronization):

Đồng bộ trong hệ thống phân tán là việc duy trì tính nhất quán về thời gian hoặc trạng thái giữa các tiến trình hoặc nút. Trong đó, một vấn đề điển hình là đồng bộ hóa đồng hồ giữa các máy để đảm bảo mọi hoạt động có thể được sắp xếp theo thứ tự thời gian đúng.

Sự khác nhau chính:

Tiêu chí	Phối hợp (Coordination)	Đồng bộ (Synchronization)
Mục tiêu	Điều phối hoạt động để đạt được mục tiêu chung	Đồng bộ hóa thời gian/trạng thái giữa các thành phần
Phạm vi	Rộng: bao gồm giao tiếp, chia sẻ tài nguyên	Hẹp hơn, thường liên quan đến thời gian hoặc tiến trình
Tác động	Đảm bảo hệ thống hoạt động logic, tránh xung đột	Đảm bảo thứ tự và độ chính xác về thời gian hoặc dữ liệu
Ví dụ	Hệ thống phân quyền truy cập tài nguyên	Giao thức NTP để đồng bộ đồng hồ giữa các máy

Tầm quan trọng của việc đồng bộ hóa thời gian

Trong hệ thống phân tán, mỗi nút có thể có đồng hồ riêng biệt và không đồng bộ với nhau. Nếu không có cơ chế đồng bộ hóa thời gian, hệ thống sẽ không thể xác định chính xác thứ tự các sự kiện, gây ra lỗi hoặc hành vi không mong muốn.

Ví dụ minh họa:

Hai lập trình viên cùng làm việc trên một tệp mã nguồn trên hệ thống phân tán:

- Giả sử:
    - Lập trình viên A làm việc trên máy tính A.
    - Lập trình viên B làm việc trên máy tính B.

- Cả hai đều chỉnh sửa và lưu file main.cpp trên một máy chủ trung gian (shared file server).
- Đồng hồ hệ thống trên máy A chậm hơn 5 phút so với máy B.

Tình huống lỗi:

1. Lập trình viên A lưu file vào 10:00 AM (theo giờ máy A).
2. Lập trình viên B lưu file vào 10:02 AM (theo giờ máy B).
3. Máy chủ kiểm tra timestamp:
   - File từ máy A có timestamp: 10:00
   - File từ máy B có timestamp: 10:02
   - => Máy chủ giữ lại file của B (vì mới hơn).

Tuy nhiên, nếu thực tế:

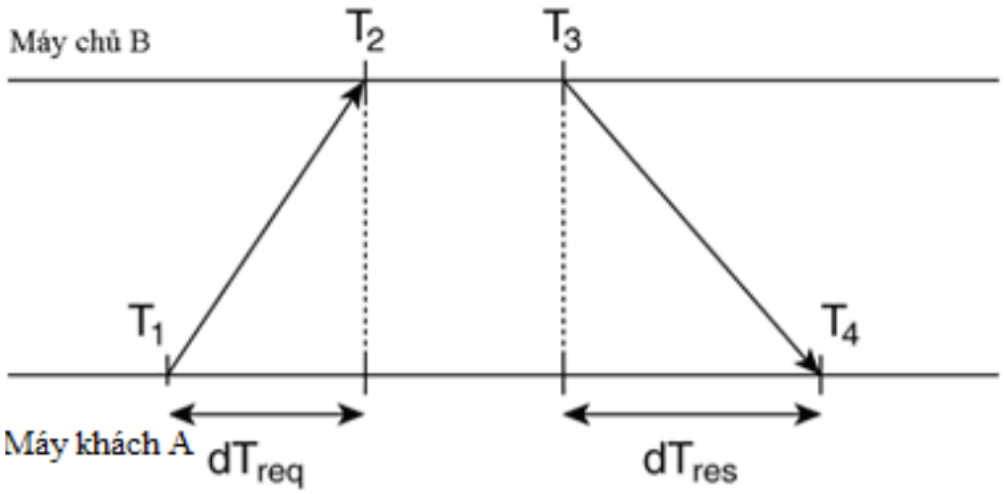
- Thời điểm thực sự của A là 10:05 (theo giờ thực tế).
- Thời điểm thực sự của B là 10:02.
  => File của A mới hơn nhưng bị ghi đè mất vì hệ thống hiểu sai thời gian do đồng hồ lệch.

Hậu quả:

- Mất dữ liệu.
- Gây xung đột trong phiên bản mã nguồn.
- Làm giảm độ tin cậy của hệ thống phân tán.

Nguyên tắc hoạt động của giải thuật Cristian

Giải thuật Cristian là một giải pháp đồng bộ thời gian một chiều giữa máy khách (client) và máy chủ thời gian (time server), được đề xuất vào năm 1989. Giải thuật này được áp dụng trong các hệ thống phân tán để đồng bộ đồng hồ của máy khách với thời gian UTC do máy chủ cung cấp, giảm sai số do độ trễ mạng gây ra.



Quy trình hoạt động gồm các bước chính như sau:

1. Máy khách gửi yêu cầu đồng bộ thời gian đến máy chủ tại thời điểm T1 (đồng hồ máy khách).
2. Máy chủ nhận yêu cầu tại thời điểm T2 (đồng hồ máy chủ).
3. Máy chủ phản hồi lại thông điệp chứa thời gian tại thời điểm T3 (cũng theo đồng hồ máy chủ).
4. Máy khách nhận phản hồi tại thời điểm T4 (đồng hồ máy khách).

