

# Contents

C1. TỔNG QUAN VỀ HỆ THỐNG PHÂN TÁN .....	6
Định nghĩa các HTPT .....	6
Phân loại các hệ thống phân tán.....	6
- Các hệ thống tính toán phân tán.....	6
<b>Các hệ thống thông tin phân tán .....</b>	8
Hệ thống di động.....	8
Mục tiêu hệ thống phân tán.....	8
Đảm bảo khả năng sẵn sàng của tài nguyên .....	9
Trong suốt đối với người sử dụng.....	9
Tính mở của hệ thống: .....	9
Qui mô hệ thống .....	9
Những điểm cần chú ý .....	9
Mục tiêu xây dựng htPT .....	9
Mong muốn ng dùng.....	9
Tính trong suốt.....	9
Phân loại tính trong suốt .....	10
Mức độ trong suốt.....	10
Toàn vẹn .....	11
Hiệu năng.....	11
Giải pháp : .....	11
Đặc điểm tt phi tập trung .....	11
Tính Chịu lỗi (5 chapter 1) .....	11
Tính mở.....	11
Các hàm nguyên thủy .....	12
Lựa chọn tiêu chí phân tán :.....	12
Kiến trúc HT PT .....	12
Mô hình phân tầng .....	13
Mô hình đối tượng phân tán.....	13
Mô hình khách/chủ .....	14
Mô hình kênh sự kiện .....	14
Mô hình dữ liệu tập trung .....	15
Kiến trúc HT .....	15
Kiến trúc tập trung .....	16
Kiến trúc không tập chung.....	16

<b>Chương 2 : VÂN ĐÈ VÀ GIẢI PHÁP TRONG HỆ THỐNG PHÂN TÁN .....</b>	<b>17</b>
2.1 Truyền thông.....	17
Các giao thức mức thấp .....	18
Phân loại truyền thông .....	18
Gọi thủ tục từ xa .....	18
Truyền thông hướn thong diep.....	21
Các phuong phap .....	21
Truyền thông hướng luồng .....	21
Truyền thông theo nhóm.....	21
<b>Chương 3 :Đặt tên , định danh ,tìm kiếm .....</b>	<b>22</b>
Các phương pháp Đặt tên và các giải pháp tìm kiếm .....	22
<b>Đặt tên phi cấu trúc .....</b>	22
Đặt tên có cấu trúc : .....	24
Không gian tên.....	24
<b>Chương 4 : Đồng bộ .....</b>	<b>24</b>
Giải thuật Cristiana .....	24
Giải thuật Berkeley .....	25
Giải thuật trung bình .....	27
Đồng bộ thời gian trong mặc không dây .....	27
Thời gian logic và các đồng hồ logic.....	28
Đồng hồ logic Lamport.....	28
Đồng hồ vector.....	28
Các trạng thái toàn cục.....	29
Các giải thuật loại trừ tương hỗ phân tán.....	29
Giải thuật tập trung .....	29
<b>Giải thuật không tập trung .....</b>	30
<b>Giải thuật phân tán.....</b>	30
Giải thuật thẻ bài.....	32
So sánh các giải thuật loại trừ .....	32
Các giải thuật bầu chọn.....	33
<b>Giải thuật nổi bọt,nổi trội.....</b>	33
<b>Thuật toán vòng .....</b>	34
<b>Chương 5 :tiến trình,ảo hóa , di trú mã trong các HTPT .....</b>	<b>35</b>
Tiến trình.....	35
Lời gọi HT .....	36
Ván đè khi SD tiến trình .....	36

Khái niệm luồng.....	37
Cài đặt luồng .....	37
Luồng trong HTPT.....	38
Ảo hóa.....	38
Vai trò ảo hóa.....	38
Kiến trúc máy ảo .....	39
Phương pháp ảo hóa.....	40
Máy khách.....	41
Ứng dụng mạng vs Gt riêng,chung .....	41
Máy chủ .....	41
Di trú mã .....	41
Các giải pháp di trú mã .....	42
Di trú và tài nguyên cục bộ.....	43
Di trú trong hệ thống không đồng nhất.....	44
<b>C6 QUẢN TRỊ GIAO TÁC VÀ ĐIỀU KHIỂN TƯƠNG TRANH.....</b>	<b>44</b>
Khái niệm giao tác .....	44
Giao tác phẳng .....	45
Các giao tác lồng nhau .....	45
Hoạt động của giao tác.....	46
Quá trình sống giao tác .....	46
Vấn đề tương tranh .....	47
-Mất mát dl.....	47
- Kết quả k đồng nhất.....	47
Biện pháp tuần tự hóa .....	49
Các luật xung đột .....	50
Xen kẽ thực hiện ( <b>không tuần tự</b> ) .....	50
Phục hồi Dl ban đầu khi hủy thực hiện.....	50
cÁc vấn đề phục hồi.....	50
Các phương pháp điều khiển tương tranh.....	51
Khóa loại trừ .....	52
Điều khiển tương tranh bằng khóa.....	53
Định nghĩa khóa.....	53
Khái niệm khóa: .....	53
2. Kỹ thuật khóa đơn giản.....	54
3. Kỹ thuật khóa đọc/ viết ( readlock/ writelock) .....	55
Livelock(Khóa chờ) .....	58

Quản lý khóa .....	59
Khóa Chết .....	60
Các kĩ thuật phòng ngừa deadlock.....	61
Tăng tính tương tranh .....	62
Khóa phân cấp .....	62
Nhược điểm cơ chế khóa .....	62
Điều khiển tương tranh lạc quan.....	62
Các luật đọc ghi .....	63
Phê chuẩn giao tác .....	63
Phê chuẩn ngc .....	64
Phê chuẩn xuôi.....	65
So sánh 2 giao tác phê chuẩn .....	65
Điều khiển tương tranh dựa trên nhãn thời gian .....	65
So sánh các pp dk tuong tranh .....	70
Chương 7 :PHỤC HỒI VÀ TÍNH CHỊU LỖI.....	70
Khái niệm tính chịu lỗi .....	70
Phân loại lỗi .....	70
Các biện pháp đảm bảo tính chịu lỗi.....	72
Che giấu lỗi bằng biện pháp dư thừa .....	72
Nhóm tiên trình bền bỉ .....	73
Che giấu lỗi và nhân bản.....	75
Đồng thuận trong các hệ thống lỗi.....	76
Phát hiện lỗi .....	79
Truyền thông khách chủ tin cậy.....	79
Các tình huống lỗi khi gọi thủ tục từ xa.....	80
Truyền thông nhóm tin cậy .....	83
Chương 8 : Tính nhất quán và nhân bản.....	93
Tính nhất quán và nhân bản .....	93
NHÂN BẢN DỮ LIỆU .....	93
Hai cách tiếp cận.....	94
Mô hình lấy DL làm trung tâm .....	95
Tính nhất quán liên tục .....	95
Mô hình nhất quán chặt .....	99
Mô hình nhất quán tuần tự .....	99
Mô hình nhất quán nhân quả.....	101
Mô hình nhất quán yếu .....	102

Mô hình nhất quá bảng dữ liệu .....	102
Mô hình nhất quán mục dữ liệu .....	103
Mô hình lấy MK làm trung tâm .....	105
Mô hình nhất quán sau cùng .....	105
Mô hình nhất quán đọc đều .....	107
Mô hình nhất quán ghi đều .....	109
Mô hình nhất quán đọc kết quả ghi .....	109
Mô hình nhất quán ghi theo sau đọc .....	110
Quản lý bản sao .....	110
Sắp xếp bản sao máy chủ .....	111
Các Gt .....	117
Bài tập youtube .....	117
Dạng 1 gt bekerley .....	117
Dạng bài Vector thời gian .....	118
GT nhãn tgian lamport .....	118
GT Cristianas .....	119
Bài tập Web .....	119
Trình bày nguyên lý hoạt động của máy điều hành ảo (Virtual Machine Monitor) .....	145
Trình bày nguyên lý chia sẻ kênh (bus) trong các máy tính nhiều bộ vi xử lý (multiprocessor). .	145
Trình bày các mô hình cung cấp dịch vụ tập tin (File Service .....	146
Nêu vai trò của các biện pháp đảm bảo an toàn thông tin trong các hệ thống phân tán. ....	147
Tính linh hoạt của hệ thống thể hiện ở những điểm nào?.....	148
Nêu vai trò và ứng dụng của lớp trung gian (Midleware) trong hệ thống phân tán .....	149
Nêu những hạn chế của việc đặt thời gian quá hạn (timeout) sử dụng trong việc truyền tin, trình bày biện pháp khắc phục.....	149
Liệt kê và phân tích những nguyên nhân gây lỗi trong các hệ thống phân tán.....	150
Nêu những điểm khác biệt giữa khái niệm liên kết lỏng và liên kết chặt trong các hệ thống phân tán.....	151
Nêu những ưu điểm và nhược điểm của hệ thống phân tán so với hệ thống tập trung.....	152
Phân tích quá trình phân giải tên miền mail.yahoo.com thành địa chỉ IP .....	152
Trình bày những điểm khác biệt cơ bản giữa hệ điều hành phân tán và hệ điều hành mạng .....	153
Trình bày giao thức LDAP. .....	154
Phân tích qui trình hoạt động của kiến trúc ngang hàng khi xây dựng hệ thống phân tán. ....	156
Viết ứng dụng bằng ngôn ngữ C# minh họa truyền thông điệp sử dụng hàng đợi .....	157
Phân tích các chỉ số đảm bảo chất lượng truyền dữ liệu dạng luồng (stream). .....	158
Trình bày qui trình viết ứng dụng khách/chủ.....	158

