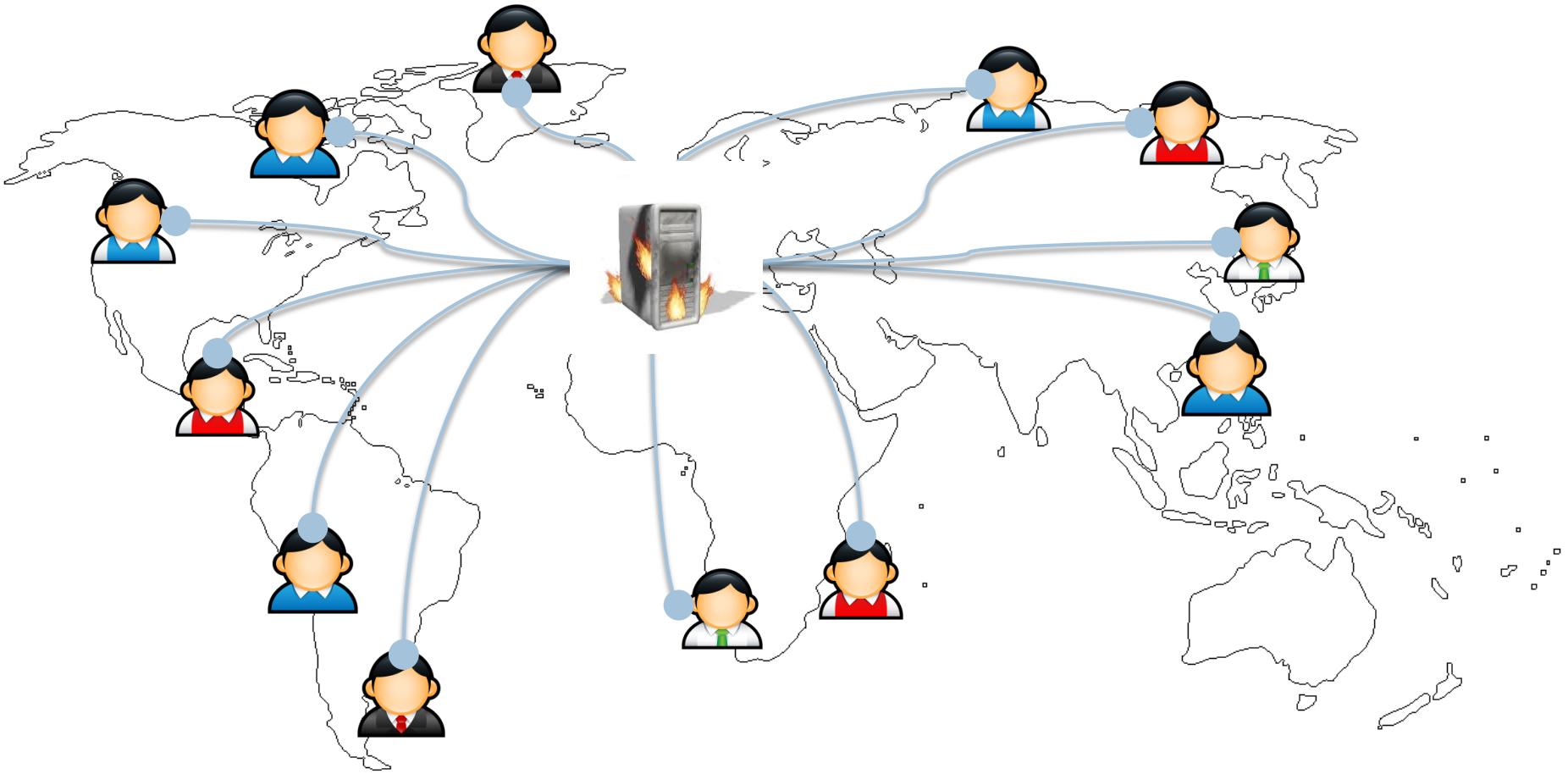
CHƯƠNG 7: SAO LƯU VÀ THỐNG NHẤT DỮ LIỆU

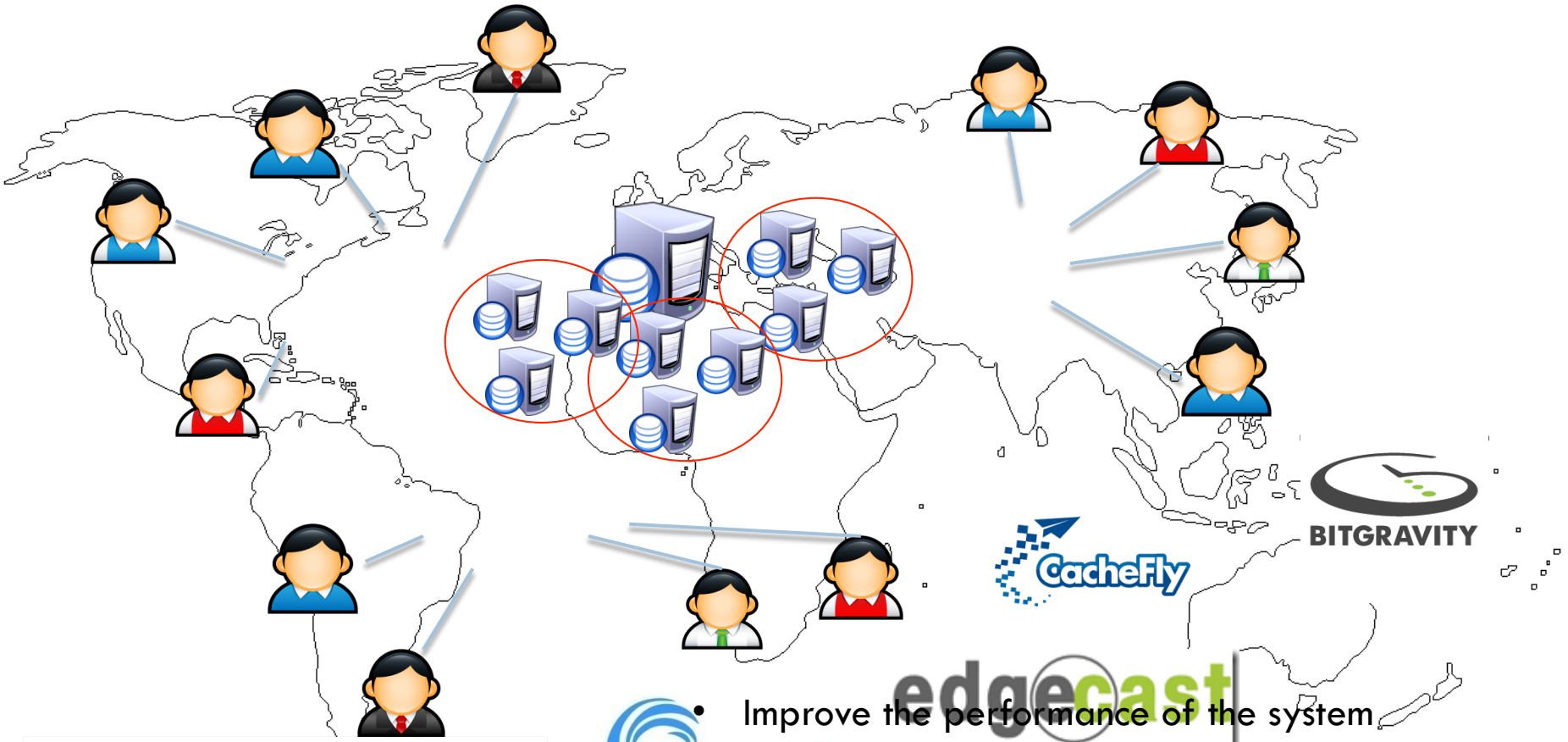
TS. Trần Hải Anh

Bài giảng được xây dựng dựa trên bài giảng của PGS. TS. Hà Quốc Trung

Problems

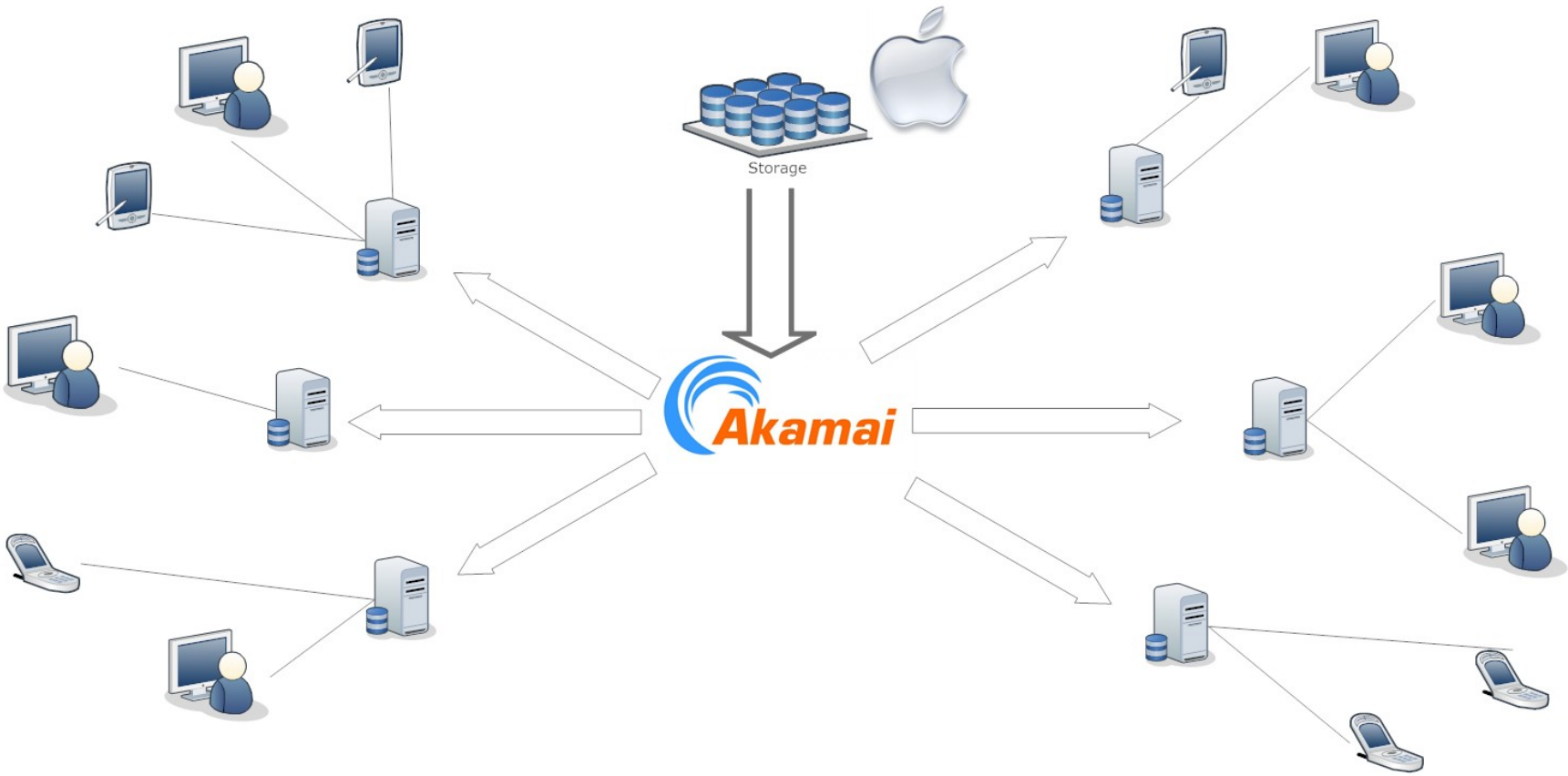


Content Delivery Network



- Improve the performance of the system
- Reduce network load
- Provide better fault tolerance

AKAMAI



Nội dung

1. Giới thiệu về sao lưu và thống nhất dữ liệu
2. Các mô hình sao lưu hướng dữ liệu
3. Các mô hình sao lưu hướng client
4. Quản lý các bản sao
5. Các giao thức sao lưu
6. Một số công cụ sao lưu

1. Giới thiệu

1.1. Vì sao phải sao lưu

1.2. Thống nhất dữ liệu

1.3. Ưu điểm, nhược điểm của sao lưu dữ liệu

1.1. Vì sao phải sao lưu

- Độ tin cậy (tính sẵn sàng)
- Hiệu năng
- Khả năng co giãn (?)