Giả thiết của giải thuật

- Độ trễ mạng chiều đi (client → server) và chiều về (server → client) là xấp xỉ bằng nhau, tức là:

$$|T2 - T1| \approx |T4 - T3|$$

3. Cách tính toán độ lệch (Θ) và hiệu chỉnh đồng hồ

Độ lệch thời gian Θ (theta) giữa máy khách và máy chủ được tính theo công thức:

$$\theta = \frac{(T2 - T1) + (T3 - T4)}{2}$$

Trong đó:

- T2−T1: thời gian đi từ client đến server.
- T3−T4: thời gian từ server trả lời đến khi client nhận được.

Giải thích:

- Tổng độ trễ ước lượng trên đường truyền: T4−T1
- Trong đó độ trễ xử lý của máy chủ: T3−T2
- Vậy độ trễ mạng mỗi chiều ước lượng là:

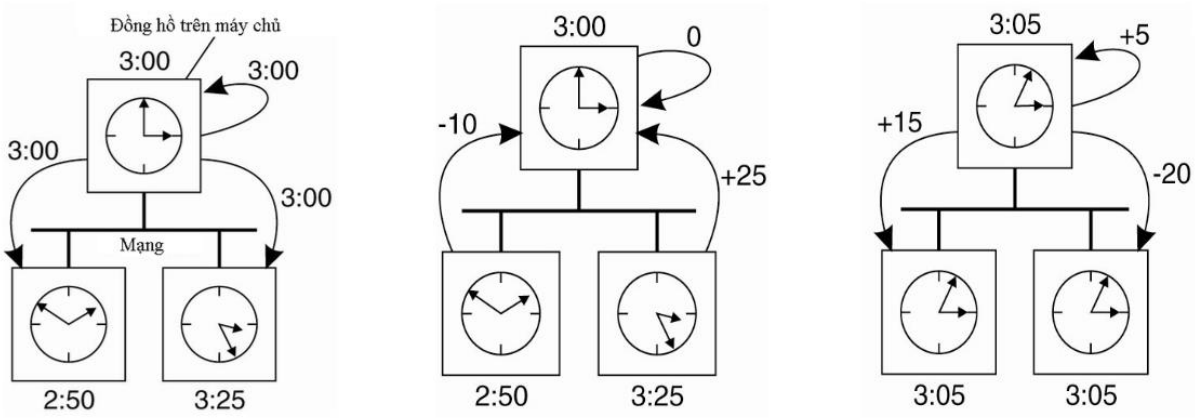
$$\delta = \frac{(T4 - T1) - (T3 - T2)}{2}$$

Cách hiệu chỉnh đồng hồ máy khách:

- Máy khách sẽ cộng thêm giá trị Θ vào thời gian hiện tại để tiến gần hơn đến thời gian thực của máy chủ.
- Tuy nhiên, nếu:
    - Θ < 0: máy khách chạy nhanh → không nên lùi thời gian mà nên điều chỉnh dần (bằng các chu kỳ điều chỉnh 10ms).
    - Θ > 0: máy khách chạy chậm → có thể tăng thời gian dần nếu không gây bỏ sót sự kiện theo lịch.

Giải thuật Berkeley – Cách thức hoạt động

Giải thuật Berkeley là một giải pháp đồng bộ thời gian tập thể trong hệ thống phân tán không phụ thuộc vào đồng hồ UTC hoặc máy chủ thời gian toàn cầu. Thay vì đồng bộ theo một chuẩn cố định, Berkeley đồng bộ tương đối, giúp các máy trong hệ thống đồng thuận về một thời gian chung.



Các bước hoạt động:

1. Khởi xướng đồng bộ:
    - Một tiến trình đóng vai trò điều phối (coordinator) sẽ gửi thông điệp đến tất cả các thành viên (bao gồm chính nó) để yêu cầu đồng bộ.
    - Thông điệp chứa thời gian hiện tại của điều phối viên.
2. Các thành viên phản hồi:
    - Mỗi máy thành viên tính độ lệch thời gian của mình so với thời gian nhận được và gửi độ lệch trở lại cho điều phối viên.
3. Điều phối viên xử lý phản hồi:
    - Nhận các độ lệch thời gian từ các thành viên.
    - Tính trung bình cộng của tất cả độ lệch (bao gồm chính nó).
    - Bỏ qua các phản hồi bất thường (nếu có).
    - Tính toán độ lệch riêng cho từng thành viên để điều chỉnh thời gian sao cho tất cả đồng bộ theo thời gian trung bình.
4. Gửi yêu cầu điều chỉnh:

    - Điều phối viên gửi yêu cầu điều chỉnh cụ thể đến từng máy để các máy cập nhật lại đồng hồ.