Trình bày cơ chế đảm bảo tính trong suốt khi gọi thủ tục từ xa (RPC).....	159
Phân biệt dự phòng nguội và dự phòng nóng trong các hệ thống phân tán.....	164
Mô hình nhất quán đi ra .....	177
Mô hình nhất quán chặt .....	178
Mô hình nhất quán đọc ghi kết quả.....	180
Các mô hình dựa trên kiến trúc hướng dịch vụ.....	181
Khái niệm chung về hệ thống phân tán.....	183
Ảo hóa.....	184
Các giao thức đảm bảo nhất quán .....	186
Truyền thông khách chủ tin cậy.....	187
Truyền thông nhóm tin cậy .....	188
Giới thiệu về kiến trúc hướng dịch vụ .....	189
Mô hình nhất quán .....	190
Trình bày các tiêu chí về hiệu năng đối với hệ thống phân tán. ....	192
Các tiêu chí về hiệu năng là:.....	192
Single Instruction Multiple Data (SIMD) và Multiple Instruction Multiple Data (MIMD) là hai kiến trúc tính toán song song khác nhau sử dụng nhiều bộ xử lý và đôi khi là nhiều máy tính để xử lý dữ liệu. ....	194

## C1. TỔNG QUAN VỀ HỆ THỐNG PHÂN TÁN

### Định nghĩa các HTPT

HTPT = Mang + Các DV hệ thống

Hệ thống phân tán bao gồm các máy tính và các thiết bị khác như thiết bị cầm tay, điện thoại di động thông minh... được kết nối với nhau để thực hiện nhiệm vụ tính toán

Yêu cầu : xây dựng các ứng dụng phân tán

- **Yêu cầu tính toán phân tán:**
- **Yêu cầu về khả năng xử lý lõi**
- **Chia sẻ tài nguyên**

### Phân loại các hệ thống phân tán

Cơ sở để phân tích và lựa chọn giải pháp phù hợp để phát triển hệ thống

- Các hệ thống tính toán phân tán  
phải đảm bảo vấn đề hiệu năng, tính toán phân tán

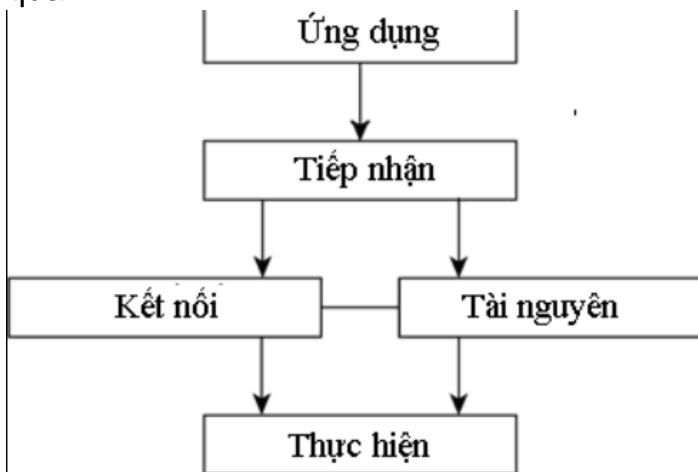
#### ***Hệ thống tính toán cụm***

- tính toán cụm sử dụng **kỹ thuật xử lý song song** trên nhiều máy tính để tăng hiệu năng xử lý cho các tác vụ

- các máy tính cá nhân được cài đặt một loại hệ điều hành và kết nối với nhau trong mạng tốc độ cao

### Hệ thống tính toán mục lướt

- không đòi hỏi tính đồng nhất của tất cả các nút
- mỗi thành viên có thể khác về cả phần cứng lẫn hệ điều hành và các chính sách quản lý
- Vấn đề cốt lõi **việc quản lý tài nguyên của các cơ quan khác nhau** người dùng thuộc về một cơ quan ảo thì có quyền truy nhập đến các tài nguyên của cơ quan ảo đó

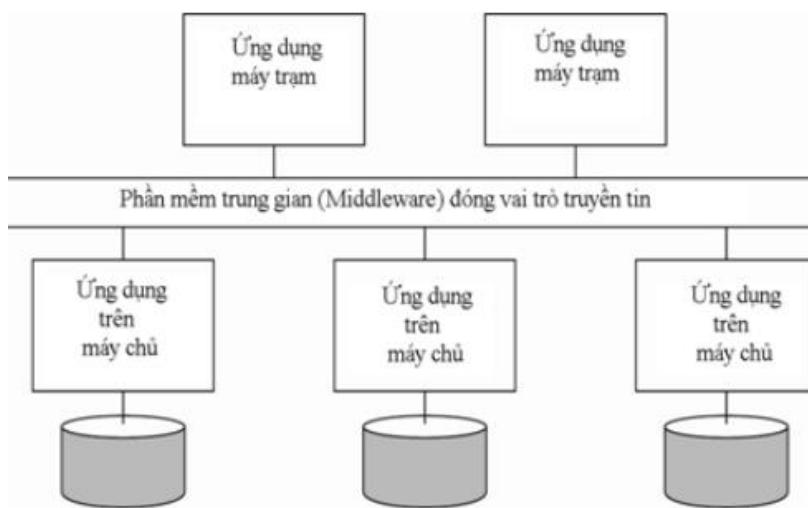


- **Tầng ứng dụng** bao gồm các ứng dụng vận hành bên trong cơ quan ảo và sử dụng môi trường điện toán lưới.
- **Tầng tiếp nhận:** Xử lý các yêu cầu truy nhập đến nhiều tài nguyên khác nhau, thường cung cấp các chức năng như: **thăm dò, định vị, lập lịch truy nhập, nhân bản tài nguyên...** Các giao thức thuộc tầng này khá nhiều, để đảm bảo cung cấp dịch vụ theo yêu cầu của tầng ứng dụng nên chúng thường **không phải là những giao thức đã được chuẩn hóa**.
- **Tầng kết nối:** Bao gồm các giao thức truyền thông để hỗ trợ cho các giao tác lưới bao trùm toàn bộ các tài nguyên, ví dụ các giao thức truy cập để di chuyển tài nguyên hoặc đơn giản chỉ là truy cập tài nguyên từ một vị trí nào đó. Tầng kết nối sẽ phải bao gồm các **các giao thức bảo mật, tính năng bảo mật có thể cho một tài khoản và cũng có thể cho một ứng dụng**.
- **Tầng tài nguyên:** Quản lý tài nguyên đơn lẻ, nó sử dụng các chức năng do tầng kết nối cung cấp và gọi trực tiếp các giao diện tầng kết cấu cung cấp để thực hiện các chức năng điều khiển truy nhập, ví dụ các chức năng thiết lập cấu hình tài nguyên, khởi tạo tiến trình đọc/ghi dữ liệu.
- **Tầng thực hiện:** Cung cấp giao diện để truy nhập tài nguyên cục bộ, các giao diện này được xây dựng để có thể thích ứng với việc cho phép chia sẻ tài nguyên bên trong một cơ quan ảo, nó thường cung cấp các chức năng để truy vấn trạng thái và khả năng của tài nguyên, các chức năng quản lý tài nguyên thực.

tầng tiếp nhận, tầng kết nối và tầng tài nguyên thường được gộp lại và gọi chung là **tầng trung gian**, nó có nhiệm vụ quản lý và cung cấp chức năng truy nhập trong suốt đến tất cả các tài nguyên phân bố trên các trang mạng khác nhau

## Các hệ thống thông tin phân tán

một yêu cầu được đưa ra từ **phía máy khách đến các máy chủ** dữ liệu thì yêu cầu đó phải được **thực thi trên tất cả các máy chủ** hoặc chỉ cần một máy chủ không thực thi được yêu cầu của máy khách thì tất cả các máy chủ khác cũng không được phép thực thi yêu cầu này.



Các Phần mềm tách các phần riêng biệt : Nên khi tích hợp cần cho chúng kết nối đc vs nhau  
-->**dẫn đến ngành công nghiệp lớp tích hợp ứng dụng doanh nghiệp**

## Hệ thống di động

- 2 hệ thống trước **đều có hình trạng cố định, các nút mạng cố định tại một vị trí địa lý và đường truyền kết nối mạng chất lượng cao tương đối ổn định.** thực hiện bằng nhiều kỹ thuật khác nhau nhằm đạt được tính trong suốt phân tán.
- Tuy nhiên khi có thiết bị di động, thiết bị nhúng gây cản trở **hình trạng lẩn tốc độ truyền dẫn.**
- Do di động có vị trí k ổn định kết nối mạng k dây còn gọi là **hệ thống lan tỏa phân tán**

Grimm đã đưa ra các yêu cầu sau cho các hệ thống lan tỏa phân tán:

- **Bao quát những thay đổi ngữ cảnh:** Thiết bị phải liên tục nhận biết được môi trường có thể thay đổi bất kỳ thời gian nào, ví dụ khi phát hiện thấy mất kết nối mạng thì thiết bị sẽ tự động tìm mạng khác thay thế.
- **Cung cấp giao diện cấu hình mặc định:** Mỗi người dùng có thói quen riêng biệt, do đó cần phải cung cấp giao diện cấu hình sao cho đơn giản nhất phù hợp với tất cả mọi người hoặc một cấu hình được cài đặt tự động.
- **Tự động nhận biết chia sẻ tài nguyên:** Một khía cạnh quan trọng của hệ thống lan tỏa là các thiết bị tham gia hệ thống theo thứ tự truy nhập thông tin, điều này đòi hỏi phải cung cấp các phương tiện để dễ dàng đọc, lưu trữ, quản lý và chia sẻ thông tin.

## Mục tiêu hệ thống phân tán

<http://www.hocvienmang.com/admin/pages/admin/elearn/ViewBookParts.aspx?VaazBrno=D8GknQDo4W8%3D>

## **Đảm bảo khả năng sẵn sàng của tài nguyên**

kết nối người sử dụng với tài nguyên hệ thống, người sử dụng được tiếp cận tài nguyên theo cách dễ dàng nhất mà không phụ thuộc vị trí địa lý của người đó

Trong suốt đối với người sử dụng

Tính mở của hệ thống:

- khả năng cung cấp các dịch vụ theo qui tắc chuẩn mô tả cú pháp và ngữ nghĩa của các dịch vụ

Qui mô hệ thống

o **dễ dàng thêm các máy tính mà không cần phải sửa đổi hệ thống**, như vậy chúng ta có thể mở rộng hay thu hẹp hệ thống phân tán theo yêu cầu thực tế, đây là **một đặc tính quan trọng nhất** của hệ thống phân tán

Những điểm cần chú ý

- Mạng đáng tin cậy.
- Mạng đã được bảo mật.
- Mạng là hệ thống đồng nhất.
- Hình trạng mạng không bao giờ thay đổi.
- Độ trễ bằng không.
- Băng thông vô hạn.
- Chi phí vận chuyển bằng không.
- Chỉ có một người quản trị.