⇒ Yêu cầu về **thống nhất dữ liệu**

1.2. Thống nhất dữ liệu

- Các bản sao cần có một dữ liệu
  - Không thể tức khắc đồng bộ
  - Khi nào, như thế nào
- Tính thống nhất mạnh và tính thống nhất yếu
- Đạt được tính thống nhất mạnh=>tốn kém về hiệu năng
- Ví dụ:Bộ nhớ đệm của trình duyệt.
  - Để đảm bảo tính thống nhất:
    - Cấm không cho dùng bộ nhớ đệm☺
    - Server cập nhật bộ nhớ đệm khi có nội dung thay đổi☹
  - Giải pháp=> thống nhất *hợp lý*

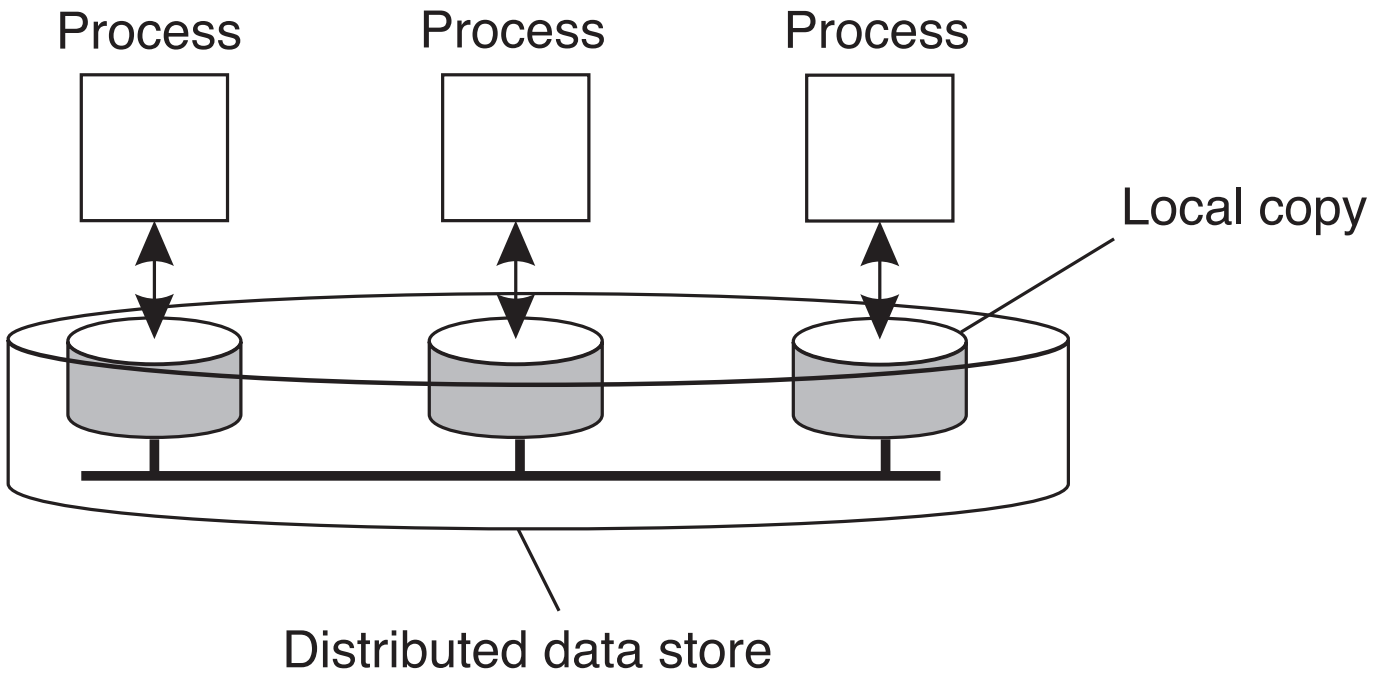
1.3. Ưu & nhược điểm

- Cải thiện tốc độ truy cập
- Giảm băng thông
- Có băng thông phát sinh
- Tăng mức độ phức tạp của hệ thống
- Phụ thuộc nhiều vào nhu cầu
  - Ví dụ: số lần cập nhật và số lần truy cập
  - Thống nhất chặt: giảm hiệu năng
  - Thống nhất lỏng: lỏng đến đâu? Mức độ thống nhất<>chi phí

2. Mô hình thống nhất hướng dữ liệu

2.1. Kho dữ liệu phân tán
2.2. Mô hình thống nhất liên tục
2.3. Connit
2.4. Thống nhất về thứ tự thực hiện

2.1. Kho dữ liệu phân tán



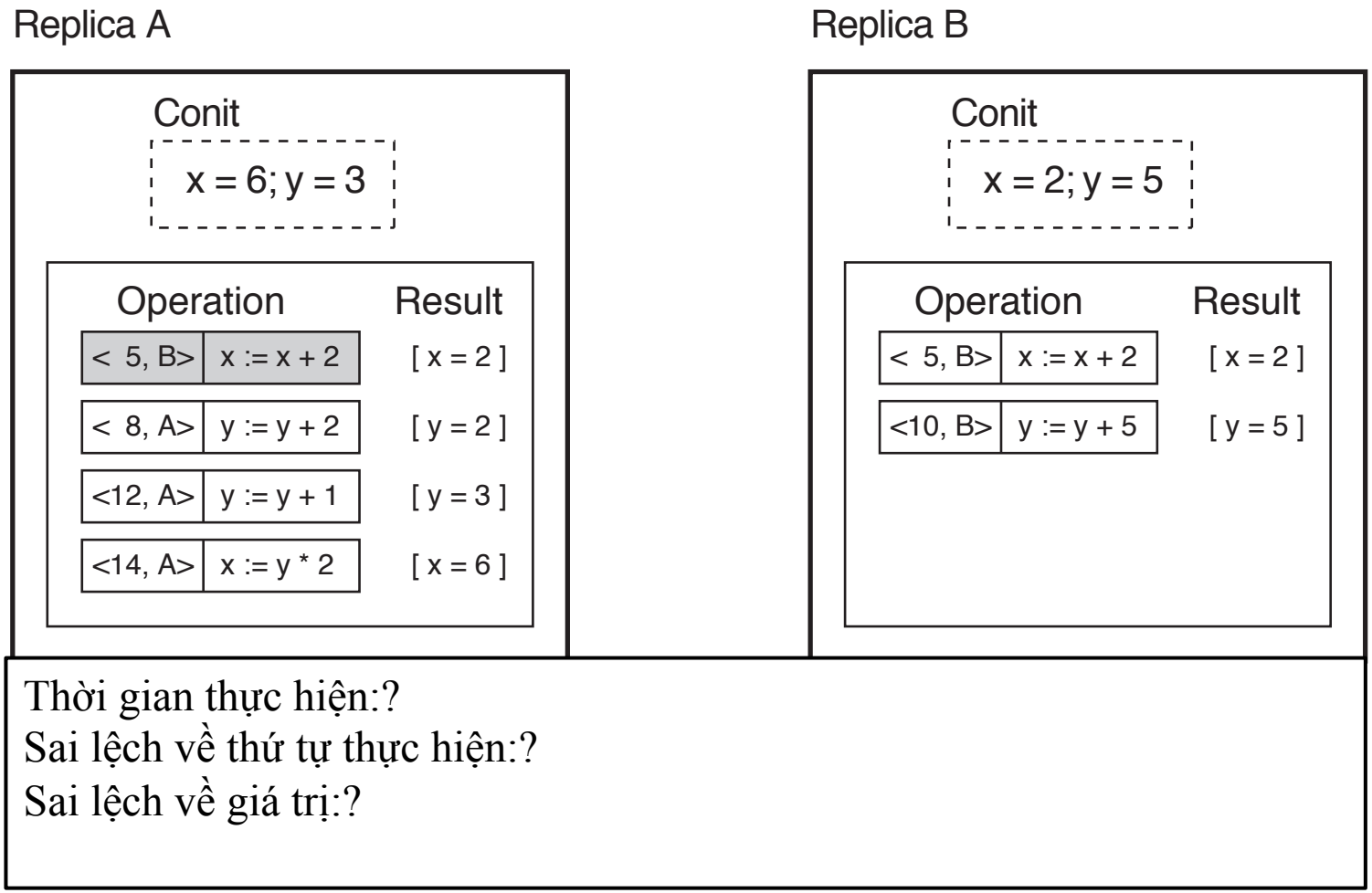
Mô hình thống nhất

- Cam kết giữa các tiến trình và kho dữ liệu
- Muốn đọc giá trị cuối cùng (mới nhất)
- Không có đồng hồ toàn cục → khó thực hiện
- Khái niệm **phạm vi** của mô hình thống nhất (độ lệch, độ sai khác)