- Điều phối viên có thời gian là 03:00.
- Các thành viên gửi lại độ lệch:
    - Điều phối viên: 0 phút
    - Thành viên bên trái: -10 phút (đang là 02:50)
    - Thành viên bên phải: +25 phút (đang là 03:25)
- Trung bình cộng: (0-10+25)/3 = 5

- Điều chỉnh:
    - Điều phối viên cộng +5 phút → 03:05
    - Thành viên trái cộng +15 phút
    - Thành viên phải lùi -10 phút

So sánh: Giải thuật Berkeley vs. Cristian

Tiêu chí	Giải thuật Cristian	Giải thuật Berkeley
Nguồn thời gian chuẩn	Máy chủ thời gian (NTP) cung cấp thời gian chuẩn UTC	Không có máy chủ chuẩn – tính trung bình giữa các thành viên
Vai trò máy chủ	Thụ động – chỉ phản hồi yêu cầu từ máy khách	Chủ động – điều phối và điều chỉnh toàn bộ quá trình
Vai trò máy khách	Gửi yêu cầu đồng bộ và tự điều chỉnh dựa trên phản hồi	Phản hồi độ lệch, nhận chỉ thị điều chỉnh từ điều phối viên
Đồng bộ theo	Chuẩn thời gian toàn cầu (UTC)	Thời gian trung bình nội bộ hệ thống
Tính chính xác	Cao nếu mạng ổn định và máy chủ chính xác	Phụ thuộc vào độ ổn định mạng và thành viên
Phù hợp	Khi có máy chủ chuẩn (NTP)	Hệ thống nội bộ, cô lập, không có đồng hồ chuẩn
Xử lý độ trễ mạng	Có – giả thiết mạng đối xứng, tính toán trễ mỗi chiều	Có thể tính nếu đo thời gian phản hồi, nhưng không chính xác
Độ tin cậy	Tin cậy nếu có máy chủ thời gian hoạt động tốt	Kém hơn – nếu điều phối viên hỏng hoặc có thành viên không phản hồi, thuật toán thất bại
Khả năng mở rộng	Tốt – nhiều máy khách có thể đồng bộ với cùng máy chủ	Kém – thêm thành viên yêu cầu tính toán lại toàn hệ thống
Bảo mật	Có thể kết hợp xác thực với NTP	Dễ bị tấn công nếu thành viên độc hại gửi sai lệch thời gian

Giải thuật đồng bộ thời gian Trung Bình

Nguyên tắc hoạt động:

- Không có thành viên điều phối.
- Mỗi máy tính trong hệ thống định kỳ quảng bá thời gian của mình vào mỗi chu kỳ đồng bộ (khoảng thời gian thứ i tính từ $T_0 + iR$).
- Trong thời gian lắng nghe (S), máy tính thu thập thông tin thời gian từ các thành viên khác.
- Để giảm nhiễu, các thành viên loại bỏ M giá trị lớn nhất và nhỏ nhất.
- Sau đó, tính trung bình thời gian còn lại và cập nhật lại đồng hồ của mình theo giá trị trung bình đó.

Ưu điểm:

- Không phụ thuộc vào một nút điều phối → tăng tính sẵn sàng và linh hoạt.
- Có thể hoạt động tốt trong môi trường không ổn định, nơi một số thành viên không phản hồi.
- Loại bỏ giá trị ngoại lai → tăng độ chính xác.
- Dễ triển khai trong các nhóm máy tính nội bộ.

Nhược điểm:

- Không đồng bộ theo thời gian UTC.
- Không đảm bảo khởi động đồng bộ cùng lúc → nếu thời gian lệch quá xa, có thể bỏ sót chu kỳ.
- Việc lựa chọn tham số S (thời gian nghe) và R (chu kỳ) phải phù hợp để tránh mất thông tin.
- Không phù hợp với mạng có độ trễ cao hoặc trôi đồng hồ mạnh.

Giải thuật đồng bộ Tham Chiếu Quảng Bá

Nguyên tắc hoạt động:

- Dùng trong mạng không dây.
- Một nút sẽ phát thông điệp đồng bộ (mà không chứa thời gian).
- Các thành viên nhận thông điệp sẽ ghi lại thời gian tại thời điểm nhận được tín hiệu.
- Sau đó, các thành viên trao đổi giá trị thời gian ghi nhận với nhau để tính toán độ lệch thời gian.
- Loại bỏ thời gian chờ trong giao diện mạng và tập trung vào thời gian lan truyền vật lý, giúp đồng bộ chính xác hơn trong môi trường không dây.
- Nếu lặp lại quá trình nhiều lần (M lần), có thể tính trung bình hoặc dùng hồi quy tuyến tính để mô hình hóa độ lệch thời gian.

Ưu điểm:

- Phù hợp với mạng không dây, nơi độ trễ không ổn định và giao thức cạnh tranh.
- Loại bỏ thời gian chờ gửi và chi phí tạo thông điệp → tăng độ chính xác.
- Không cần gửi thời gian cụ thể → giảm phức tạp trong xây dựng thông điệp.
- Có thể đồng bộ nhiều máy cùng lúc.

Nhược điểm:

- Không đồng bộ theo thời gian thực hoặc UTC.
- Phụ thuộc vào tính ổn định của thời gian lan truyền vật lý, nên chỉ tốt nếu không cần định tuyến qua nhiều nút.
- Độ chính xác còn phụ thuộc vào chênh lệch xử lý trên từng thiết bị.
- Nếu không áp dụng hồi quy tuyến tính, giá trị trung bình sẽ kém chính xác do sự trôi đồng hồ.