## **Mục tiêu xây dựng htPT**

- tính thân thiện : Giao diện ng dùng thân thiện, ng dùng ít phải quản lý tài nguyên
- Hiệu năng : Khắc phục sự chậm trễ trong tin liên lạc, các vấn đề vật lý, nghẽn mạng
- Linh hoạt : Dễ dàng phát triển dịch vụ, có k/n mở rộng và di chuyển, k/n module hóa, k/n tương tác
- Nhất quán : Thông nhất trong sd dvu, DL trên toàn ht
- Chịu lỗi : Tự khởi tạo lại tuển trình tốt nhất, đồng thời đảm bảo mất mát nhỏ nhất:
- AT và BMTT : Thông tin đáng tin cậy, bảo mật tốt, dkien truy cập

## **Mong muốn ng dùng**

Tính trong suốt

Toàn vẹn : Thông nhất toàn ht

Hiệu năng

Chịu lỗi

Tính mở : Giao tiếp vs ht khav

Bảo trì/nâng cấp

## **Tính trong suốt**

Che giấu sự thật các tiến trình và tài nguyên phân tán trên nhiều máy tính, một hệ thống phân tán có thể tự trình diễn cho người dùng và các ứng dụng như thể đang chạy trên một máy tính, đó gọi là tính trong suốt.

Tính trong suốt coi như 1 GD chuyển đổi mà ở đó ng dùng thao tác vs HT bằng bộ lệnh dc chuẩn hóa

Tính trong suốt đối với người sử dụng nhằm che giấu vị trí thực của tài nguyên, người sử dụng không biết tài nguyên ở đâu và xử lý trên máy tính nào. Hệ thống phân tán gồm nhiều loại trong suốt, người thiết kế hệ thống cần phải trả lời cho câu hỏi cần thiết phải trong suốt hay không và nếu có thì trong suốt ở mức độ nào.

## Phân loại tính trong suốt

ISO 1995 tính trong suốt bao gồm các loại sau:

- Truy nhập: Che giấu việc thể hiện dữ liệu và cách truy nhập tài nguyên.
- Vị trí: Che giấu nơi đặt tài nguyên.
- Di trú: Che giấu việc tài nguyên có thể chuyển sang vị trí khác
- Đặt lại vị trí: Che giấu tài nguyên có thể di chuyển sang vị trí khác mà không làm gián đoạn hoạt động của hệ thống.
- Nhân bản: Che giấu nhân bản tài nguyên.
- Tương tranh: Che giấu việc chia sẻ tài nguyên cho nhiều người đồng thời sử dụng.
- Lỗi: Che giấu lỗi và việc phục hồi tài nguyên.

### CÁC ĐẶC TÍNH TRONG SUỐT

- *Trong suốt truy cập:* cho phép các đối tượng cục bộ, hay từ xa đều được truy cập bằng các hoạt động giống nhau.
- *Trong suốt vị trí:* các đối tượng sử dụng dịch vụ không cần biết vị trí cài đặt của các đối tượng cung cấp dịch vụ.
- *Trong suốt nhân bản:* cho phép nhiều bản sao của các đối tượng được sử dụng để tăng độ tin cậy và hiệu năng của hệ thống mà không cho người sử dụng thấy được sự nhân bản đó.
- *Trong suốt tương tranh:* cho phép nhiều xử lý được thực hiện đồng thời sử dụng thông tin chia sẻ mà không xảy ra xung đột.
- *Trong suốt lỗi:* cung cấp khả năng chịu lỗi để người sử dụng có thể hoàn thành được công việc ngay cả khi hệ thống gặp lỗi về phần cứng hoặc phần mềm

### CÁC ĐẶC TÍNH TRONG SUỐT

- *Trong suốt di chuyển:* cho phép di chuyển các đối tượng trong hệ thống mà không ảnh hưởng đến hoạt động của người sử dụng. Khả năng di chuyển thường được thể hiện khi các đối tượng thành phần trong hệ thống chuyển dịch vị trí khi chuyển sang cơ chế dự phòng để khắc phục lỗi tại máy tính chính, hoặc nhằm mục đích cẩn bắng tái.
- *Trong suốt hiệu năng:* cho phép cấu hình lại hệ thống để tăng hiệu năng.
- *Đồng nhất:* Dữ liệu được đồng nhất giữa bộ nhớ và thiết bị lưu trữ ngoài
- *Trong suốt quy mô:* cho phép các thành phần trong hệ thống có thể thay đổi quy mô mà không ảnh hưởng đến cấu trúc của hệ thống cũng như hoạt động của hệ thống.

## Mức độ trong suốt

Che giấu quá mức không phải là điều tốt

mỗi cập nhật thực hiện trên một bản sao thì sẽ phải lan tỏa đến tất cả các bản sao khác, như vậy để đảm bảo tính nhất quán sẽ thì thời gian thực hiện sẽ lâu hơn, điều này không nên che giấu người sử dụng, có thể hiển thị thông báo giao dịch đang được thực hiện hoặc thời gian dự kiến hoàn thành để người sử dụng yên tâm chờ đợi

## Toàn vẹn

- Truy cập Ht
- Nhận bản dl
- 
- Lưu trữ dl

## Hiệu năng

Vấn đề : Khi tăng số lượng mt

- 1 máy chủ thực hiện hết cv
- Thuật toán tập trung

Giải pháp :

- Xử lý tương trang
- Xử lý //
- Tìm điểm tắc nghẽn
- Phân tải xl

Kỹ thuật : Xử lý qui mô lớn (tiền xl trên may khách)

## Đặc điểm tt phi tập trung

- Không áng huống khi 1 máy lỗi
- Đưa ra quyết định dựa trên thông tin cục bộ
- Không có mt nào có toàn bộ tt của cả ht

## Tính Chịu lỗi (5 chapter 1)

Đánh giá lỗi

- Phát hiện lỗi
- Lưu viết
- Tự sửa lỗi

Giải pháp :

Tăng độ dư thừa tài nguyên pcung

Tăng khả năng phục hồi cho pm : Pm tự phát hiện sửa lỗi phục hồi về trạng thái bd

Dự phòng người : ko làm gì, có sự cố chuyển sang ht dự phòng

Dự phòng nóng : Đảm bảo ht k bị gián đoạn

## Tính mở

Xđ ht có mở rộng đc k

...

Pm có tính mở khi :

Đề tương tác

Dễ chuyển mang

## Các hàm nguyên thủy

Begin Transaction :

End transaction

RollBack

Read

Write, Commit

Các thuộc tính đặc trưng

- Atomic( nguyên tử) : giao tác k chia nhỏ hơn
- Consistent (nhất quán) : giao tác k vi phạm tính bất biến
- Isolated(Cách ly )
- Durable( có thời hạn)

## Lựa chọn tiêu chí phân tán :

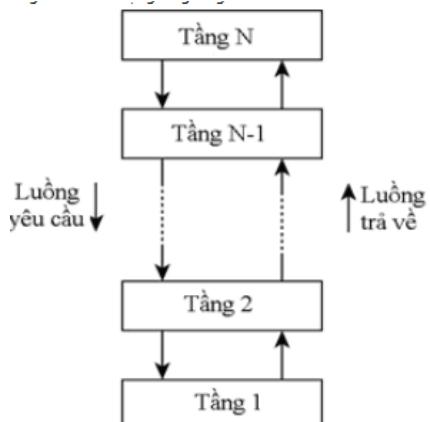
- PT theo chức năng
- PT theo nhu cầu phân tải xl ( phân tải tĩnh, phân tải động)
- PT để đề phòng sự cố
- PT dữ liệu

## Kiến trúc HT PT

hệ thống phân tán gồm bốn mô hình sau:

- Mô hình phân tầng
- Mô hình dựa trên đối tượng
- Mô hình dựa trên sự kiện
- Mô hình dữ liệu tập trung

## Mô hình phân tầng



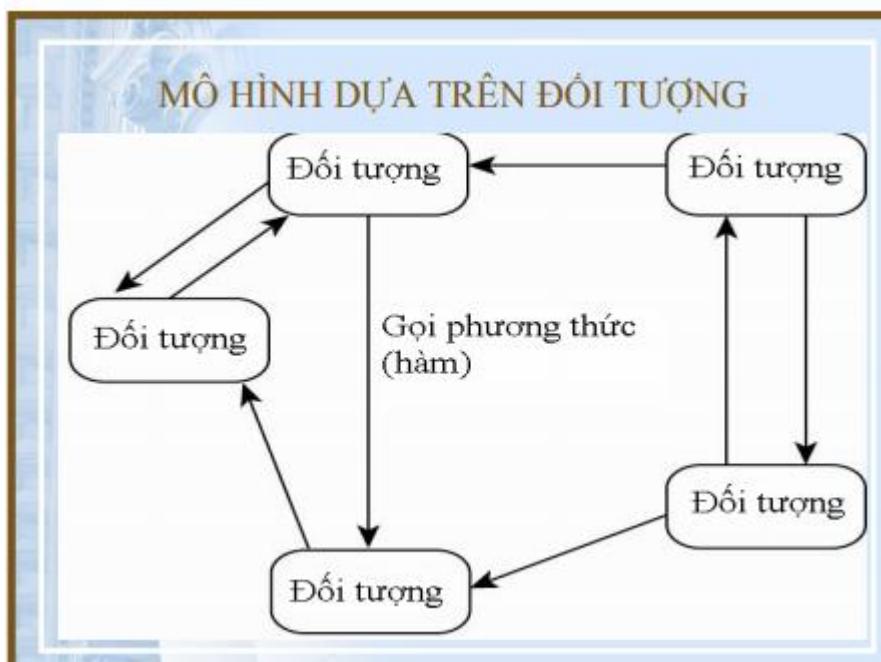
Hình 1.5 Mô hình phân tầng

tầng trên gọi các thành phần tầng dưới liền kề

Yêu cầu của bên gửi sẽ được chuyển từ tầng trên xuống tầng dưới, kết quả trả về được chuyển từ tầng dưới lên tầng trên

hiệu năng hệ thống thông thường sẽ giảm khi số lượng tầng tăng lên

## Mô hình đối tượng phân tán



- ràng buộc lỏng hơn mô hình phân tầng
- mỗi đối tượng được coi là một thành phần và được kết nối thông qua cơ chế gọi thủ tục từ xa
- Các đối tượng trong mô hình này hoạt động tương đối độc lập, dễ dàng thay đổi và nâng cấp