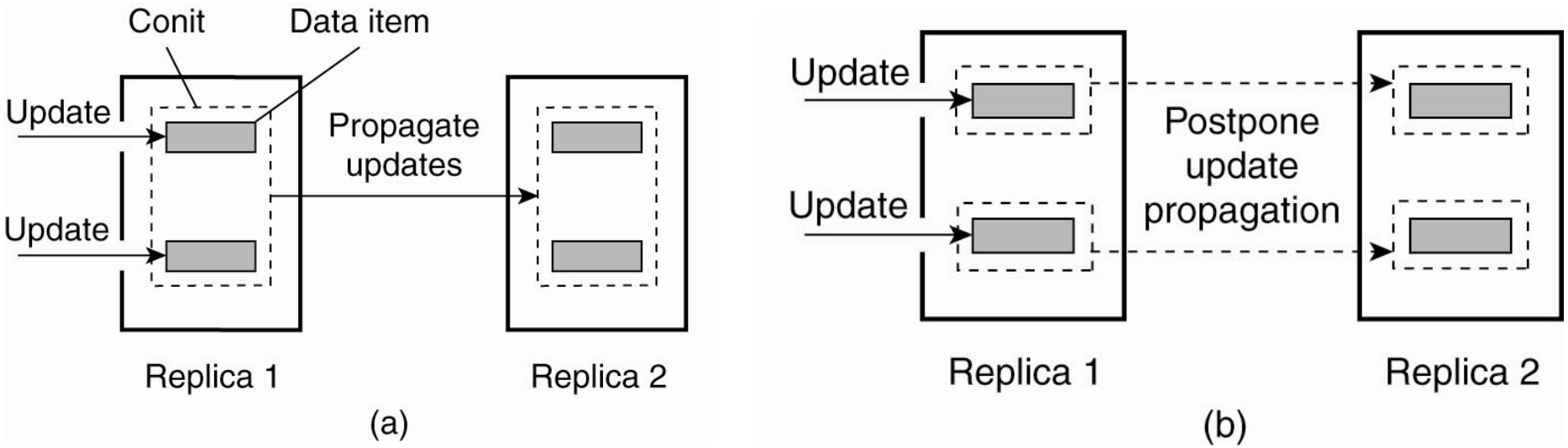
2.2. Mô hình thống nhất liên tục

- Những yếu tố đánh giá sự bất đồng bộ:
  - Chênh lệch giá trị của các biến (nhiệt độ, giá cả, …..)
  - Chênh lệch thời gian cập nhật
  - Thứ tự các thao tác cập nhật
- Khi độ lệch vượt quá một giá trị cho trước, MW sẽ tiến hành các thao tác đồng bộ để đưa độ lệch về giới hạn

2.3. Conit (consistency unit)



Kích thước nhỏ: thống nhất cao



- Kích thước lớn: Các bản sao sẽ sớm bị rơi vào trạng thái không thống nhất
- Kích thước nhỏ: số lượng conit nhiều: quản lý phức tạp
- => Bài toán: cho trước một (phần) tập dữ liệu, xác định kích thước connit theo các tiêu chí tối ưu

API cho lập trình viên

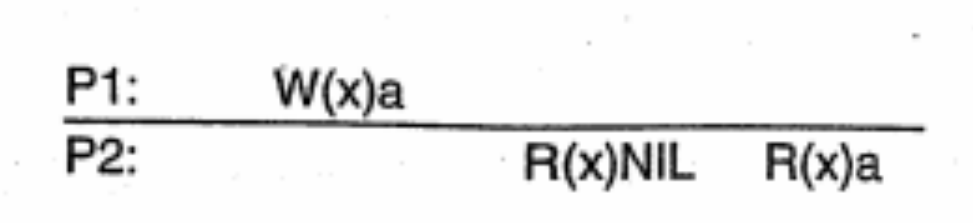
- Đặc tả được các yêu cầu về tính thống nhất với một conit
  - AffectsConit(aMQ, 1,1);
  - aMQ.add(m)
- Khai báo thao tác ảnh hưởng đến Conit
  - DependsConit(aMQ,4,0,60);
  - aMQ.read(m)

2.4. Mô hình thống nhất theo thứ tự thao tác

- Truy cập tương tranh đến các tài nguyên chia sẻ
- Tài nguyên chia sẻ là dữ liệu được sao lưu
- Mạnh hơn mô hình liên tục
- Khi thực hiện cập nhật, thứ tự cập nhật được thống nhất giữa các replicas

Một vài ký hiệu

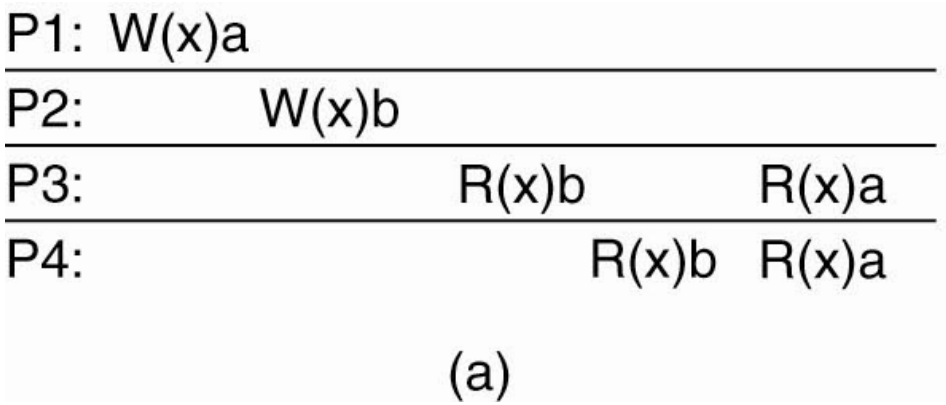
- Với các quá trình thực hiện khác nhau, tất cả các tiến trình luôn luôn cho một kết quả
- Các thao tác trên dữ liệu
  - Đọc (Ri(x)b)
  - Ghi (Wi(x)a)
  - Giá trị khởi tạo của các dữ liệu là NIL



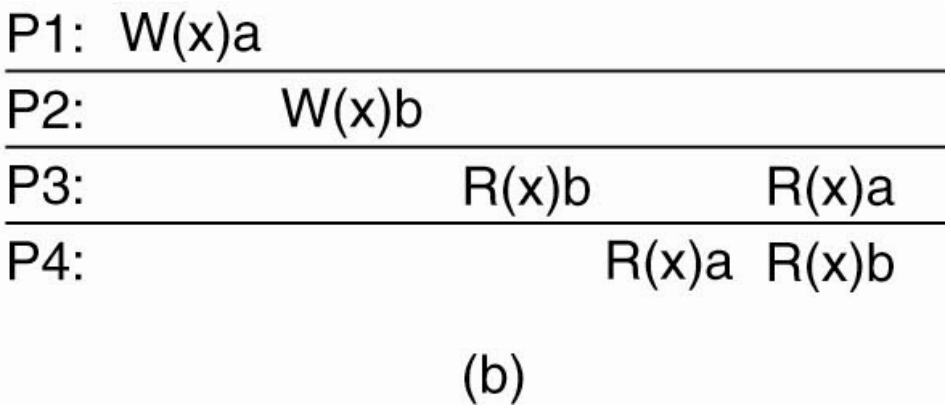
Thống nhất tuần tự

- Các tiến trình đều có một chuỗi thao tác cục bộ
- Các thao tác cục bộ của các tiến trình được tổng hợp thành thứ tự thực hiện các thao tác trên kho dữ liệu
- Có thể có các thứ tự thực hiện khác nhau trên kho dữ liệu
- Điều kiện của thống nhất tuần tự
  - Nếu thứ tự các thao tác cục bộ của một tiến trình không thay đổi trong thứ tự thực hiện chung trên kho dữ liệu =>Kết quả luôn luôn như nhau.
- Tất cả các tiến trình đều nhìn thấy một thứ tự của các thao tác ghi

Ví dụ - 1


P1: W(x)a
P2: W(x)b
P3: R(x)b R(x)a
P4: R(x)b R(x)a

(a)

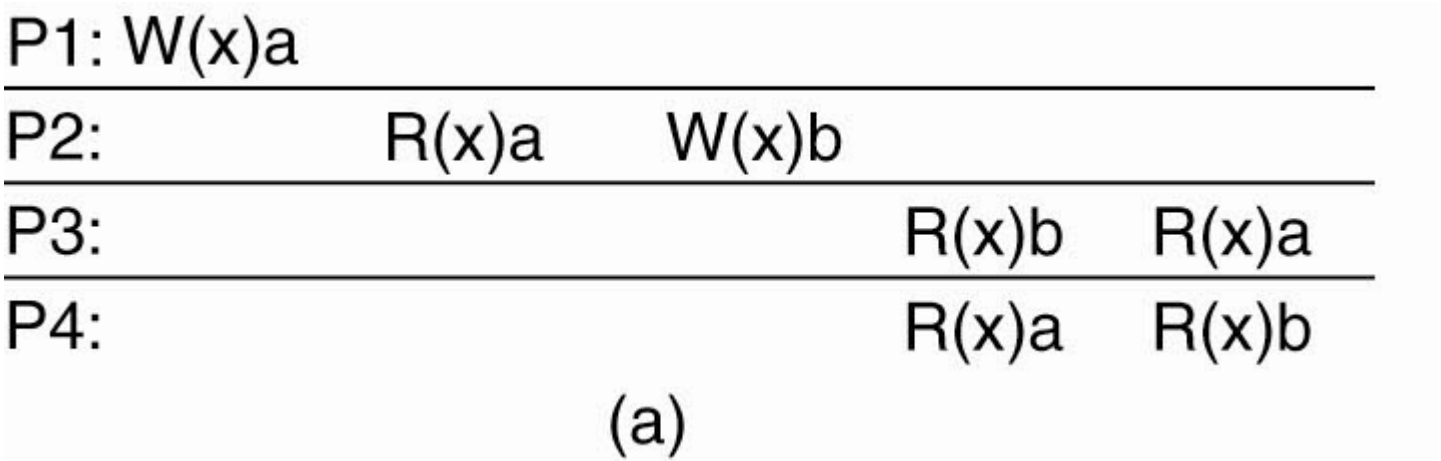

P1: W(x)a
P2: W(x)b
P3: R(x)b R(x)a
P4: R(x)a R(x)b

(b)

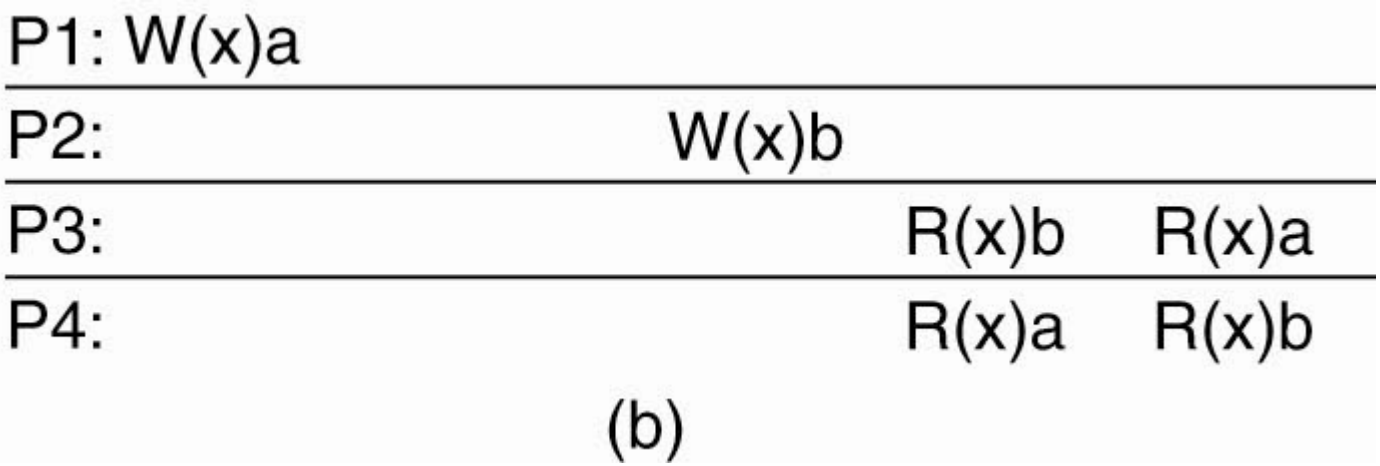
Thống nhất nhân quả

- Các sự kiện có quan hệ nhân quả đảm bảo thứ tự
  - Những sự kiện khác không cần
  - => Tính thống nhất yếu
- Các thao tác ghi có ràng buộc nhân quả cần được các tiến trình nhìn thấy theo cùng một thứ tự

Thống nhất nhân quả (cont.)