Câu 2 :

Tại sao việc đồng bộ hóa thời gian vật lý trong hệ thống phân tán gặp nhiều khó khăn?

- Trong hệ thống phân tán, mỗi máy tính có đồng hồ vật lý riêng, nhưng các đồng hồ này không thể hoàn toàn đồng bộ do:
    - Độ trễ truyền thông không đồng đều (mạng không ổn định).
    - Đồng hồ trôi (clock drift) khác nhau giữa các máy.
    - Không thể đảm bảo độ chính xác tuyệt đối vì thời gian lan truyền không đo được chính xác.
- Một số hệ thống chọn cách "ai cập nhật cuối cùng là đúng", nhưng điều này không đảm bảo nhất quán khi đồng hồ sai lệch.
- Kết luận: Không thể chỉ dựa vào đồng hồ vật lý để duy trì tính nhất quán và độ tin cậy giữa các tiến trình trong hệ phân tán.

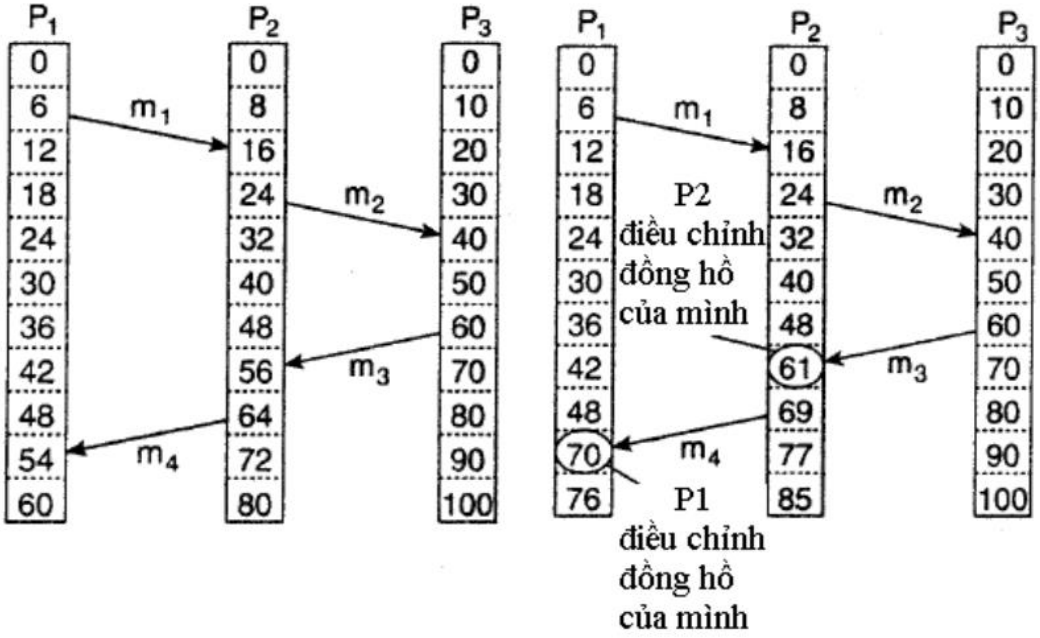
Khái niệm đồng hồ logic do Lamport đề xuất và vai trò của nó:

- Lamport (1978) đưa ra đồng hồ logic để thay thế việc đồng bộ thời gian vật lý.
- Đồng hồ logic:
    - Hoạt động độc lập với thời gian thực và độ trễ mạng.
    - Ghi nhận thứ tự nhân quả giữa các sự kiện (không phải thời điểm tuyệt đối).
- Vai trò: Đảm bảo rằng nếu một sự kiện xảy ra trước một sự kiện khác, thì thời gian logic của nó nhỏ hơn. Điều này cực kỳ quan trọng trong việc phát hiện lỗi, xử lý xung đột và duy trì thứ tự cập nhật trong hệ thống nhân bản.

Định nghĩa quan hệ “xảy ra trước” (happened-before) của Lamport:

- Ký hiệu: $a \rightarrow b$
- Gồm 3 trường hợp:
    1. Cùng một tiến trình: Nếu a xảy ra trước b thì $a \rightarrow b$.
    2. Gửi và nhận thông điệp: Nếu a là gửi, b là nhận cùng một thông điệp thì $a \rightarrow b$.
    3. Tính chất bắc cầu: Nếu $a \rightarrow b$ và $b \rightarrow c$ thì $a \rightarrow c$.

- Nếu không thể xác định x → y hay y → x, thì x và y là sự kiện tương tranh (concurrent), ký hiệu: x || y.



Hình 4.6 Đồng bộ nhãn thời gian Lamport

Cách hoạt động của thuật toán đồng hồ Lamport trong việc gán nhãn thời gian cho các sự kiện:

Thuật toán đồng hồ Lamport (Lamport Logical Clock) dùng để gán nhãn thời gian logic (timestamp) cho các sự kiện trong hệ thống phân tán nhằm đảm bảo thứ tự nhân quả (causality) giữa các sự kiện.

1. Khi xảy ra sự kiện nội bộ:

Mỗi tiến trình Pi có một bộ đếm đồng hồ logic riêng là Ci.