- triển khai cho dịch vụ khách/chủ, ứng dụng yêu cầu thời gian thực
- Nhược điểm : Ko biết đối tượng bị gọi sẵn sàng không

## Mô hình khách/chủ

là hình thức trao đổi thông tin giữa các tiến trình cung cấp dịch vụ (Máy chủ) và tiến trình sử dụng dịch vụ (Máy khách)

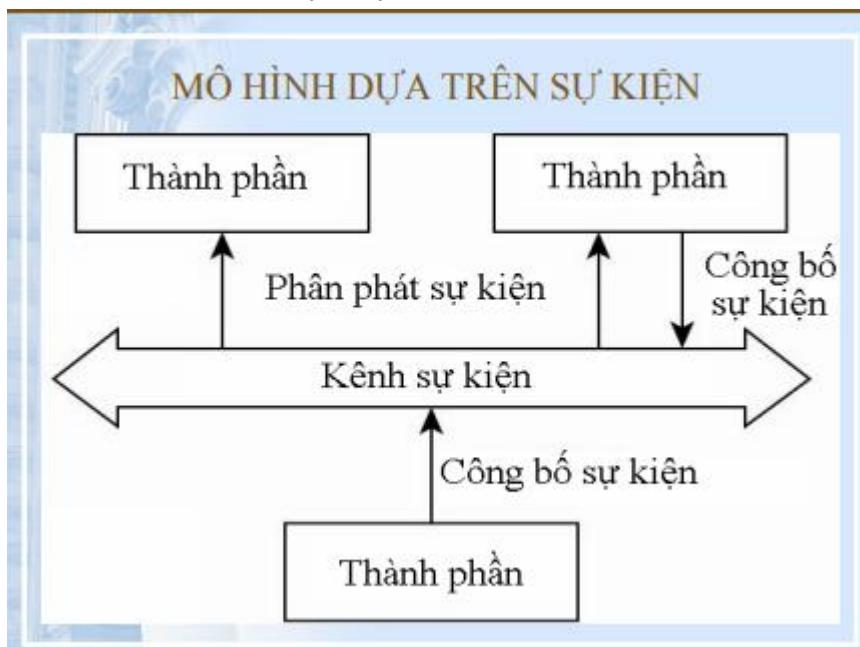
ưu điểm:

<Giao trình 18

Hoặc :

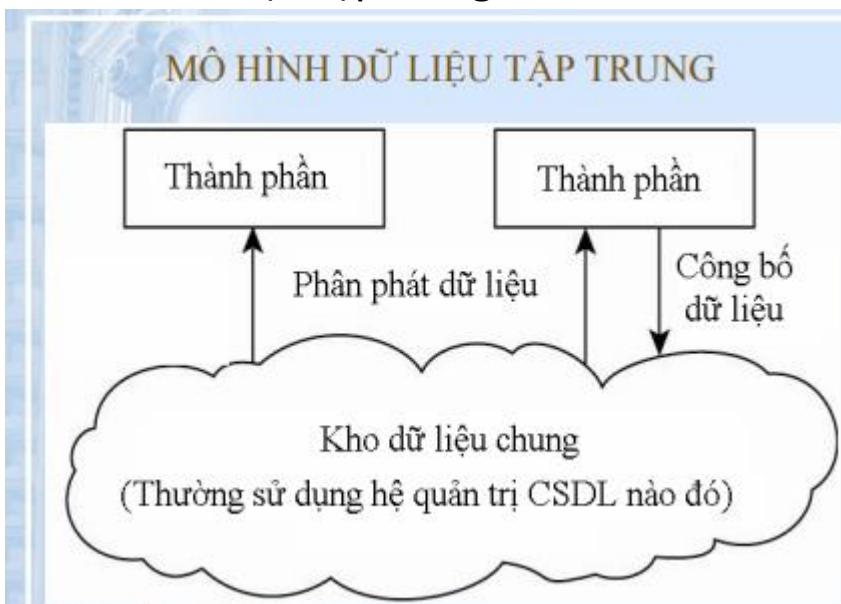
<http://www.hocvienmang.com/admin/pages/admin/elearn/ViewBookParts.aspx?VaazBrno=hLifbX59M0E%3D>

## Mô hình kênh sự kiện



- tiến trình trao liên lạc với nhau thông qua sự lan tỏa các sự kiện
- các sự kiện có thể mang theo dữ liệu và thường được gắn với các luật phân phát sự kiện.
- Các tiến trình phát tán sự kiện sau khi đã được phần mềm trung gian đảm bảo chỉ những tiến trình đã đăng ký mới nhận được sự kiện
- Mức độ ràng buộc giữa các tiến trình của mô hình này tương đối thấp

## Mô hình dữ liệu tập trung



Ba mô hình đã đề cập trên đòi hỏi **các tiến trình cung cấp dịch vụ** luôn phải ở **trạng thái hoạt động**,

hơn nữa nếu **lượng dữ liệu lớn** có thể dẫn tới hiện tượng quá thời gian.

**Mô hình dữ liệu tập :** **các tiến trình trao đổi thông tin** với nhau qua **kho dữ liệu chung** theo phương thức chủ động hoặc thụ động.

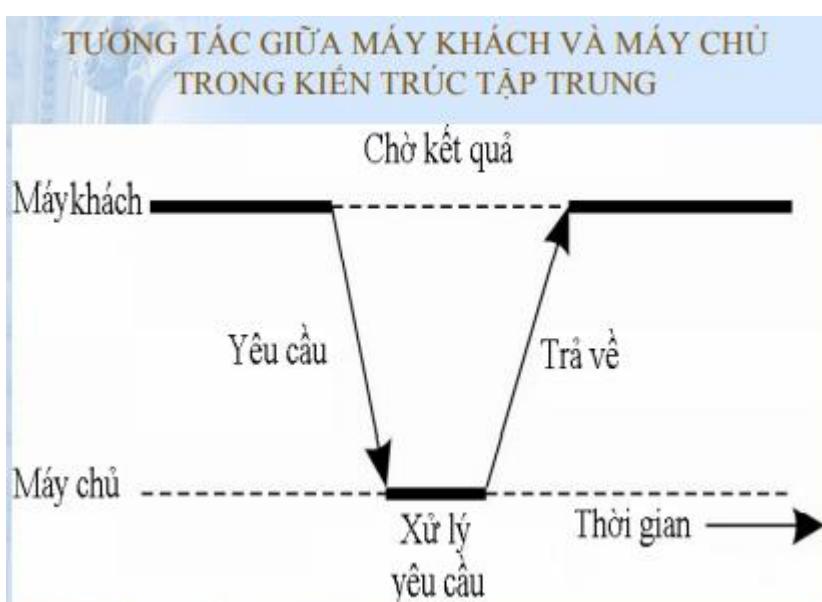
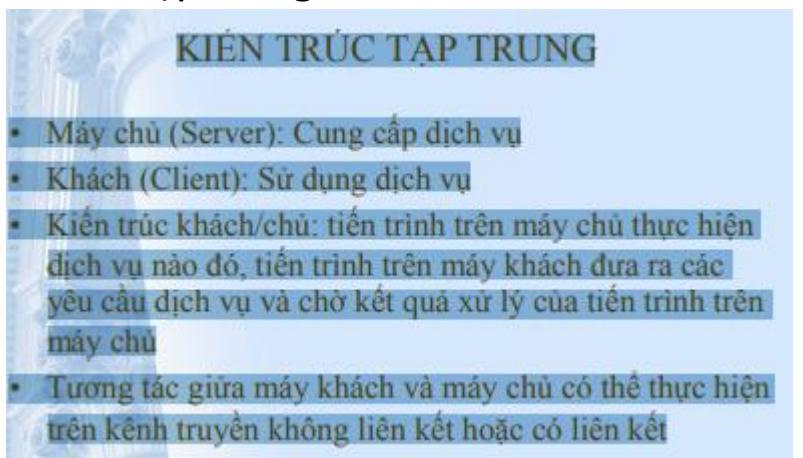
Mô hình này đảm bảo tính độc lập giữa các thành phần trong hệ thống và đồng thời

tiện lợi cho việc chia sẻ dữ liệu lớn.