(a)



(b)

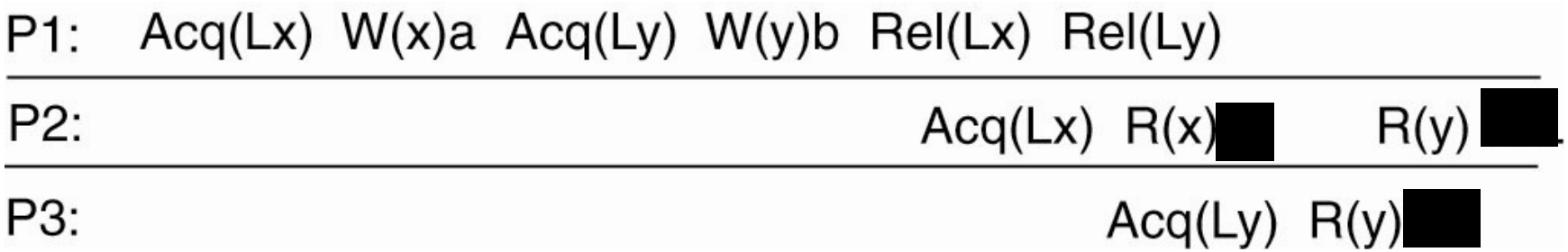
Các thao tác nhóm

- Thống nhất tuần tự và nhân quả là sản phẩm của bộ nhớ chia sẻ dùng chung
- Phù hợp với các quan hệ điểm điểm
- Buộc lập trình viên phải thiết kế các giao thức=> phức tạp
- Trong một số trường hợp, dữ liệu cần được quảng bá một lần cho tất cả các bản sao=> các thao tác nhóm
- Một trong các giao thức được sử dụng rộng rãi là sử dụng đoạn găng
- **ENTER_CS & LEAVE_CS**

Nguyên tắc

- Truy cập đoạn găng
  - Nhập (loại trừ-ghi, không loại trừ-đọc);
  - Xuất
- Đoạn găng xác định cho từng thành phần dữ liệu
- Nguyên tắc:
  1. Chỉ được truy cập đoạn găng khi tất cả các thao tác cập nhật đã hoàn tất
  2. Chỉ được truy cập đoạn găng (để ghi) khi không có tiến trình nào giữ quyền truy cập
  3. Trước khi truy cập đoạn găng để đọc, cần kiểm tra với chủ đoạn găng về tính cập nhật của dữ liệu
- Chia dữ liệu thành các mảnh để quản lý bằng các đoạn găng
  - Bảng, dòng, trường, cột, đối tượng, v.v…

Ví dụ



Thống nhất (consistency) và Phù hợp (coherence)

- Thống nhất (consistency): áp dụng cho tập hợp phần tử dữ liệu.
- Phù hợp (coherence): áp dụng cho một đơn vị dữ liệu/phần tử dữ liệu

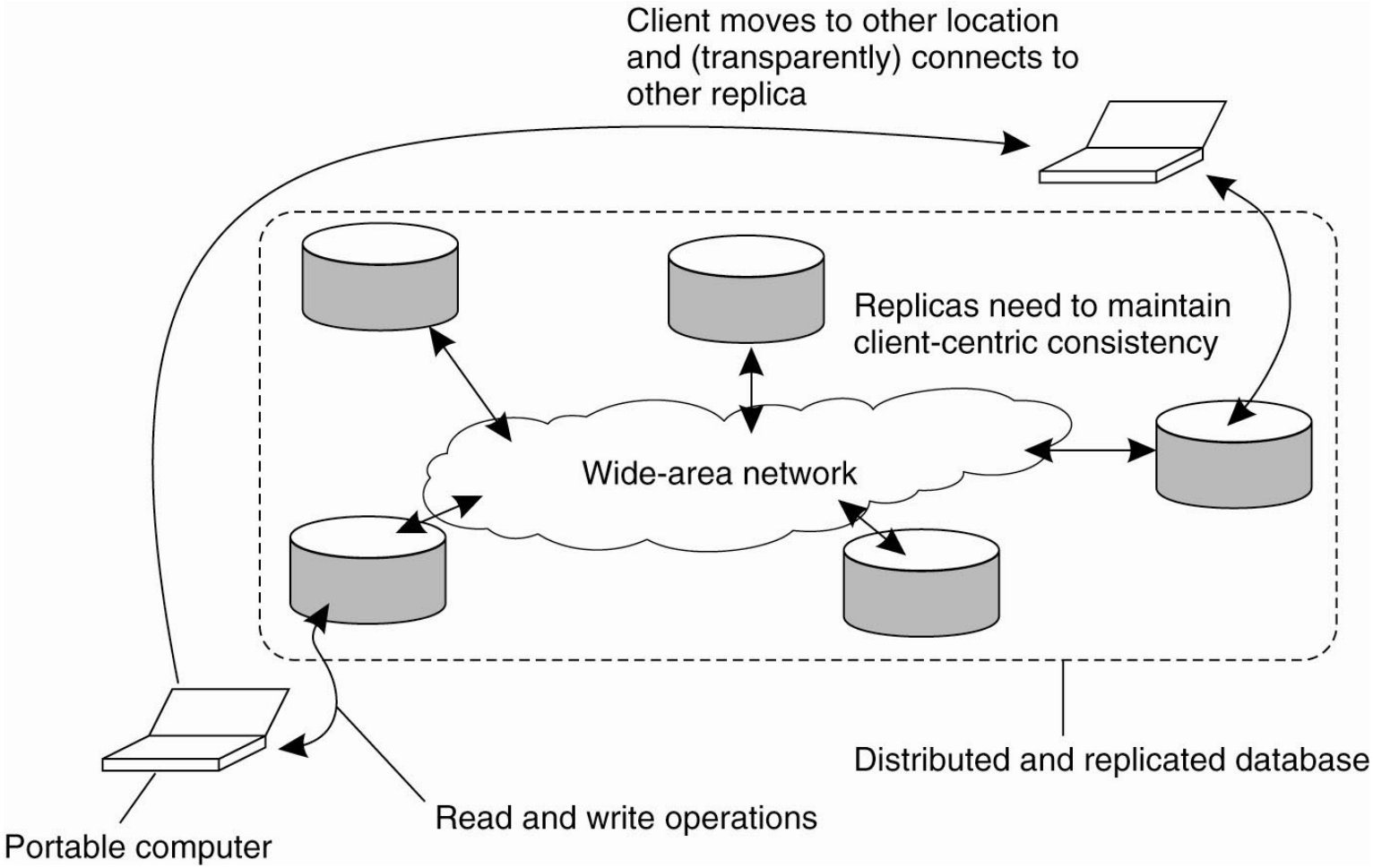
3. Mô hình hướng client

3.1. Thống nhất cuối cùng (eventual consistency)
3.2. Đọc đơn điệu
3.3. Ghi đơn điệu
3.4. Đọc kết quả đã ghi
3.5. Ghi theo thao tác đọc

3.1. Eventual Consistency

- Chủ yếu các tiến trình thực hiện đọc. Rất ít các tiến trình thực hiện cập nhật
- VD: DNS, WWW, v.v…
- Xung đột ghi-ghi hầu như không xảy ra
- Xem xét xung đột đọc-ghi
- Nếu dữ liệu không bị thay đổi trong thời gian đủ dài → thống nhất (Eventual Consistency)

Vấn đề của Eventual Consistency



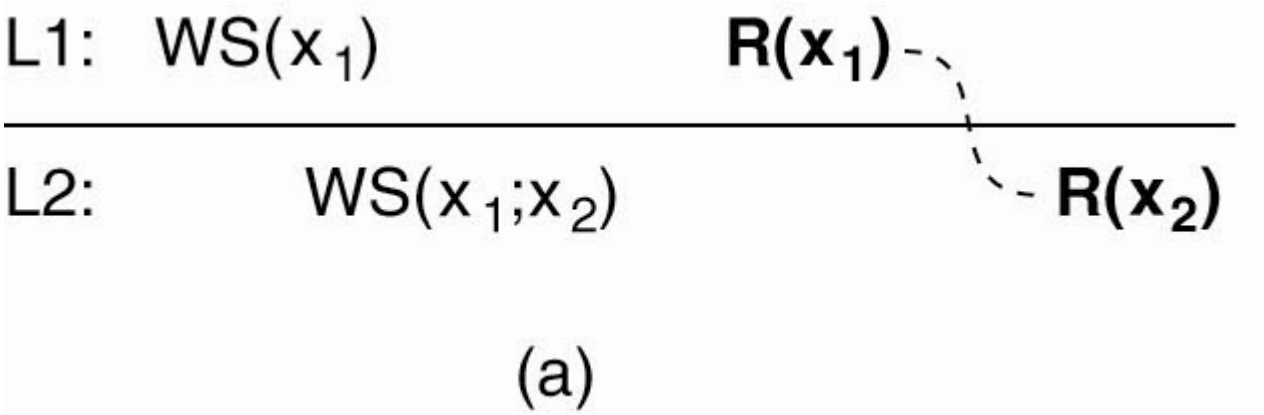
Mô hình thống nhất hướng client

- Cung ứng đảm bảo thống nhất cho các truy cập của một client đơn vào kho dữ liệu.
- Chú ý: không đảm bảo thống nhất cho các truy cập cạnh tranh của các tiến trình khác.
- Các kiểu mô hình:
  - Đọc đơn điệu
  - Ghi đơn điệu
  - Đọc kết quả đã ghi
  - Ghi theo thao tác đọc