- Khi một sự kiện nội bộ xảy ra (tức là không liên quan đến gửi hay nhận thông điệp), tiến trình Pi sẽ tăng bộ đếm logic thêm 1 đơn vị:

$$C_i \leftarrow C_i + 1$$

2. Khi gửi thông điệp:

- Trước khi tiến trình Pi gửi một thông điệp m, nó cũng tăng đồng hồ logic của mình thêm 1 đơn vị:

  Ci←Ci+1

- Sau đó, nó đính kèm nhãn thời gian ts(m) = C_i vào thông điệp m trước khi gửi đi.

3. Khi nhận thông điệp:

- Khi một tiến trình Pj nhận được thông điệp m có đính kèm nhãn thời gian ts(m), tiến trình Pj sẽ cập nhật đồng hồ logic của mình theo công thức:

$$C_j \leftarrow \max(C_j, ts(m)) + 1$$

  Điều này đảm bảo rằng sự kiện nhận luôn xảy ra sau sự kiện gửi về mặt logic.

Cơ chế hiệu chỉnh nhãn thời gian (khi thông điệp đến với thời gian không hợp lệ):

Lamport chỉ cho phép đồng hồ logic được tăng, không được giảm. Trong một số trường hợp, do tốc độ đếm khác nhau giữa các tiến trình, thông điệp nhận có thể có nhãn thời gian nhỏ hơn thời điểm gửi. Để đảm bảo quan hệ nhân quả a → b thì phải có C(a) < C(b).

Ví dụ minh họa trong Hình 4.6:

TH1 - Bình thường (không cần điều chỉnh):

- Tại thời điểm 6, tiến trình P1 gửi thông điệp m1.
- Tại thời điểm 16, tiến trình P2 nhận được thông điệp m1.

Vì 16 > 6 nên P2 không cần điều chỉnh. Việc truyền thông điệp mất 10 đơn vị thời gian logic là hợp lệ.

TH2 - Nhận thông điệp với thời gian nhỏ hơn thời gian gửi (phải điều chỉnh):

- Tiến trình P3 gửi thông điệp m3 tại thời điểm 60.
- Tiến trình P2 nhận được m3 tại thời điểm 56.

Tức là: C(a) = 60, C(b) = 56 → không đảm bảo C(a) < C(b) → vi phạm quan hệ nhân quả.

Lamport yêu cầu: nếu ts(m) ≥ Cj thì cần cập nhật đồng hồ Cj như sau:

$$C_j \leftarrow ts(m) + 1 \Rightarrow C_j = 61$$

→ Tiến trình P2 điều chỉnh đồng hồ từ 56 thành 61 để đảm bảo 60 → 61.

TH3 - Tương tự m3, đến lượt m4:

- P2 gửi m4 tại thời điểm 69 (sau khi đã điều chỉnh từ m3).
- P1 nhận được m4 tại thời điểm 54.

→ 54 < 69 → Vi phạm nhân quả → P1 phải cập nhật:

$$C_1 \leftarrow ts(m) + 1 = 69 + 1 = 70$$

→ P1 điều chỉnh từ 54 thành 70.

So sánh Đồng hồ Lamport và Đồng hồ Vector trong việc theo dõi quan hệ nhân quả giữa các sự kiện:

Tiêu chí	Đồng hồ Lamport	Đồng hồ Vector
Theo dõi quan hệ nhân quả	Không đầy đủ – C(a) < C(b) không đủ để kết luận a → b	Chính xác – nếu VC(a) < VC(b) thì chắc chắn a → b
Loại nhãn thời gian	Số nguyên (duy nhất trên toàn hệ thống)	Vector (mảng số nguyên, độ dài bằng số tiến trình)
So sánh nhãn thời gian	Sử dụng so sánh số nguyên	So sánh vector phần tử tương ứng
Kết luận quan hệ xảy ra trước	Chỉ nếu các sự kiện thuộc cùng tiến trình hoặc gửi–nhận thông điệp	Có thể xác định quan hệ → giữa mọi sự kiện nếu tồn tại
Phân biệt các sự kiện đồng thời	Không thể	Có thể: nếu **VC(x)
Chi phí lưu trữ	Nhỏ (1 số nguyên cho mỗi sự kiện)	Lớn hơn (vector có độ dài bằng số tiến trình)
Ứng dụng	Hữu ích cho thứ tự tổng quát	Hữu ích cho đồng bộ nhân quả

Nguyên tắc hoạt động cơ bản của Đồng hồ Vector:

Cấu trúc dữ liệu:

- Mỗi tiến trình Pi duy trì một vector VCi = [v_0, v_1, ..., v_{n-1}] có độ dài n (số tiến trình).
- VCi[j] biểu diễn số lượng sự kiện mà tiến trình Pi biết là đã xảy ra trên tiến trình Pj.

