Các giả thiết về hệ thống phân tán

- Thông tin được lưu rải rác trong nhiều máy
- Các tiến trình quyết định dựa trên thông tin cục bộ
- Cần phải tránh lỗi thắt nút tại 1 điểm
- Không tồn tại một đồng hồ chung hay nhịp thời gian tổng thể

Mô hình dùng đồng hồ logic

- Dùng timer thay vì clock, tức là chỉ quan tâm tới quy ước trước/sau, không quan tâm tới giờ thực tế
- Giải pháp của Lamport:
  - trên cùng máy: nếu *a* xảy ra trước *b* thì $a \Rightarrow b$
  - khác máy: nếu *a* là sự kiện gửi thông báo, *b* là sự kiện nhận thông báo do từ sự kiện *a* thì $a \Rightarrow b$

Gán thời gian logic

- Gọi C(*a*) là nhãn thời gian gán cho sự kiện *a*
- Ràng buộc về thời gian logic:
  - nếu $a \Rightarrow b$ thì C(a) < C(b)
  - nếu a ≠ b thì C(a) ≠ C(b)

  điều kiện sau dễ dàng đạt được bằng cách gán số nhận dạng tiến trình thêm vào nhãn thời gian, ví dụ: 10.1 và 10.2 ứng với a và b xảy ra tại cùng nhãn thời gian 10

Đồng bộ nhãn thời gian logic

- Tại mỗi tiến trình lưu một bộ đếm thời gian. Các thông báo trao đổi đều đính kèm nhãn thời gian của tiến trình gửi
- Quy ước *a* là sự kiện gửi, *b* là nhận
- Nếu $C(a) > C(b)$, đồng hồ bên nhận phải được chỉnh thành $C(a)+1$
- Không được phép vặn lùi đồng hồ
- Quá trình vặn xuôi được thực hiện từ từ bằng cách tăng giá trị đếm độ dài 1 giây lên cho tới khi chỉnh xong

Mô hình dùng đồng hồ vật lý

- Đồng bộ nhịp thời gian qua máy thu thời gian UTC qua sóng ngắn radio hoặc từ GEOS. Độ chính xác (±10ms hoặc ±5ms)
- Phương pháp Cristian:
  - dùng 1 máy trung tâm với UTC
  - các máy khác đồng bộ với máy trung tâm bằng cách ước lượng trừ đi thời gian truyền

Phương pháp lấy khoảng trung bình

- Nhiều nguồn UTC
- Các máy broadcast khoảng thời gian UTC của mình [utc-σ,utc+σ]
- Mỗi máy lấy giao các khoảng nhận được (bỏ khoảng nằm ngoài) và lấy điểm giữa

Vấn đề loại trừ lẫn nhau

- Phương pháp tập trung:
  - mô phỏng lại một hệ thống tập trung
  - một máy làm nhiệm vụ quản lý critical region và semaphore để vào vùng đó
  - giải pháp này tạo ra cổ chai và single-point failure

...loại trừ lẫn nhau

- Giải pháp phân tán: tiến trình cần vào critical region sẽ broadcast tới nhóm
  - nếu tiến trình nhận không trong critical region và không cần vào, nó trả lời OK
  - nếu tiến trình nhận đang trong critical region, nó chưa trả lời OK cho tới khi ra khỏi vùng
  - nếu tiến trình nhận cũng muốn vào critical region, nó sẽ so nhãn thời gian của 2 bên; bên nào nhỏ nhất sẽ được vào trước
- Cải tiến: gửi có xác nhận và chỉ cần quá nửa số máy trả lời

...loại trừ lẫn nhau

- Giải pháp phân tán theo vòng:
  - mô hình token ring: ai giữ token được phép vào critical region
  - vấn đề phát hiện mất token: dùng thông báo alive đều đặn phát bởi người giữ

Atomic transaction

- Các primitives:
  - begin_transaction
  - end_transaction
  - abort_transaction
  - read, write
- Tính chất của transaction:
  - tuần tự hoá được
  - tính atomic
  - tính bền vững

Giao dịch lồng nhau (nested transaction)

- Transaction có thể sinh ra các transactions con (do vậy tính chất permenance chỉ áp dụng cho transaction cha)
- Mỗi transaction con phải thao tác trên không gian riêng và chỉ cập nhật lại cho transaction cha khi commit (end_transaction)