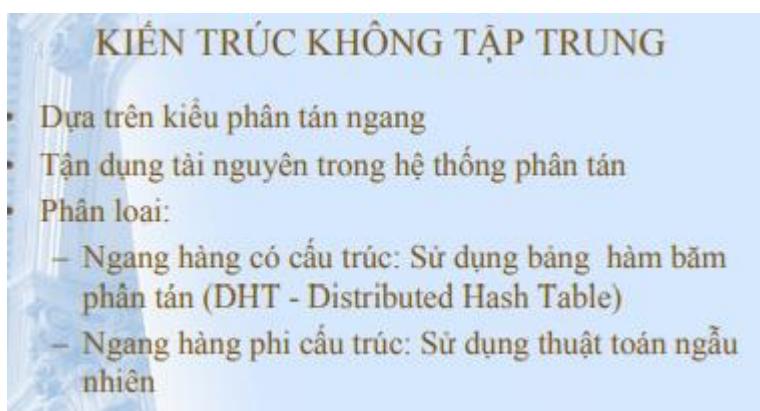
## Kiến trúc HT

- **kiến trúc phân tán đọc**  
các công việc xử lý được thực hiện bằng **cách đặt các máy tính lớn theo cấu trúc lớp**. Các **tiến trình xử lý** được phân cho các lớp thấp hơn tương ứng với cấu trúc tổ chức và loại nhiệm vụ.
- **phân tán ngang**
- Kiến trúc phân tán ngang bao **gồm nhiều máy tính** được kết nối **ngang hàng** vào **mạng** để **xử lý công việc**, có thể thêm máy tính nhằm nâng cao độ linh hoạt và nâng cấp hệ thống. Các công việc trước kia được tập trung trên một máy tính thì có **thể chia tách** **toán** với **các máy tính khác**. Có thể **sử dụng các thư viện** được cung cấp từ **các máy tính khác**, điều này đảm bảo được sự phân tán chức năng và sử dụng chung các nguồn tài nguyên.
- **lai ghép hai loại trên.**

## Kiến trúc tập trung



## Kiến trúc không tập chung



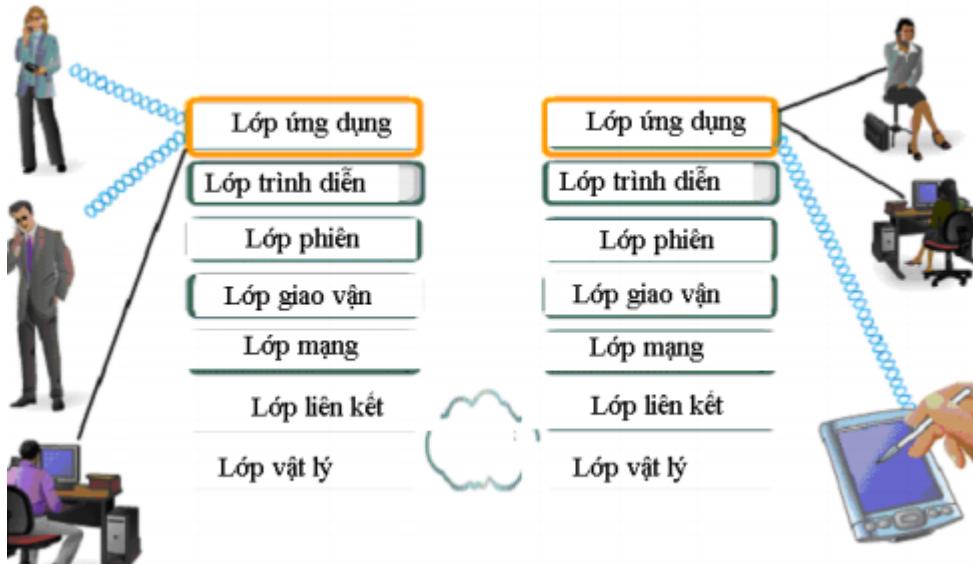
## Chương 2 : VĂN ĐỀ VÀ GIẢI PHÁP TRONG HỆ THỐNG PHÂN TÁN

### 2.1 Truyền thông

tiến trình trong hệ thống phân tán không sử dụng chung bộ nhớ, do đó việc trao đổi thông tin phải dựa hoàn toàn bằng phương thức truyền thông điệp

- tiến trình A muốn trao đổi thông tin với tiến trình B
- tạo một thông điệp trong vùng nhớ riêng của mình và thực hiện lời gọi hệ thống,
- khi đó hệ điều hành sẽ thực hiện chức năng chuyển thông điệp đó đến tiến trình B qua mạng.
- Về nguyên lý thì đơn giản như vậy, trong thực tế quá trình này khá phức tạp bởi trong hệ thống phân tán có thể có các máy tính thuộc các nhà sản xuất khác nhau và sử dụng tiêu chuẩn mã hóa thông tin khác nhau.
- Để khắc phục vấn đề này, tổ chức chuẩn hóa Quốc tế ISO đã đưa ra mô hình liên kết hệ thống mở (OSI). Mặc dù các giao thức trong mô hình OSI ít được sử dụng, tuy nhiên đó là mô hình khá tốt để hiểu về mạng máy tính. Mô hình OSI được phân thành 7 tầng, mỗi tầng bao gồm các giao thức qui định khuôn dạng dữ liệu và các thủ tục xử lý như cách gửi/nhận đơn vị dữ liệu, cách xử lý khi xảy ra lỗi. Xét trên khía cạnh cách thiết lập liên kết giữa các thực thể truyền thông người ta phân ra hai loại giao thức:
  - Giao thức có liên kết: Cần phải thiết lập liên kết trước khi truyền dữ liệu, sau khi truyền xong sẽ hủy bỏ liên kết.
  - Giao thức không liên kết: Không cần phải thiết lập liên kết khi truyền dữ liệu

Giao thức mạng : mô hình OSI:



- Tầng ứng dụng: Cung cấp giao diện phục vụ cho người sử dụng và các ứng dụng khác
- Tầng trình diễn: Thực hiện mã hóa/giải mã, nén/giải nén và bảo mật dữ liệu.
- Tầng phiên: Tạo ra các phiên làm việc.
- Tầng giao vận: Tạo liên kết giữa đầu cuối với đầu cuối, điều khiển tốc độ truyền dữ liệu, xử lý lỗi truyền tin.
- Tầng mạng: Quản lý địa chỉ logic của các đối tượng tham gia vào mạng, tìm đường đi tốt nhất cho mỗi gói tin.(GT IP)
- Tầng liên kết: Thiết lập liên kết giữa hai thiết bị vật lý kề cạnh nhau
- Tầng vật lý: Biến đổi các bit dữ liệu thành các tín hiệu phù hợp với môi trường truyền dẫn và thực hiện thu phát các tín hiệu đó.

## Các giao thức mức thấp

chỉ các giao thức thuộc ba lớp thấp nhất của mô hình OSI: tầng vật lý, tầng liên kết dữ liệu và tầng mạng.

## Phân loại truyền thông

- Gọi thủ tục từ xa
- Truyền thông luồng thông điệp
- Truyền thông luồng luồng
- Truyền thông nhóm

Gọi thủ tục từ xa

Phương pháp gọi thủ tục từ xa cho phép cài đặt **các hệ thống phân tán theo mô hình khách/chủ**

là phương pháp được áp dụng phổ biến nhất

gọi thủ tục trên máy chủ được thực hiện tương tự như gọi thủ tục trên máy cục bộ  
máy khách chuyển các tham số đầu vào khi gọi thủ tục  
dịch vụ trên máy chủ sẽ kiểm tra tính hợp lệ của các tham số đó  
thực hiện tính toán và trả về các giá trị theo yêu cầu của ứng dụng máy khách  
`count = read(fd, buff, nbytes);`  
trong đó `fd` là con trỏ tập tin, `buff` là vùng đệm, `nbytes` là số lượng byte cần đọc.

Khi chương trình gọi thủ tục, nó tạo ra một ngăn xếp dành cho thủ tục đó

thực hiện xong thủ tục, giá trị trả về sẽ được chuyển tới các thanh ghi, giải phóng vùng nhớ và chuyển quyền điều khiển cho chương trình gọi

Việc truyền tham số có thể được thực hiện bằng một trong ba phương pháp: **Truyền giá trị, truyền con trỏ và truyền tham chiếu**.

**Mục tiêu** : che giấu quá trình thực hiện thủ tục trên máy tính khác, điều này được thực hiện bằng cách che giấu quá trình trao đổi thông tin giữa các máy tính  
thành phần trên máy khách gọi là **Stub** sẽ bao bọc các tham số vào thông điệp

Tại máy chủ : Skeleton sẽ giải mã thông điệp đã nhận được và thực thi các mã lệnh trả về Skeleton sẽ giải mã thông điệp đã nhận được và thực thi các mã lệnh

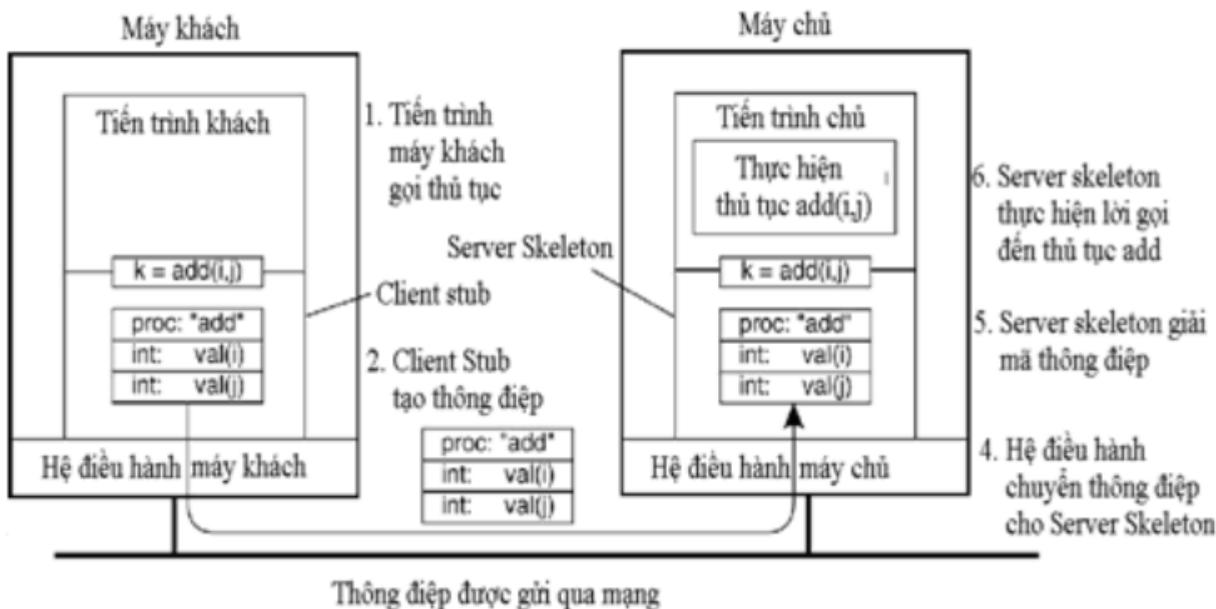
- **Gọi thủ tục từ xa và mô hình OSI:**
  - Người dùng không mở và đóng liên kết, thậm chí không biết đang sử dụng mạng.
  - Gọi thủ tục từ xa có thể bỏ qua các lớp giao thức nhằm nâng cao hiệu quả
  - Gọi thủ tục từ xa rất phù hợp cho các tương tác khách chủ luân phiên luồng điều khiển.