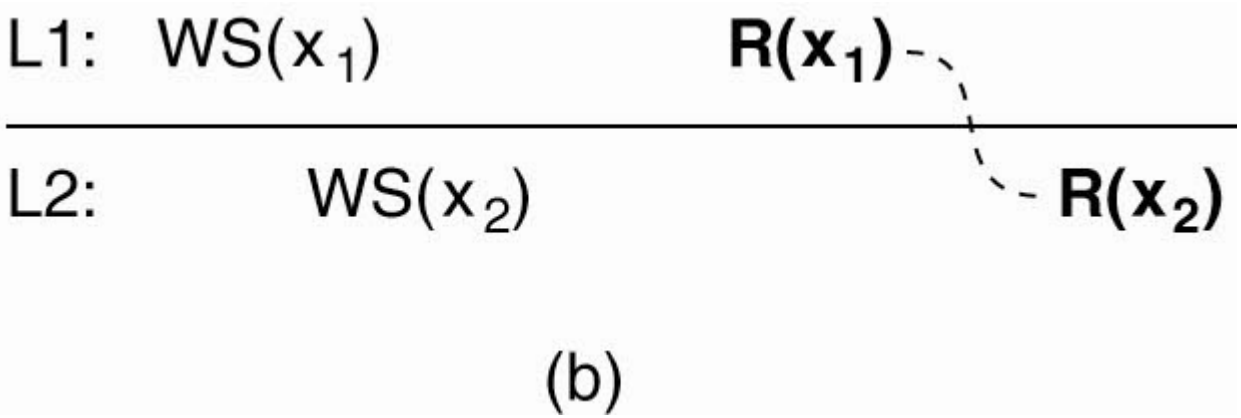
Ký pháp

- L_i: bản sao thứ i
- $x_i[t]$: phần tử dữ liệu x ở bản sao cục bộ L_i, thời điểm t
- $WS(x_i[t])$: các thao tác ghi phần tử x tại L_i đã được thực hiện từ lúc khởi đầu
- $WS(x_i[t_1]; x_j[t_2])$: Tất cả các thao tác $WS(x_i[t_1])$ đã được phổ biến đến bản sao L_j, sau khoảng thời gian t_2

3.2. Thống nhất đơn điệu đọc



(a)

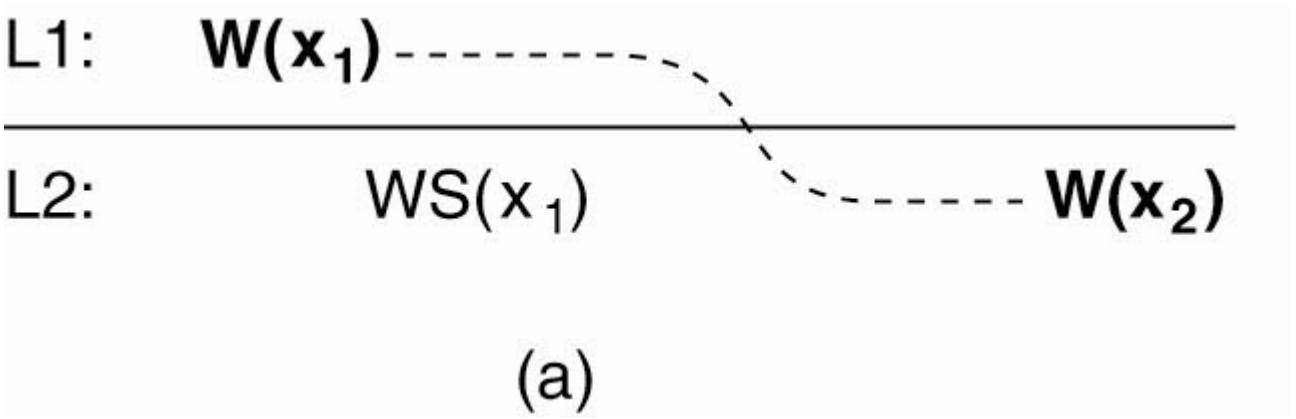


(b)

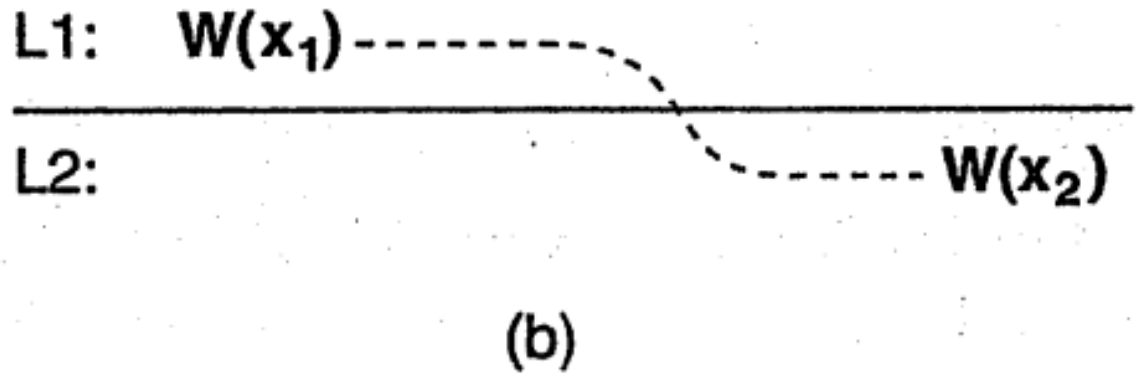
3.3. Đơn điệu ghi

- Các thao tác ghi của một tiến trình trên dữ liệu là rời nhau
- Tương tự như FIFO, nhưng chỉ có giá trị với một tiến trình
- Các thao tác ghi của một tiến trình cần được kết thúc trước khi tiến trình thực hiện bất cứ một thao tác ghi nào trên cùng một phần tử dữ liệu
- Các thao tác ghi cần chờ các thao tác ghi trước kết thúc

Đơn điệu ghi



(a)

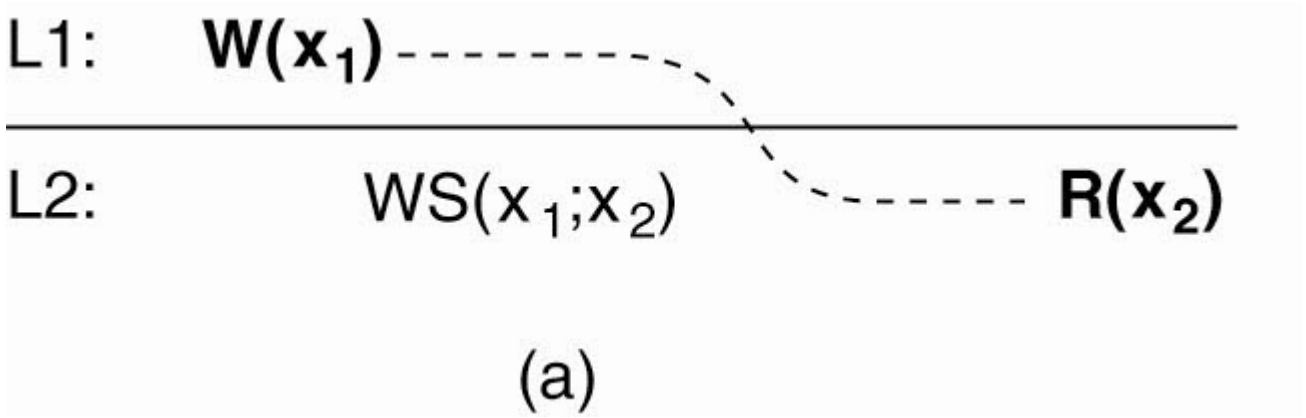


(b)

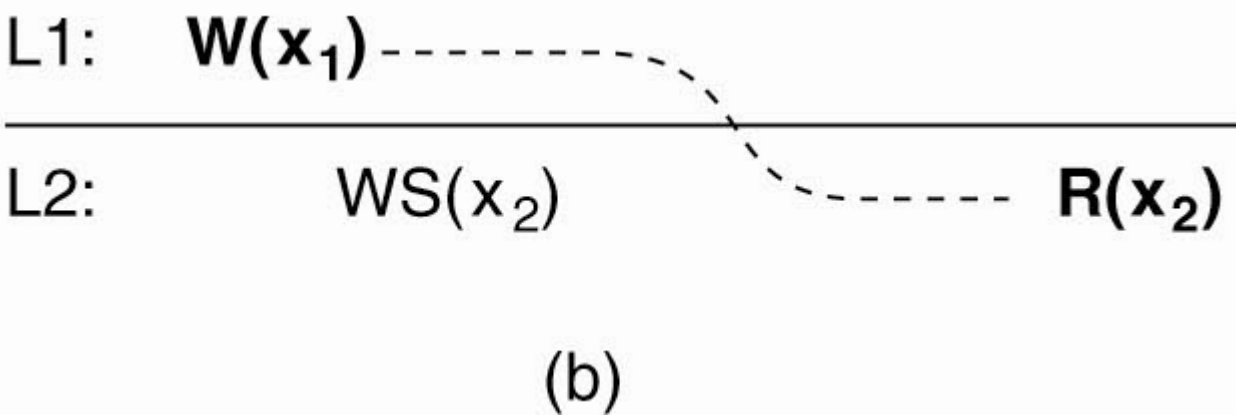
3.4. Đọc dữ liệu ghi

- Trên một tiến trình
  - Nếu thao tác đọc xảy ra sau thao tác ghi, thao tác đọc sẽ xảy ra sau khi thao tác ghi hoàn thành
  - Các thao tác đọc sẽ chờ sau khi thao tác ghi hoàn thành mới thực hiện
- Ví dụ
  - Cập nhật trang web, đọc nội dung trang web
  - Cập nhật mật khẩu

Đọc dữ liệu ghi



(a)

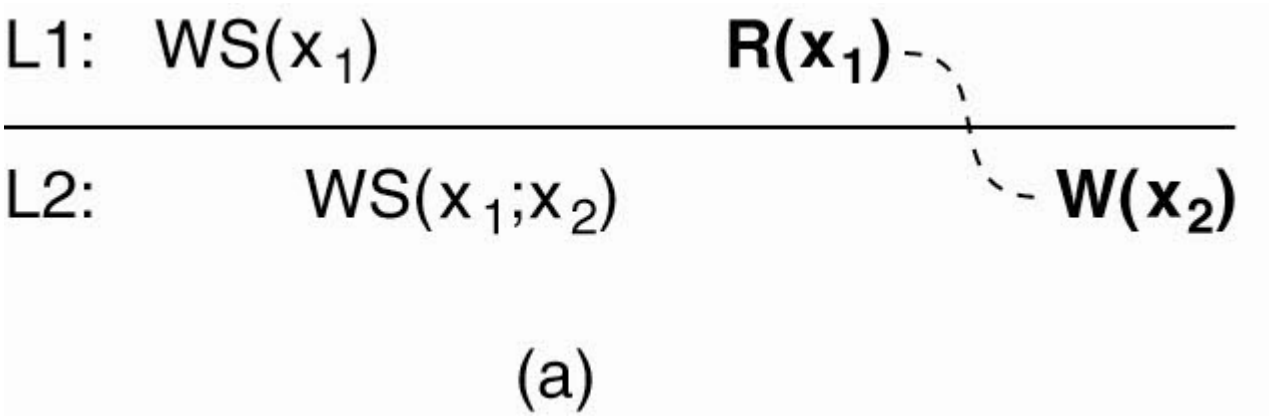


(b)