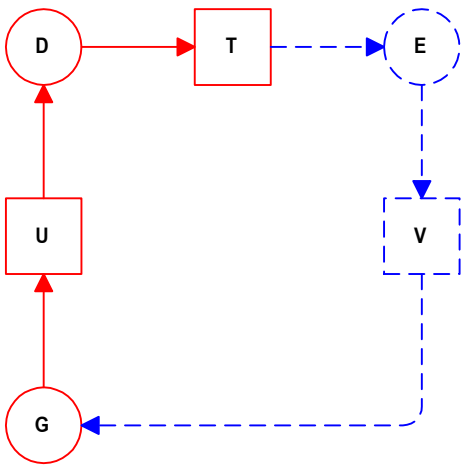
Cài đặt nested transactions

- Giải pháp copy toàn bộ tài nguyên transaction cha: chi phí quá đắt
- Giải pháp sử dụng bảng chỉ mục (dạng bảng i-nodes đối với đĩa):
  - bảng chỉ mục tài nguyên transaction cha được sao cho transaction con
  - thao tác *read* của transaction con thực hiện trực tiếp trên vùng tài nguyên của cha
  - thao tác *write* của transaction con được thực hiện trên vùng cấp phát mới và bảng chỉ mục được sửa trỏ tới vùng này
  - việc cập nhật/huỷ bỏ được thực hiện khi commit hoặc abort

Deadlock trong hệ thống phân tán

- Bốn phương pháp truyền thống:
  - đà điểu
  - phát hiện và khắc phục
  - loại trừ một cách hệ thống
  - tránh deadlock bằng việc cấp phát tài nguyên chặt chẽ
- Giải pháp cuối không thực tế (vd bài toán banker) khi số tài nguyên và tiến trình luôn thay đổi. Giải pháp đầu thực sự là không làm gì.

Phát hiện deadlock



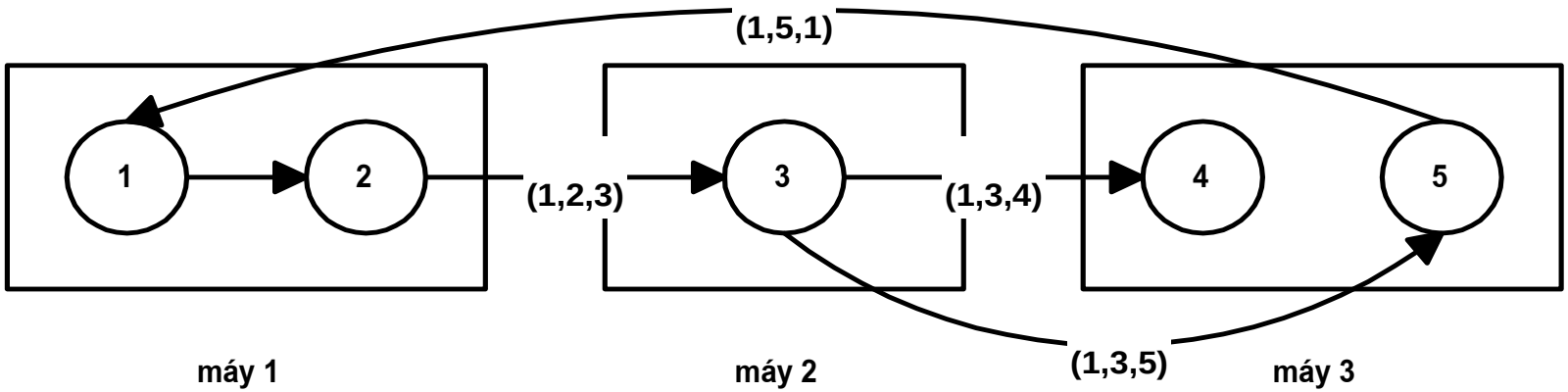
nét liền: thuộc máy 1

nét gạch: thuộc máy 2

- Giải pháp tập trung: máy server quản lý và lưu đồ thị phụ thuộc giữa tiến trình và tài nguyên
- Nếu phát hiện có chu trình, t.l deadlock, thì huỷ bỏ yêu cầu
- Vấn đề: false deadlock do thứ tự thông báo tới server. Cần giải quyết bằng cách gán nhãn thời gian logic

Phát hiện deadlock

- Tiến trình chờ tài nguyên sẽ gửi thông báo đính kèm 3 thông tin: (origin,sender,receiver)
- nếu thông báo quay vòng chứng tỏ có dead lock xảy ra; việc chờ bị huỷ bỏ



Loại trừ deadlock

- Giải pháp phân tán:
  - thực hiện chờ tài nguyên theo nguyên tắc: chỉ có tiến trình “già” chờ tiến trình “trẻ” (dựa trên thời gian logic) hoặc ngược lại. Do vậy không có chu trình
  - Phương thức cấp phát wait-die:
    - tiến trình già có thể chờ tiến trình trẻ
    - tiến trình trẻ chờ tiến trình già sẽ bị killed-off
  - Phương thức cấp phát wound-wait:
    - tiến trình già chờ tiến trình trẻ: thực hiện lấy tài nguyên của tiến trình trẻ cấp cho tiến trình già và sát thương (kill) tiến trình trẻ
    - tiến trình trẻ chờ tiến trình già: cho chờ