Quá trình thực hiện gọi thủ tục từ xa được thực hiện qua mười bước sau:

1. Thủ tục trên máy khách gọi stub như phương pháp gọi thủ tục truyền thống.
2. Stub tạo thông điệp và chuyển đến hệ điều hành của máy khách .
3. Hệ điều hành của máy khách gửi thông điệp đến hệ điều hành của máy chủ.
4. Hệ điều hành của máy chủ chuyển thông điệp đến Skeleton .
5. Skeleton giải mã thông điệp thành các tham số và gọi thủ tục xử lý tương ứng.
6. Máy chủ thực hiện lời gọi thủ tục và trả về giá trị cho Skeleton.
7. Skeleton đóng gói tham số giá trị trả về thành thông điệp và chuyển đến hệ điều hành của máy chủ.
8. Hệ điều hành của máy chủ chuyển thông điệp đến hệ điều hành của máy khách.

9. Hệ điều hành của máy khách nhận thông điệp và chuyển cho Stub.

10. Stub giải mã kết quả và trả về các tham số theo yêu cầu của thủ tục đã gọi.



gọi thủ tục từ xa, không thể áp dụng phương pháp truyền con trỏ do đó chỉ có thể thực hiện bằng cách truyền giá trị hoặc truyền tham chiếu

*Fương pháp gọi thủ tục từ xa*

phương pháp đồng bộ hoặc không đồng bộ

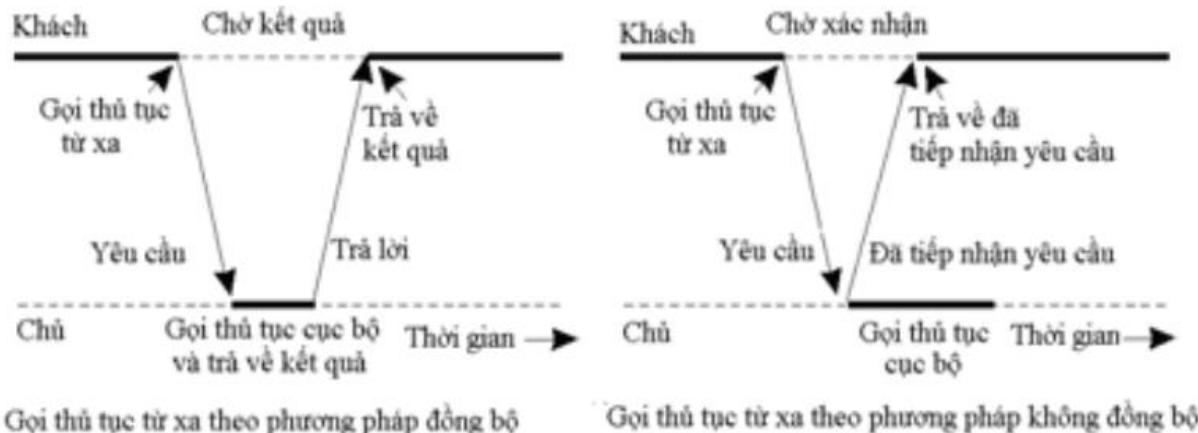
**f** *phương pháp gọi đồng bộ:*

- Tiến trình trên máy khách sẽ bị phong tỏa cho đến khi nhận được kết quả trả về, việc chờ đợi này đồng nghĩa với sự suy giảm hiệu năng của hệ thống.

**f** *Phương pháp gọi không đồng bộ:*

Cho phép tiến trình gọi trên máy khách gửi yêu cầu đến máy chủ, sau khi máy chủ xác nhận đã nhận được yêu cầu, tiến trình trên máy khách có thể tiếp tục xử lý các tác vụ khác mà không cần chờ đợi kết quả xử lý của máy chủ

sau khi hoàn thành, tiến trình trên máy chủ sẽ gọi tiến trình trên máy khách để trao kết quả trả về



## Truyền thông hướn thong diep

**Gọi tt từ xa :** đòi hỏi tiến trình cung cấp dịch vụ phải luôn ở trạng thái sẵn sàng tiếp nhận yêu cầu

Các phuong phap  
cung cấp giao diện trao đổi thông điệp

- **Truyền nhất thời:** Thông điệp chỉ được lưu trong thời gian thực hiện chuyển thông tin, nếu không thành công thì thông điệp đó sẽ bị hủy bỏ.
- **Truyền bền bỉ:** Thông điệp được lưu tại bộ đệm cho đến khi toàn bộ thông điệp đã gửi thành công cho bên nhận. Chuyển thông điệp bền bỉ tương tự như tổ chức mạng lưới bưu chính, bưu phẩm được chuyển đến các bưu cục và người đưa thư có nhiệm vụ chuyển các bưu phẩm đó đến người nhận, quá trình này có thể phải được chuyển qua một hoặc nhiều bưu cục.

## Truyền thông hướng luồng

Truyền thông bằng cách gọi thủ tục từ xa hay truyền thông điệp mới chỉ chú trọng đến việc chuyển dữ liệu mà chưa đề cập đến loại dữ liệu cần chuyển, thời gian thực hiện, tính liên tục và yêu cầu đồng bộ. **Truyền tin yêu cầu thời gian thực phải được thực hiện với độ trễ thấp nhất** có thể nhưng vẫn phải đáp ứng các tiêu chuẩn về chất lượng dịch vụ. Các dịch vụ đa phương tiện bao gồm nhiều luồng dữ liệu khác nhau như văn bản, âm thanh, hình ảnh..., chúng đòi hỏi tính đồng bộ giữa các luồng dữ liệu và đồng thời sự liên tục của thông tin cũng là thước đo của chất lượng dịch vụ.

## Truyền thông theo nhóm

Truyền thông theo nhóm là **phương pháp truyền dữ liệu từ một thành viên đến nhiều thành viên khác trong cùng một nhóm chỉ với một lần thực hiện**. Truyền thông theo nhóm được sử

dụng trong rất nhiều lĩnh vực của ứng dụng phân tán như: dịch vụ phát thanh, truyền hình, hội thảo....

## Chương 3 :Đặt tên , định danh ,tìm kiếm

Tên là một xâu các bit hoặc các ký tự dùng để tham chiếu đến một thực thể

Định danh là một loại tên đặc biệt, mỗi thực thể chỉ được tham chiếu bởi duy nhất một định danh và mỗi định danh tham chiếu duy nhất đến một thực thể

- Định danh chỉ tham chiếu đến duy nhất một thực thể
- Mỗi thực thể chỉ có một định danh
- Không được tái sử dụng định danh.

Địa chỉ là điểm truy nhập đến thực thể, các điểm truy nhập này cũng phải được đặt tên và tên đó chính là địa chỉ của thực thể

- Một tên có thể gồm nhiều địa chỉ
- Thực thể có thể thay đổi địa chỉ trong quá trình tồn tại
- Một địa chỉ có thể trỏ đến các thực thể khác nhau trong các thời điểm khác nhau
- Đảm bảo có thể tham chiếu tới các tài nguyên bằng tên

### Các phương pháp Đặt tên và các giải pháp tìm kiếm

#### Đặt tên phi cấu trúc

còn gọi là tên phẳng đơn thuần chỉ gồm chuỗi các bit ngẫu nhiên không chứa bất kỳ thông tin nào liên quan tới thực thể

Các giải pháp tìm kiếm đơn giản

#### GIAI PHAP ĐƠN GIAN

Gửi quảng bá: ARP, chỉ phù hợp cho các mạng nhỏ

Gửi theo nhóm: dùng địa chỉ Multicast

Dùng con trỏ chuyển tiếp: Khi một thực thể di chuyển tới vị trí mới, nó để lại thông tin tham chiếu đến vị trí mới. Việc tìm kiếm phải đi qua chuỗi mắt xích địa chỉ.

Cách tìm kiếm dựa trên nguồn gốc

## CÁCH TIEP CẬN DỰA TREN NGUON GOC (Home based)

- Quảng bá hoặc multicast không phù hợp với các mạng lớn
- Chuyển tiếp con trỏ làm tăng độ trễ, đôi khi nảy sinh trường hợp con trỏ chuyển tiếp không tới được đích
- Cách tiếp cận dựa trên nguồn gốc:
  - Cho phép máy nơi đối tượng được sinh ra luôn lưu giữ tham chiếu đến vị trí hiện hành của đối tượng đó (vị trí này gọi là nguồn gốc của đối tượng).
  - Tham chiếu được lưu trữ và duy trì theo cách chịu lỗi (fault tolerance)
  - Khi một mắt xích bị lỗi, nó sẽ hỏi nguồn gốc: đối tượng hiện nay đang ở đâu.