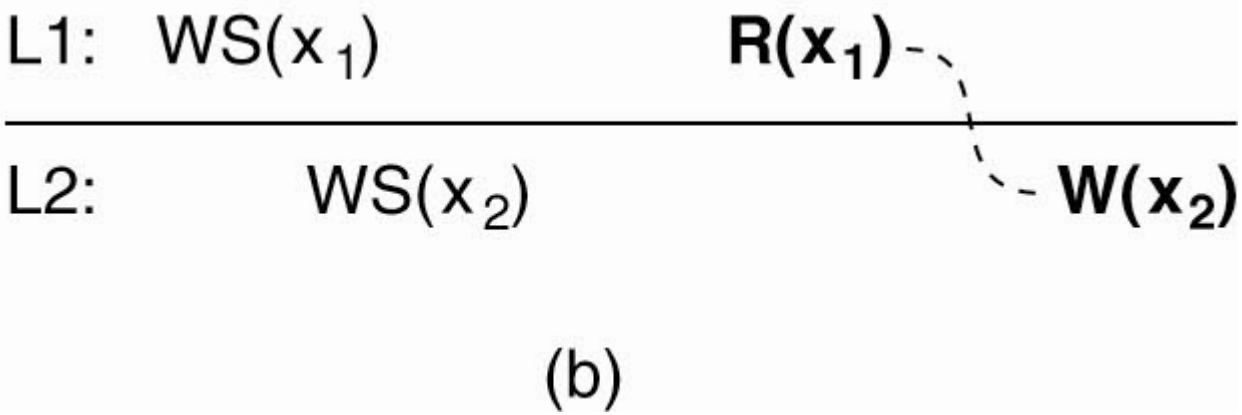
3.5. Ghi sau khi đọc

- Thao tác ghi thực hiện sau thao tác đọc
- Thao tác ghi chỉ được thực hiện sau khi thao tác đọc hoàn thành
- Ví dụ: Chỉ có thể trả lời sau khi đã đọc nội dung thư
- Thư đã ở bản sao cục bộ của dữ liệu=> có thể trả lời

Ghi sau khi đọc



(a)



(b)

4. Quản lý các bản sao

4.1. Quản lý máy chủ

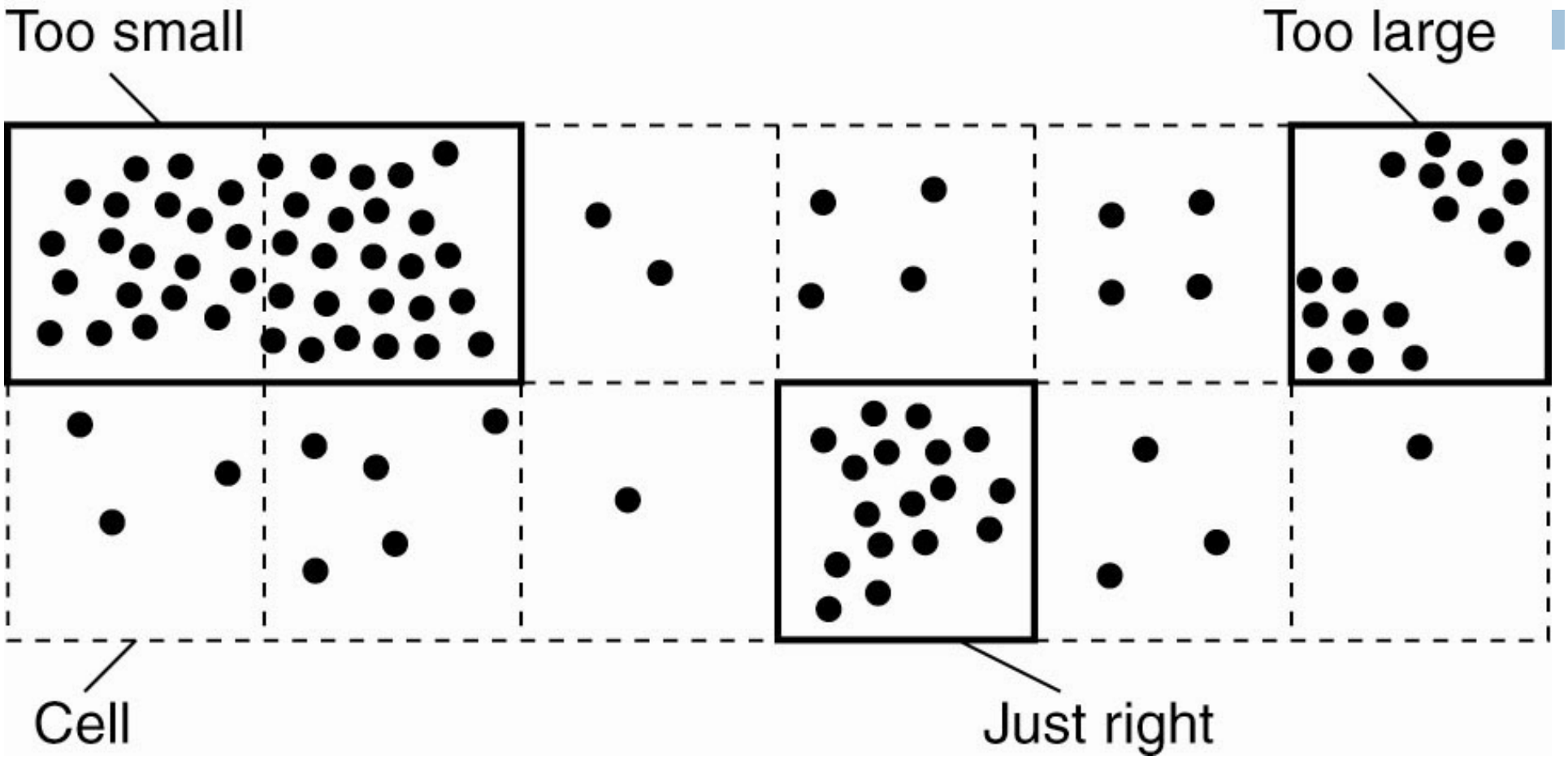
4.2. Quản lý nội dung

4.3. Phân phối nội dung

4.1. Quản lý máy chủ

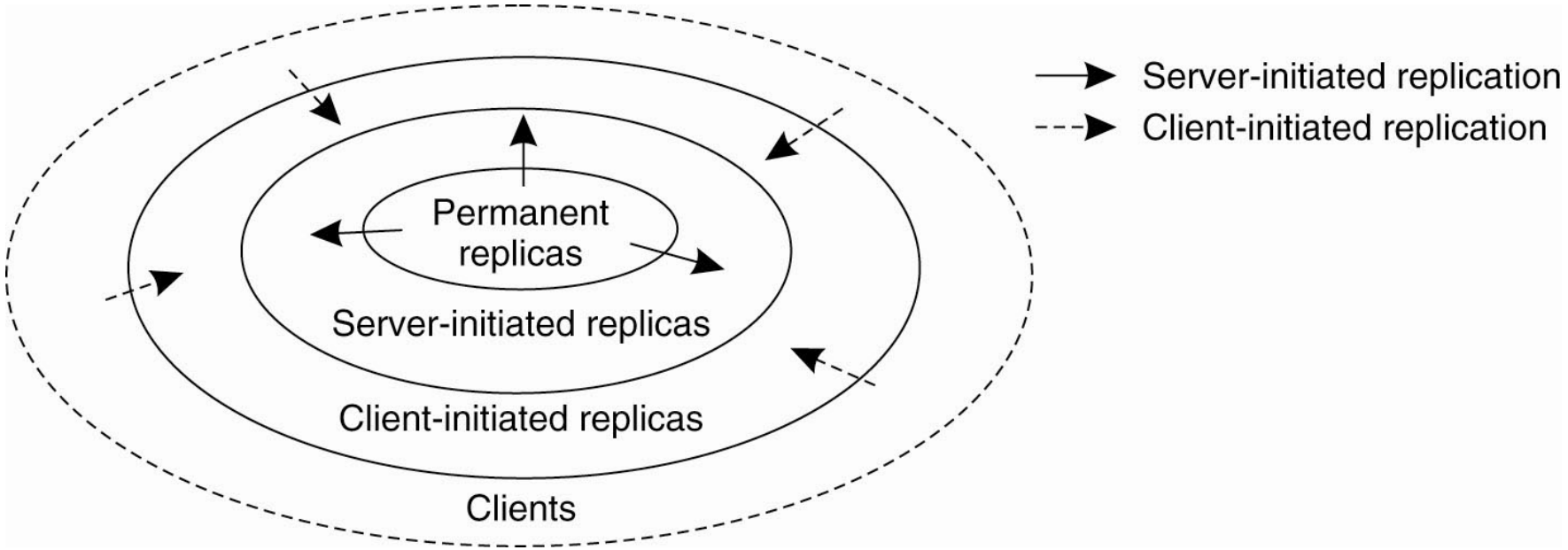
- Bài toán
  - Cho trước N vị trí đặt máy chủ
  - Xác định K trong N vị trí tối ưu để đặt các bản sao
- Giải pháp 1
  - Dựa vào Khoảng cách giữa Client và các bản sao
  - Xác định từng server.
- Giải pháp 2: không phụ thuộc vào vị trí client
  - Chia thành các cell - autonomous systems
  - Chọn AS lớn nhất
    - Đặt server ở vị trí có nhiều link nhất
  - Tiếp tục với các AS nhỏ hơn
- Độ phức tạp $O(N^2)$

Quản lý các bản sao-kích thước cell



Độ phức tạp O(Nx max(log N, k))