Cập nhật đồng hồ vector:

Tình huống	Cách cập nhật
Sự kiện nội tại tại tiến trình Pi	VCi[i] ← VCi[i] + 1
Gửi thông điệp m từ Pi → Pj	Trước khi gửi: VCi[i] ← VCi[i] + 1 Gửi ts(m) = VCi (vector thời gian hiện tại của Pi)
Nhận thông điệp m tại Pj	VCj[j] ← VCj[j] + 1 VCj[k] ← max(VCj[k], ts(m)[k]) với mọi k ∈ [0..n-1]

Điều kiện để thông điệp được chuyển lên tầng ứng dụng:

Giả sử tiến trình Pj nhận được thông điệp m từ tiến trình Pi với nhãn thời gian vector ts(m), để m được xử lý thì hai điều kiện sau phải thỏa mãn:

1. ts(m)[i] == VCj[i] + 1
   → Đảm bảo rằng đây là thông điệp tiếp theo từ tiến trình Pi mà Pj đang chờ.
2. ∀k ≠ i: ts(m)[k] ≤ VCj[k]
   → Đảm bảo rằng Pj đã xử lý tất cả các thông điệp từ các tiến trình khác mà Pi đã biết trước khi gửi m.

Nếu một trong hai điều kiện không thỏa mãn, thông điệp sẽ bị lưu trữ trong vùng đệm cục bộ, chờ đến khi đủ điều kiện mới được chuyển lên tầng ứng dụng.

Khái niệm Trạng thái Toàn cục của Hệ thống Phân tán

Trong một hệ thống phân tán, trạng thái toàn cục (global state) là sự kết hợp của:

1. Trạng thái của từng tiến trình (gồm biến cục bộ, bộ đếm, dữ liệu nội tại,...).
2. Trạng thái của các kênh truyền thông (các thông điệp đang được gửi nhưng chưa được nhận).

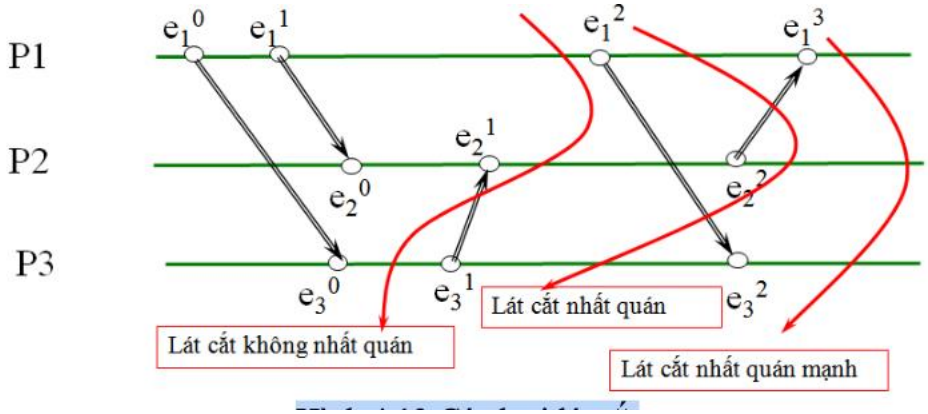
Tuy nhiên, do thiếu đồng hồ tuyệt đối, không thể xác định chính xác trạng thái của toàn hệ thống tại cùng một thời điểm thực. Vì vậy, ta cần một khái niệm thay thế: lát cắt (cut).

Khái niệm Lát cắt (Cut) trong Hệ thống Phân tán

Một lát cắt (cut) trong hệ thống phân tán là một tập hợp các sự kiện được chọn trên mỗi tiến trình, đại diện cho "thời điểm ghi nhận" trên tiến trình đó.

Cụ thể:

- Với mỗi tiến trình Pi, chọn một sự kiện thứ ci.
- Lát cắt C = {ei,ci | i = 1..n}, gồm sự kiện cuối cùng được ghi lại trên mỗi tiến trình tại thời điểm ci.
- Trạng thái tại lát cắt bao gồm:
  - Trạng thái cục bộ của mỗi tiến trình trước sự kiện được chọn.
  - Các thông điệp "đang bay" – được gửi trước lát cắt và nhận sau lát cắt.



Các loại Lát cắt :

Loại lát cắt	Đặc điểm	Ví dụ minh họa
Lát cắt không nhất quán	Chứa sự kiện nhận thông điệp nhưng không chứa sự kiện gửi tương ứng. → Trạng thái "không thể xảy ra" trong thực tế.	Nhận m đã xảy ra nhưng không thấy sự kiện gửi m.
Lát cắt nhất quán	Nếu e thuộc lát cắt và có f → e, thì f cũng thuộc lát cắt. → Có thể chứa sự kiện gửi nhưng chưa chứa sự kiện nhận tương ứng.	Đã gửi m nhưng chưa nhận m.
Lát cắt nhất quán mạnh	Với mọi quan hệ nhân quả f → e, cả f và e đều thuộc lát cắt. → Mọi gửi/nhận thông điệp đều được ghi lại đầy đủ.	Gửi m và nhận m đều nằm trong lát cắt.

Câu 3:

Khái niệm Tương tranh (Race Condition)

Tương tranh (race condition) là hiện tượng xảy ra khi hai hoặc nhiều tiến trình truy cập và thao tác đồng thời trên cùng một tài nguyên dùng chung (shared resource) mà thứ tự thực hiện các thao tác ảnh hưởng đến kết quả cuối cùng.

Ví dụ:

- Hai tiến trình cùng ghi dữ liệu vào một tệp tin.
- Hai giao dịch cùng sửa đổi số dư tài khoản ngân hàng.

Nếu không kiểm soát thứ tự truy cập, hệ thống có thể rơi vào trạng thái không nhất quán, gây lỗi hoặc mất dữ liệu.