Bảng băm phân tán

## BÀNG BĂM PHÂN TÁN (DHT)

Cách tiếp cận dựa trên nguồn gốc:

- Dựa trên vị trí cố định của nguồn gốc
- Thực tế không đảm bảo luôn hoạt động, khắc phục bằng cách đăng ký dịch vụ đặt tên
- Chưa giải quyết triệt để vấn đề trễ thông tin

Sử dụng bảng băm phân tán dùng thuật toán Chord:

- Sử dụng không gian m-bit (128 đến 160) để gán ngẫu nhiên các định danh cho các nút cũng như các thực thể (tập tin, tiến trình...)
- Một nút K trỏ tới nút tiếp theo là nút có định danh lớn hơn gọi là Successor(K), một nút có định danh nhỏ hơn gọi là Predecessor(K). Các nút liên kết với nhau dựa vào Successor và Predecessor của nó.
- Mỗi nút sẽ lưu một bảng định tuyến gọi là Finger Table. Thay vì phải tìm kiếm tuyến tính, bảng định tuyến cho phép một nút định tuyến tới các nút ở xa. Mỗi dòng trong bảng Finger Table sẽ lưu thông tin về 1 nút ở xa, gọi là một chỉ mục.

Tìm kiếm dựa trên phân cấp

## CÁCH TIẾP CẬN PHÂN CẤP

Vấn đề sử dụng bảng băm phân tán: các yêu cầu được gửi đi thất thường

Cách tiếp cận phân cấp:

- Dựa trên dịch vụ định vị toàn cầu
- Mạng được chia thành tập các vùng miền
- Tồn tại một vùng cao nhất bao trùm toàn mạng
- Vùng thấp nhất gọi là lá

Đặt tên có cấu trúc :

phù hợp các máy tính nhưng nó hoàn toàn không thân thiện với con người

Không gian tên

Các tên thường được tổ chức thành không gian tên

thể hiện bằng các nhãn hoặc đồ thị có hướng với hai loại nút: nút lá và nút thư mục

## Chương 4 : Đồng bộ

Trao đổi thông tin giữa các tiến trình là một trong những nhiệm vụ chủ chốt trong các hệ thống phân tán

vấn đề sử dụng chung tài nguyên vật lý, các tiến trình không thể đồng thời truy nhập mà mỗi tiến trình sẽ được cấp phép chiếm giữ tạm thời trong một khoảng thời gian nhất định

đòi hỏi phải có các thuật toán phân phối quyền sử dụng tài nguyên trong hệ thống

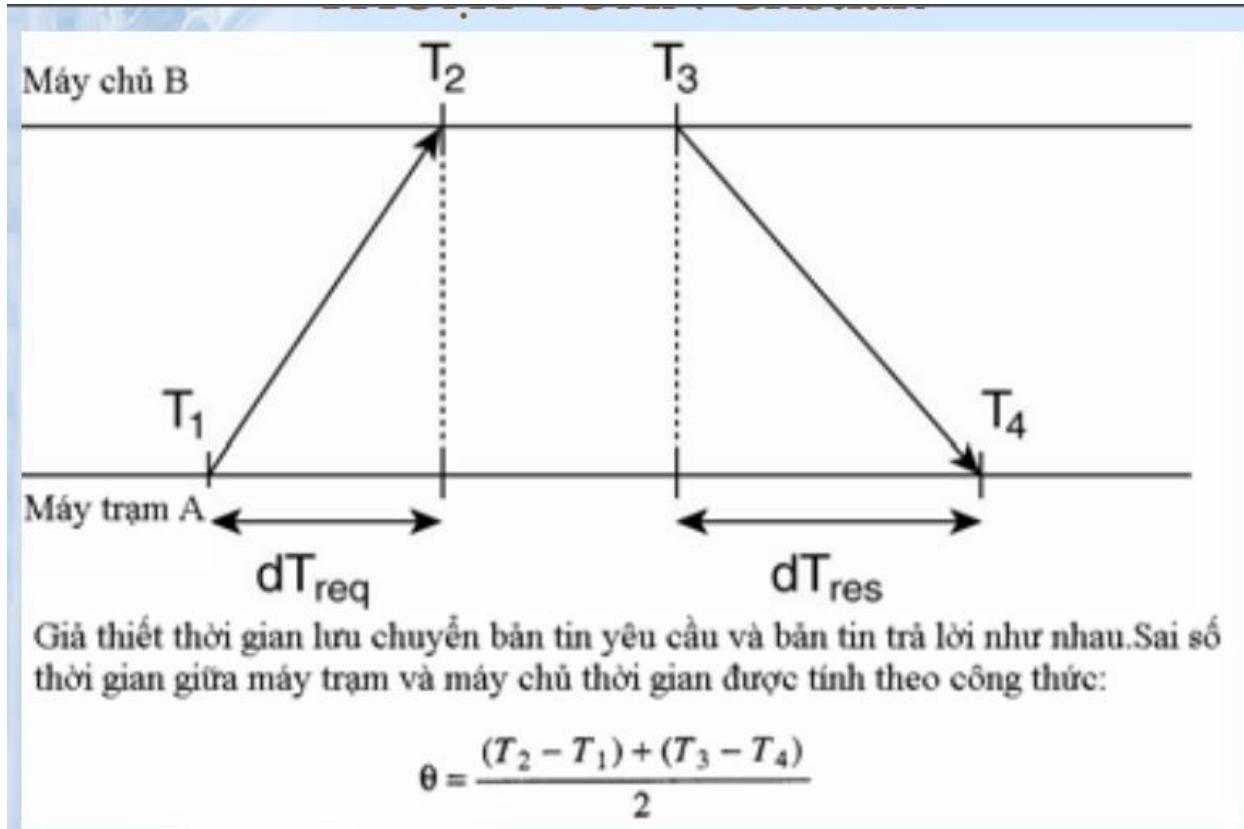
Mỗi thành viên trong hệ thống phân tán sử dụng đồng hồ vật lý riêng

không có gì đảm bảo rằng tại một thời điểm đồng hồ vật lý của tất cả các thành viên trong hệ thống đều cho giá trị giống nhau

Giải thuật Cristiana

Giao thức thời gian mạng NTP (Network Time Protocol) dùng để đồng bộ đồng hồ của các hệ thống máy tính thông qua mạng dữ liệu chuyển mạch gói với độ trễ thay đổi

Mỗi máy được cài đặt bộ phận nhận thời gian từ máy chủ trên mạng, nó giải quyết vấn đề trễ thông tin lưu chuyển trên mạng bằng cách tính toán tương đối độ lệch thời gian theo giải thuật Cristian và độ chính xác có thể đạt <50 ms.



Giả thiết thời gian lưu chuyển bản tin yêu cầu và bản tin trả lời như nhau. Sai số thời gian giữa máy trạm và máy chủ thời gian được tính theo công thức:

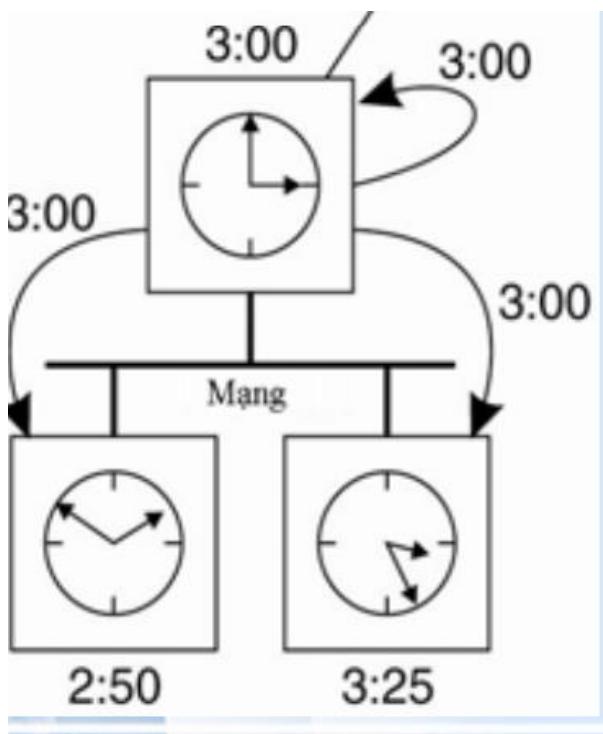
$$\theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

Nếu giả thiết thời gian lưu chuyển thông tin yêu cầu gửi đi và kết quả trả về bằng nhau, thì máy khách có thể cập nhật thời gian đồng bộ là thời gian trả về của máy chủ cộng thêm  $(T_4 - T_1)/2$ .

### Giải thuật Berkeley

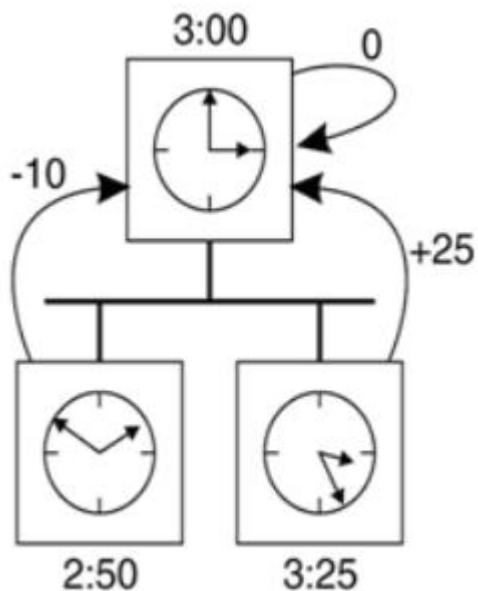
- Giải thuật Cristiana đòi hỏi phải có một máy chủ cung cấp thời gian cho các thành viên trong hệ thống, như vậy thời gian của mỗi thành viên sẽ hoàn toàn phụ thuộc vào máy chủ này.
- Giải thuật Berkeley thực hiện tính toán thời bằng cách tham khảo thời gian của tất cả các thành viên trong hệ thống
- máy chủ thời gian chỉ đóng vai trò như một thành viên điều phối
- thích hợp cho hệ thống không có khả năng đồng bộ với các máy chủ thời gian quốc tế
- Máy chủ yêu cầu máy các máy trạm cung cấp thời gian của mỗi máy,
- máy chủ tính toán thời gian trung bình và lấy đó làm thời gian mới của hệ thống
- sau đó tính toán độ lệch thời gian trung bình với thời gian thực của mỗi máy trạm và gửi yêu cầu tới các máy trạm cập nhật lại thời gian hệ thống.