4.2. Quản lý nội dung



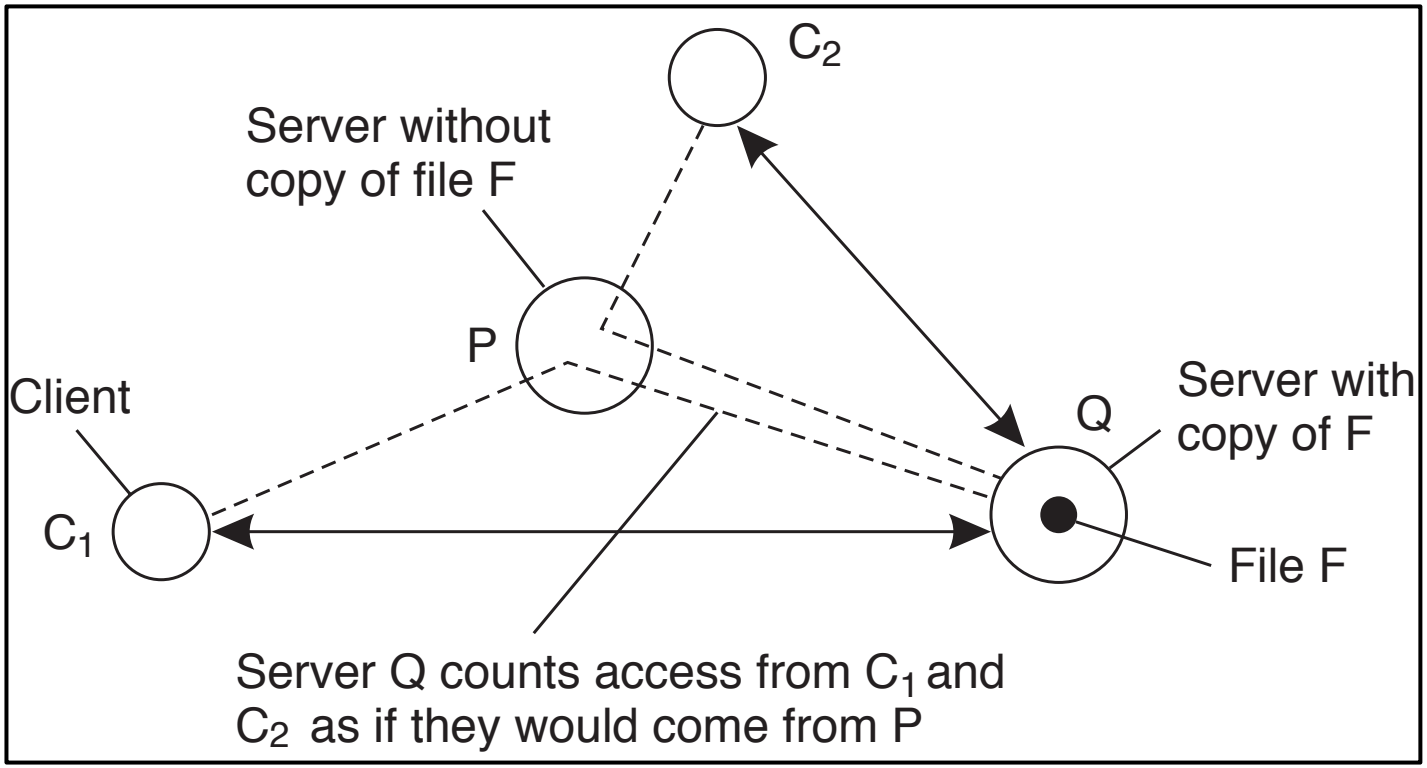
Sử dụng các bản sao cố định

- Là các bản sao tồn tại từ khi khởi động kho dữ liệu
- Số lượng các bản sao nhỏ
- Cách tổ chức 1
  - Dữ liệu được sao lưu trên các bản sao khác nhau
  - Khi có yêu cầu sử dụng dữ liệu, yêu cầu sẽ được chuyển đến một bản sao theo chiến thuật Roundrobin
- Cách tổ chức 2
  - Client chọn một trong các bản sao để truy cập
- Có thể dùng cho web, cho database
- Nguyên tắc chung: không chia sẻ tài nguyên giữa các bản sao

Bản sao kích hoạt bởi server

- Server đang hoạt động
  - Số lượng các yêu cầu tăng
  - Kích hoạt các bản sao ở các vị trí khác phụ thuộc theo yêu cầu
- Giảm tải cho bản sao (cũ)
- Cập nhật dữ liệu lên bản sao (mới) gần với client hơn

Bản sao kích hoạt bởi server



Kết hợp giữa khởi tạo server và bản sao bền vững?
Hosting services

Kích hoạt bởi client

- Caching
- Client quản lý cache, quyết định việc cập nhật cache
  - Xóa
  - Ghi
- Chính sách caching
- Có thể chia sẻ cache giữa các client

4.3. Phân phối nội dung

- Trạng thái/thao tác
- Pull/Push
- Unicast/Multicast

Trạng thái/thao tác

- Giải pháp cập nhật dữ liệu
  - Chỉ thông báo có cập nhật
    - Giảm thông tin cần truyền. Thích hợp cho trường hợp ghi nhiều - đọc ít
  - Truyền dữ liệu cập nhật
    - Thích hợp cho trường hợp đọc nhiều ghi ít
  - Truyền thao tác cập nhật
    - Các bản sao cần theo dõi trạng thái của dữ liệu
    - Chỉ cần truyền các tham số cần thiết cho thao tác cập nhật
- Trạng thái nhiều=> thao tác ít và ngược lại

Pull/Push

- Push: server sau khi cập nhật dữ liệu thông báo cho tất cả các client
  - Bản sao kích hoạt bởi server
  - Đảm bảo tính thống nhất cao
  - Tính tương tác yếu (vd khi client hoặc bản sao cần cập nhật dữ liệu)
  - Server cần có danh sách tất cả các client đang kết nối
- Pull: client khi cần dữ liệu sẽ hỏi server
  - Thường dùng cho client cache
  - Thích hợp cho ghi nhiều, đọc ít
  - Thời gian truy cập tăng (khi có cache miss)
- Mixed

Uni/multicast

- Multicasting:
  - Thích hợp trong trường hợp 1 bản sao muốn quảng bá cập nhật cho (N-1) bản sao khác trong 1 data store
  - Hiệu quả và tiết kiệm hơn gửi (N-1) lần phân biệt
  - Phù hợp với hướng tiếp cận push-based
  - Không phù hợp nếu các node đích lại thuộc 1 mạng LAN
- Unicasting:
  - Phù hợp với pull-based

5. Các giao thức đảm bảo thống nhất

5.1. Thống nhất liên tục
5.2. Thống nhất dựa trên bản sao primary
5.3. Replicated write
5.4. Cache coherence

5.1. Thống nhất liên tục

- Giới hạn sai lệch giá trị
- Giới hạn sai lệch thời gian
- Giới hạn sai lệch thứ tự thao tác

Giới hạn về sai lệch giá trị

- Xét một đơn vị dữ liệu x. Thao tác ghi W(x) có trọng số là giá trị cập nhật của x, ký hiệu weight(W(x))
- Tiến trình xuất phát của thao tác ghi được ký hiệu là origin(W(x))
- Mỗi server lưu trữ một log Li về các thao tác ghi được tiến hành trên server
- Gọi TW[i,j] là trọng số gộp của các thao tác ghi xuất phát từ server *j* và phổ biến được đến server *i*

$$TW[i,j] = \sum \{weight(W) \mid origin(W) = S_j \ \& \ W \in L_i\}$$

  - TW[k,k] là trọng số của các thao tác ghi trên k

Giới hạn về sai lệch giá trị

- Giá trị của x tại i

$$v(t) = v(0) + \sum_{k=1}^{N} TW[k,k]$$

$$v_i = v(0) + \sum_{k=1}^{N} TW[i,k]$$

$$v_i \leq v(t)$$

  - Cần xác định 1 ngưỡng sao cho:

$$v(t) - v_i \leq \delta_i$$

Giới hạn về sai lệch thời gian

- Có thể sử dụng thời gian cục bộ của tiến trình để đánh giá
  - Server S_k có thông tin vector clock RVC_k
  - Nếu thời gian $RVC_k[i]=T(i)$ => S_k đã nhìn thấy tất cả các thao tác ghi được cập nhật trên S_i đến thời điểm $T(i)$
  - $T(i)$ chỉ thời gian cục bộ của server *i*
  - Khi $T(k)-RVC_k[i]>$ delta => bỏ các thao tác có $T>RVC_k[i]$

Giới hạn về sai lệch thứ tự thao tác

- Mỗi replica có 1 hàng đợi các thao tác ghi
- Thứ tự toàn cục cần được xem xét
- Số lượng lớn nhất các thao tác ghi đang nằm trong hàng đợi
- Khi vượt quá số này, server sẽ dừng việc thực thi và sẽ thỏa thuận với các server khác về thứ tự

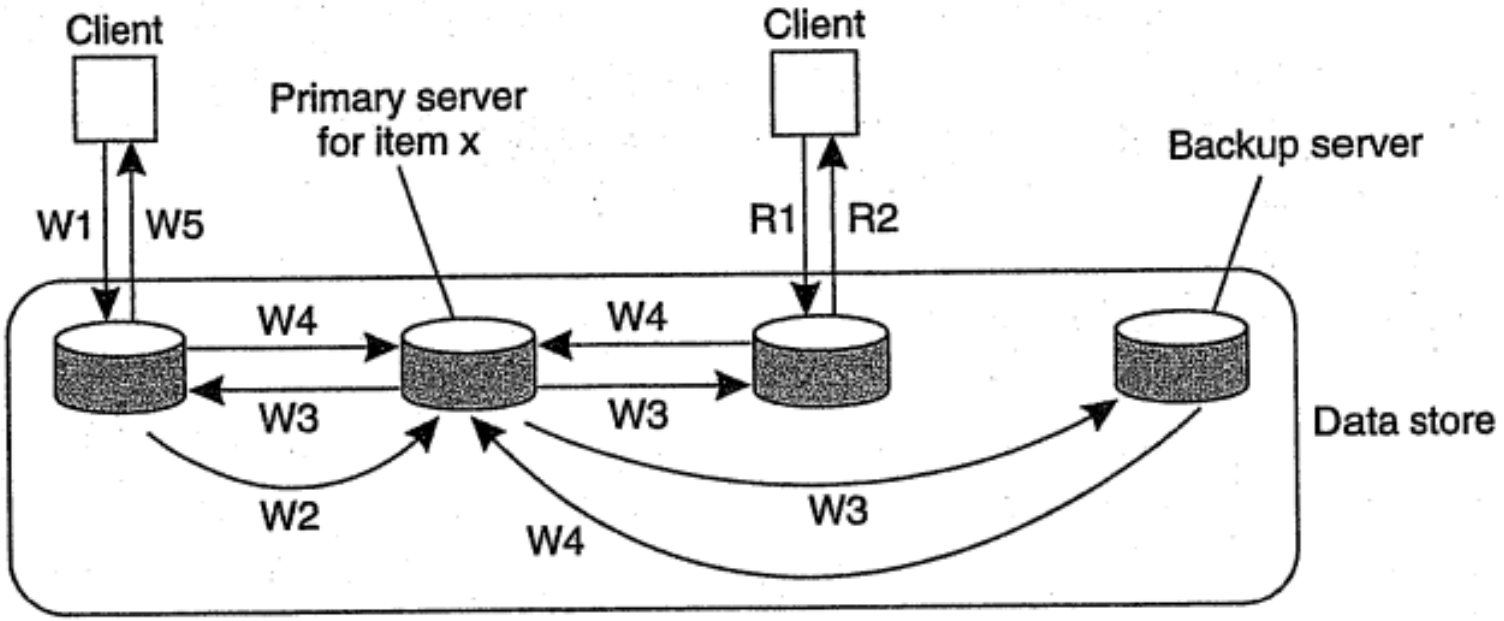
5.2. Các giao thức dựa vào bản sao primary (nguyên thủy)

- Mô hình thống nhất=> phức tạp
- Các nhà phát triển cần các mô hình đơn giản hơn
- Mỗi phần tử dữ liệu có một bản sao (primary) chịu trách nhiệm điều khiển các thao tác trên phần tử dữ liệu đó
  - Primary cố định (giao thức ghi từ xa)
  - Primary cục bộ (giao thức ghi cục bộ)