---

2. Nguyên tắc Loại trừ Lẫn nhau (Mutual Exclusion)

Loại trừ lẫn nhau (mutual exclusion) là nguyên tắc đảm bảo rằng tại một thời điểm chỉ có một tiến trình được phép truy cập vào tài nguyên dùng chung (critical section – đoạn tới hạn).

Các yêu cầu chính của mutual exclusion:

1. An toàn (Safety): Tại mọi thời điểm, chỉ một tiến trình được trong vùng tới hạn.
2. Sống (Liveness): Mỗi tiến trình muốn vào vùng tới hạn sẽ được phép vào trong thời gian hữu hạn.
3. Công bằng (Fairness): Không có tiến trình nào bị bỏ đói vô thời hạn (no starvation).

---

3. Vì sao cần giải thuật Loại trừ Tương hỗ trong Hệ thống Phân tán

Trong hệ thống phân tán, việc đảm bảo loại trừ lẫn nhau phức tạp hơn so với hệ thống đơn tiến trình hoặc đa tiến trình vì:

Không có bộ nhớ chung

- Các tiến trình không chia sẻ không gian địa chỉ, giao tiếp thông qua truyền thông tin qua mạng.

Không có đồng hồ toàn cục

- Không thể dựa vào thời gian tuyệt đối để sắp xếp hoặc điều phối truy cập tài nguyên.

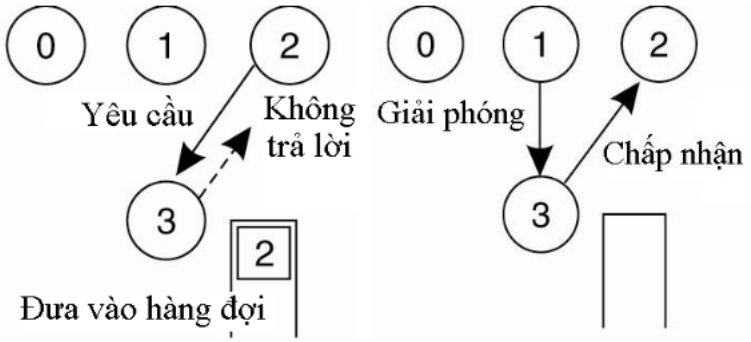
Mạng không tin cậy

- Thông điệp có thể bị trễ, mất mát hoặc đến không đúng thứ tự.

Vì vậy, cần sử dụng giải thuật loại trừ tương hỗ phân tán (distributed mutual exclusion algorithm) để đảm bảo:

- Tính nhất quán: Chỉ một tiến trình truy cập vào tài nguyên tại một thời điểm.
- Không phụ thuộc đồng hồ vật lý.
- Chịu lỗi một phần.

Giải thuật tập trung :



Hình 4.14 Nguyên lý giải thuật tập trung

Cách thức hoạt động

1. Chỉ định điều phối viên:
    - Hệ thống chỉ định một tiến trình làm điều phối viên, có thể:
        - Tĩnh: xác định sẵn từ đầu.
        - Động: sử dụng giải thuật bầu chọn (election) như chọn tiến trình có ID lớn nhất.
2. Gửi yêu cầu truy cập:
    - Khi một tiến trình (Pi) muốn truy cập tài nguyên:
        - Gửi thông điệp "yêu cầu" đến điều phối viên.
3. Xử lý yêu cầu tại điều phối viên:
    - Nếu tài nguyên đang rảnh, điều phối viên:

        - Gửi thông điệp "cấp phép" cho Pi.
    - Nếu tài nguyên đang bận, điều phối viên:
        - Đưa yêu cầu của Pi vào hàng đợi (queue).
4. Giải phóng tài nguyên:
    - Khi tiến trình Pi sử dụng xong:
        - Gửi thông điệp "giải phóng" cho điều phối viên.
    - Điều phối viên kiểm tra hàng đợi:
        - Nếu hàng đợi không rỗng, gửi "cấp phép" cho tiến trình đầu tiên trong hàng.

ƯU Điểm

Ưu điểm	Giải thích
Đơn giản	Thiết kế dễ hiểu và triển khai dễ dàng.
Chính xác	Đảm bảo loại trừ lẫn nhau 100%.
Công bằng (Fairness)	Yêu cầu nào đến trước sẽ được xử lý trước (FIFO).
Hiệu quả thông điệp	Mỗi lần truy cập chỉ cần 3 thông điệp (request, grant, release).

Nhược điểm:

Nhược điểm	Giải thích
Điểm lỗi đơn (Single Point of Failure)	Nếu điều phối viên bị lỗi, toàn hệ thống tê liệt, không truy cập được tài nguyên.
Bế tắc khi tiến trình treo	Nếu tiến trình sử dụng tài nguyên bị lỗi hoặc treo, không gửi release, điều phối viên sẽ chờ mãi.
Thắt cổ chai (Bottleneck)	Khi số lượng tiến trình tăng lên, điều phối viên phải xử lý nhiều yêu cầu, gây chậm trễ và giảm hiệu năng.
Không có thông báo rõ ràng khi bị trì hoãn	Tiến trình yêu cầu nếu chưa được cấp phép sẽ không biết khi nào được phục vụ vì không có phản hồi trung gian.