Giao thức ghi từ xa

- Tất cả các thao tác ghi được chuyển đến cho một server
- Thao tác đọc có thể được thực hiện cục bộ
- Thời gian cập nhật chậm
  - Thao tác cập nhật dừng=> không dừng=> độ tin cậy
- Đảm bảo được thống nhất tuần tự

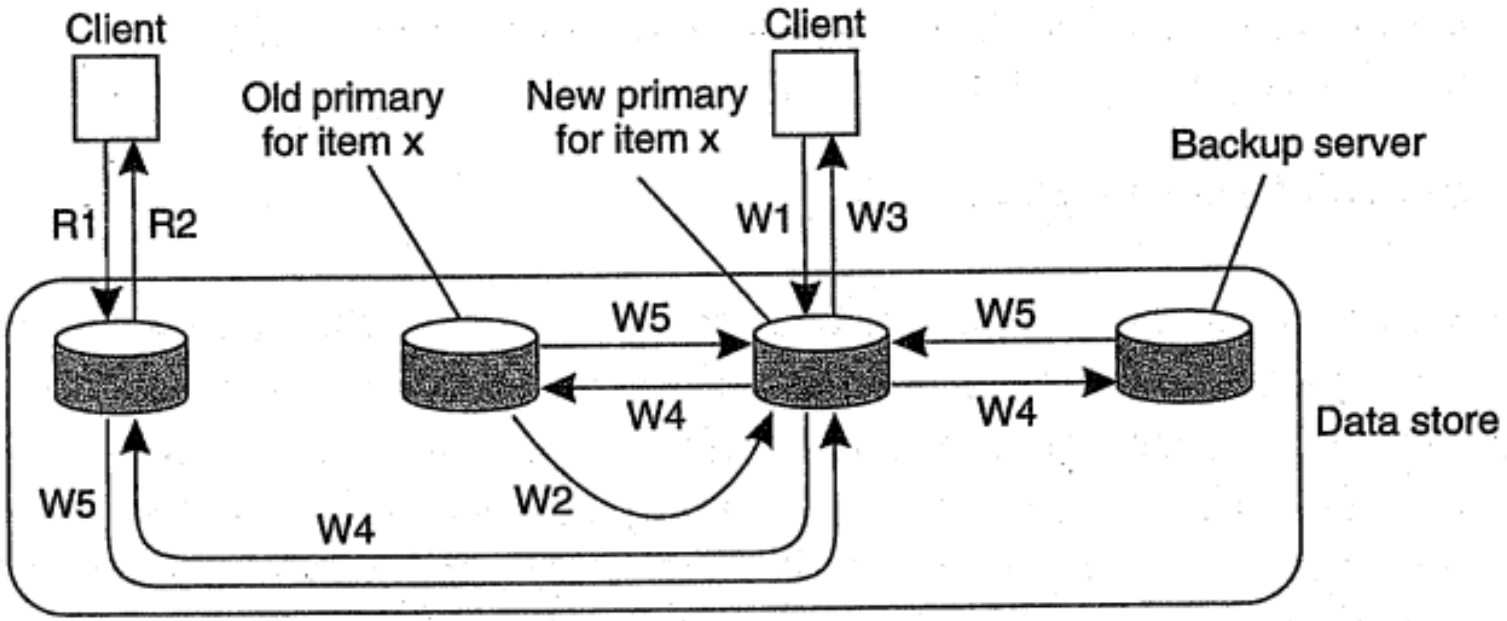
Giao thức ghi từ xa



W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

Giao thức ghi cục bộ



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

5.3. Ghi trên các bản sao (replicated write)

1. Sao lưu tích cực
2. Sao lưu dựa trên túc số (quorum)
3. Cache Coherence

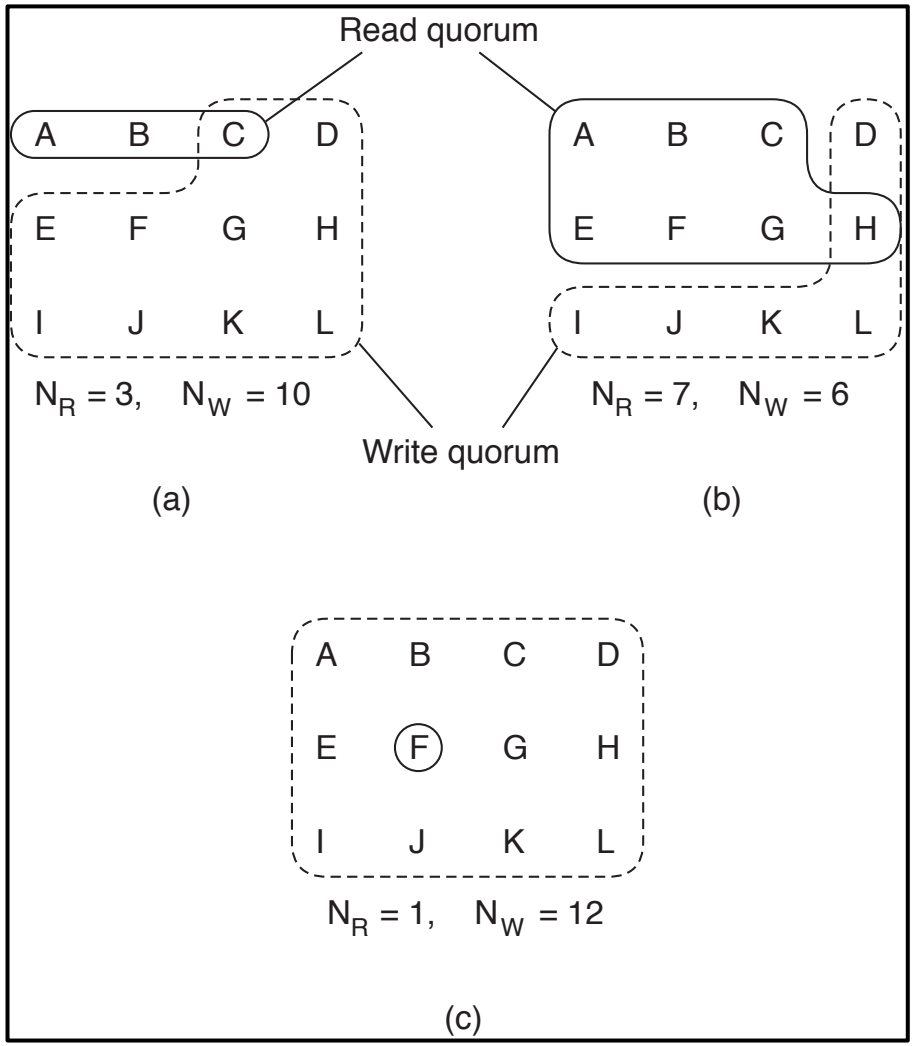
5.3.1. Sao lưu tích cực

- Một tiến trình chịu trách nhiệm phổ biến thao tác cập nhật đến tất cả các bản sao
- Cần có một cơ chế trật tự toàn cục (total ordered mechanism)
  - Có thể sử dụng thời gian logic (Lamport)=> không co giãn được
  - Có thể sử dụng sequencer đảm bảo trật tự toàn cục

5.3.2. Sao lưu dựa trên túc số (quorum)

- Sao lưu theo vote-Giải thuật quorum (túc số)
  - Để có thống nhất mạnh=> cần cập nhật tất cả các bản sao
  - Sau khi cập nhật với chi phí tốn kém=> không phải tất cả các bản sao đều được đọc=> chi phí vô ích
  - Liệu có giảm được số lượng bản sao cần cập nhật (ngay)?
- Khi đọc dữ liệu
  - có khả năng đọc phải dữ liệu cũ
  - Đọc thêm dữ liệu ở một số bản sao khác=> lựa chọn bản sao có dữ liệu mới nhất
- Write Quorum; Read Quorum
  1. $N_R + N_W > N$ → tránh read-write conflict
  2. $N_W > N/2$ → tránh write-write conflict

Ví dụ về quorum



5.4. Giao thức đảm bảo sự phù hợp của bộ nhớ đệm-Cache Coherence

- Caching là một trường hợp đặc biệt của sao lưu
- Các giao thức caching=> sự phù hợp của nội dung cache
  - Chính sách phát hiện sai khác: tĩnh/động
- Đảm bảo thống nhất giữa cache và server
  - Không cache, luôn luôn cache, mix
  - Invalidation message
- Cập nhật thông tin từ server
  - Quảng bá từ server
  - Yêu cầu từ client

Chính sách phát hiện sai khác

- Đảm bảo đủ dữ liệu chính xác cho thao tác
- Tĩnh: trước khi thực hiện
- Động: trong khi thực hiện
  - Có thể thực hiện trước khi tiến hành thao tác: đảm bảo có đủ dữ liệu chính xác mới thực hiện
  - Có thể thực hiện trong khi đang tiến hành thao tác
  - Có thể thực hiện khi thao tác đã được tiến hành xong. Nếu phát hiện một trong các dữ liệu đầu vào sai=> thực hiện lại

Công cụ sao lưu Syncthing

Syncthing

- Công cụ mã nguồn mở
- Đồng bộ tệp/thư mục giữa các máy tính trong mạng
- Truyền dữ liệu trực tiếp giữa các hệ thống
- Không giống như Dropbox, công cụ Syncthing giúp người dùng hoàn toàn làm chủ dữ liệu, không có sự góp mặt của bên thứ 3 (P2P synchronization tool).

Secure & Private

- **Private.** None of your data is ever stored anywhere else than on your computers. There is no central server that might be compromised, legally or illegally.
- **Encrypted.** All communication is secured using TLS. The encryption used includes perfect forward secrecy to prevent any eavesdropper from ever gaining access to your data.
- **Authenticated.** Every node is identified by a strong cryptographic certificate. Only nodes you have explicitly allowed can connect to your cluster.

5.5. Các giao thức thống nhất hướng client

- Các thao tác ghi có một mã số được đặt bởi bản sao thực hiện thao tác
- Mỗi client có 2 tập hợp các thao tác ghi
  - Tập các thao tác ghi tương ứng với các lệnh đọc đã được thực hiện
  - Tập các thao tác ghi được thực hiện bởi client

Đọc đơn điệu

- Khi chương trình client đọc dữ liệu
- Kiểm tra liệu tất cả các thao tác ghi (tập hợp 1) có liên quan có được thực hiện cục bộ hay không
- Nếu không-> kết nối với các bản sao tương ứng
  - Thực hiện thao tác đọc
  - Bổ sung thao tác ghi vào tập hợp 2
- Cần có liên kết giữa ID thao tác ghi và địa chỉ của bản sao thực hiện

Vấn đề hiệu năng

- Tập hợp 1=> có thể rất lớn
- Tạo ra các tập hợp con cho mỗi ứng dụng=> dữ liệu phiên
- Các thao tác ghi từ một server được lưu trong session bằng thời gian cập nhật cuối cùng
- So sánh giữa thời gian của thao tác tương ứng với thời gian cập nhật cuối cùng=> quyết định có phải thực hiện thao tác ở xa hay không