Giải thuật không tập trung (Voting / Decentralized Algorithm)

Giải thuật không tập trung là một phương pháp loại trừ tương hỗ trong hệ thống phân tán, trong đó không có một tiến trình điều phối duy nhất như giải thuật tập trung. Thay vào đó, quyền truy cập tài nguyên được kiểm soát bởi một nhóm các tiến trình điều phối, còn gọi là tập các "phiên bản" điều phối.

Nguyên lý hoạt động

- Mỗi tài nguyên được nhân bản N lần, mỗi bản có một tiến trình điều phối riêng.
- Khi một tiến trình (ví dụ: Pi) muốn truy cập tài nguyên:
    1. Nó gửi yêu cầu truy cập đến tất cả N điều phối viên.
    2. Nếu nhận được hơn N/2 phiếu đồng ý, nó được cấp quyền truy cập.
    3. Nếu không, tiến trình bị từ chối và phải chờ đợi trước khi thử lại.

Cách thức cấp quyền truy cập

- Mỗi điều phối viên chỉ cấp phép nếu:
    - Tài nguyên chưa được cấp cho ai khác.
    - Chưa cấp quyền truy cập cho một tiến trình khác.
- Khi tiến trình sử dụng xong:
    - Gửi thông điệp giải phóng đến những điều phối viên đã cấp phép trước đó.

Ưu điểm	Mô tả
Không có điểm lỗi đơn (no single point of failure)	Hệ thống vẫn hoạt động nếu một vài điều phối viên bị lỗi.
Phân tán trách nhiệm	Tránh quá tải trên một điều phối viên duy nhất.
Tăng tính sẵn sàng	Tài nguyên vẫn có thể được truy cập nếu một số điều phối viên không phản hồi.

Hạn chế khi nhiều tiến trình cạnh tranh mạnh

- Khi nhiều tiến trình gửi yêu cầu cùng lúc, có thể xảy ra:
    - Mỗi tiến trình nhận được một số phiếu không đủ (>N/2).
    - Không tiến trình nào được cấp quyền truy cập.
    - Dẫn đến hiện tượng "vòng lặp bế tắc giả" — không ai truy cập được tài nguyên, dù không có lỗi thật sự.
    - Gọi hiện tượng này là "hiện tượng loạn" (livestock or livelock).

Giải pháp đề xuất: Hồi phục bằng thời gian chờ ngẫu nhiên

Để giảm cạnh tranh đồng thời và tránh bế tắc giả, mỗi tiến trình sử dụng chiến lược chờ ngẫu nhiên trước khi gửi lại yêu cầu:

1. Biến đếm R:

- Mỗi lần bị từ chối truy cập, giá trị R tăng thêm 1.

2. Thời gian chờ (WaitTime) được tính như sau:

$$\text{WaitTime} = K \times T$$

- T: hằng số thời gian chuẩn trong hệ thống.
- K: số ngẫu nhiên trong khoảng

$$[0, 2^{\min(R,10)} - 1].$$

3. Tác dụng:
   - Các tiến trình sẽ có thời gian thử lại khác nhau.
   - Giảm xác suất va chạm yêu cầu cùng lúc.
   - Tăng khả năng một tiến trình nào đó có thể nhận đủ phiếu và truy cập thành công.

Tiêu chí	Tập trung (Centralized)	Phi tập trung (Voting)	Phân tán (Distributed - Ricart & Agrawala)	Thẻ bài (Token-based)
Số lượng thông điệp mỗi yêu cầu	3 (yêu cầu, cấp phép, giải phóng)	≥ 2N (yêu cầu + phản hồi từ N điều phối viên)	2(N-1) (broadcast yêu cầu và nhận phản hồi)	1 (chuyển token)
Độ trễ (Latency)	2 message delays	≥ 2 message delays	2 message delays	1 message delay
Điều phối viên trung tâm?	Có	Có nhiều (N bản sao)	Không (mọi tiến trình bình đẳng)	Không
Công bằng (Fairness)	Có	Có (tùy cơ chế hồi phục)	Có, nếu các yêu cầu được xử lý đúng thứ tự	Có (theo vòng token)
Chịu lỗi điều phối viên	Không	Một số lỗi vẫn chấp nhận	Không lỗi tiến trình thì vẫn hoạt động	Token mất gây lỗi
Phức tạp	Thấp	Trung bình	Cao (cần đồng bộ toàn bộ tiến trình)	Thấp
Hiệu năng khi hệ thống lớn	Thắt cổ chai	Khá tốt (nếu phân phối đều)	Tăng thông điệp nhanh, dễ nghẽn	Rất tốt
Khả năng xảy ra bế tắc/loạn	Có nếu tiến trình giữ tài nguyên bị lỗi	Có thể loạn nếu cạnh tranh mạnh	Có (nếu yêu cầu không đến đủ nơi hoặc lỗi truyền)	Có thể mất token
Khả năng mở rộng	Kém	Trung bình	Kém nếu không tối ưu	Tốt (token vòng, mạng nhẫn)

Phù hợp với hệ thống có	Ít tiến trình, đơn giản	Tài nguyên được sao chép kiểm soát	Cần công bằng cao, mạng ổn định	Vòng tuần tự, mạng ổn định