Vd



- Server sẽ gửi đồng hồ của mình cho các Client
- T2 và T3 tính toán độ lệch thời gian và gửi về cho Server lần lượt là:

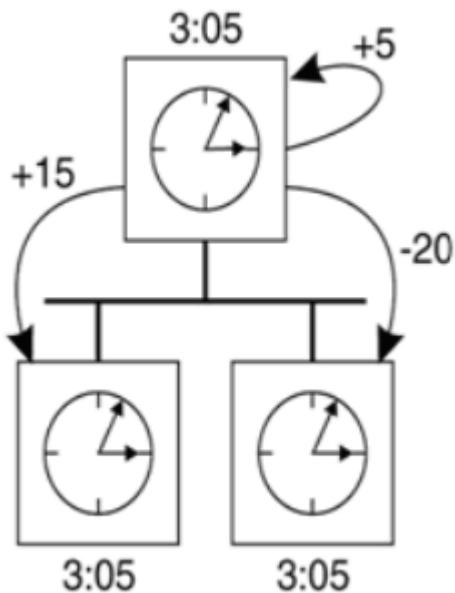
$$T1' = -10' \text{ và } T2' = +25'$$



- Sau đó Server sẽ đặt thời gian của mình là:

$$T_{\text{new}} = 3h + 5'$$

Và báo cho các máy Client T2 (+15') và T3 (-20')



### Giải thuật trung bình

Theo chu kỳ, tất cả các máy gửi quảng bá thời gian của máy đó. Sau khi gửi, nó bắt đầu nhận các tin quảng bá thời gian từ các máy tính khác. Loại bỏ các tin quảng bá có giá trị biên, lấy thời gian trung bình của các tin quảng bá thời gian còn lại để làm thời gian mới của máy tính.

### Đồng bộ thời gian trong mảng không dây

#### ĐỒNG BỘ THỜI GIAN TRONG MẠNG KHÔNG DÂY

- Độ trễ lưu chuyển thông tin trong mạng không dây phụ thuộc nhiều yếu tố:
  - Thông tin phải di chuyển qua bộ định tuyến không dây
  - Tốc độ trong mạng không dây không ổn định, phụ thuộc nguồn năng lượng, khoảng cách...
  - Không lường trước được thời gian chờ để được thiết bị định tuyến không dây phục vụ
- Áp dụng phương pháp đồng bộ tham chiếu quảng bá, sử dụng thuật toán trung bình

## Thời gian logic và các đồng hồ logic

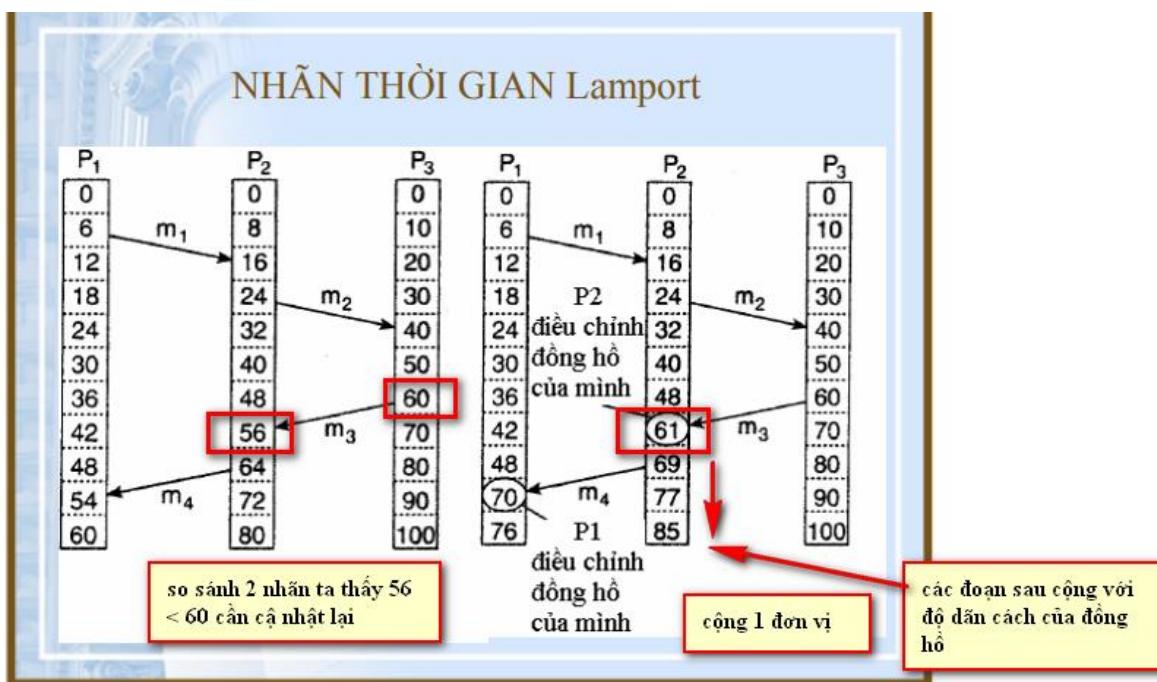
tiến trình trao đổi thông tin với nhau không nhất thiết phải sử dụng thời gian vật lý mà chỉ cần xác định sự kiện nào xảy ra trước

do đó người ta đưa ra khái niệm đồng hồ logic

### Đồng hồ logic Lamport

Sự kiện a xảy ra trước sự kiện b, ký hiệu là  $a \rightarrow b$ , được gọi là đúng nếu:

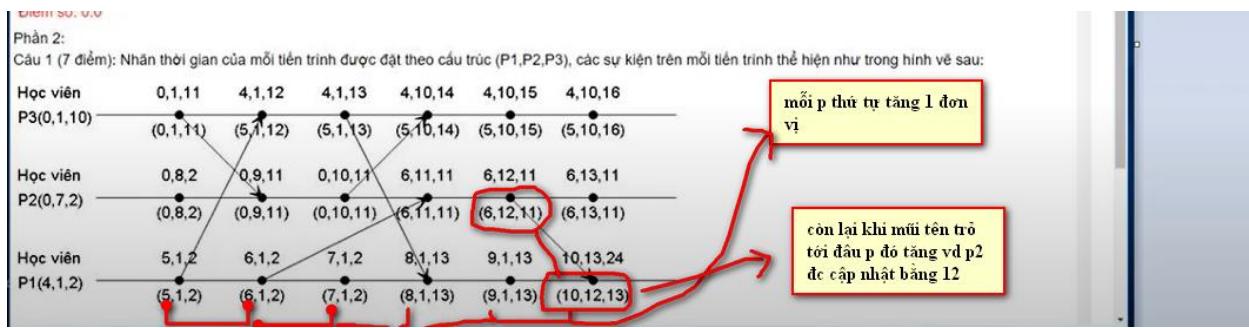
- Nếu a và b là hai sự kiện xảy ra trong cùng một tiến trình và a xảy ra trước b.
- Nếu a và b không thuộc cùng một tiến trình nhưng a là sự kiện gửi một thông điệp đi và b là sự kiện nhận thông điệp đó.
- Nếu a xảy ra trước b thì  $a \rightarrow b$  là đúng.
- Nếu a là sự kiện bản tin được gửi bởi một tiến trình nào đó và b là sự kiện bản tin đó được nhận bởi một tiến trình khác thì quan hệ  $a \rightarrow b$  là đúng.
- Quan hệ xảy ra trước có tính bắc cầu:  $a \rightarrow b, b \rightarrow c$  thì  $a \rightarrow c$ .



khi chưa áp dụng phương pháp điều chỉnh nhãn thời gian Lamport, chênh lệch giữa tiến trình P<sub>3</sub> và tiến trình P<sub>1</sub> là 40 nhịp nhưng nhờ có sự điều chỉnh nhãn thời gian Lamport, giá trị này giảm xuống chỉ còn 24.

### Đồng hồ vector

- vector VT(a) được gán cho một sự kiện a
- sự kiện a trước sự kiện b thì ta có VT(a) < VT(b)
- Nhãn thời gian Lamport không đề cập tới nguyên nhân
- Hai sự kiện a và b, nếu VT(a) < VT(b) thì sự kiện a là nguyên nhân của b



Các trạng thái toàn cục

trạng thái toàn cục của một hệ thống phân tán được đặc trưng bởi trạng thái của từng tiến trình và các thông điệp đang lưu chuyển trong hệ thống

Một trong những phương pháp để xác định trạng thái toàn cục là sử dụng lát cắt, lát cắt mô tả sự kiện cuối cùng mà sự kiện này được ghi lại cho mỗi tiến trình

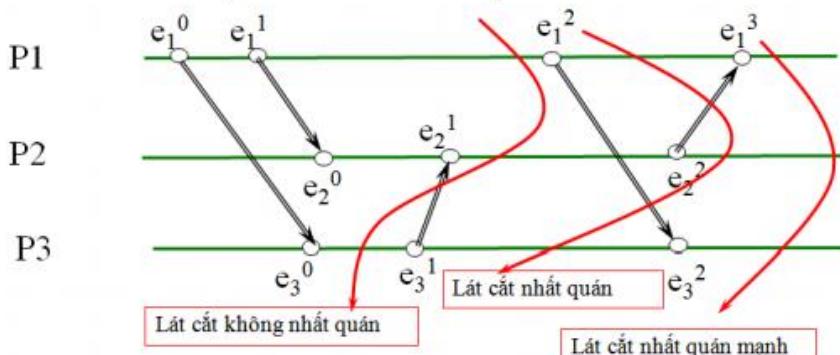
bằng cách này nó có thể kiểm tra lại rằng tất cả các thông điệp nhận đều tương ứng với các thông điệp gửi được ghi lại trên đường cắt.

Lát cắt mô tả sự kiện cuối cùng mà sự kiện này được ghi lại cho mỗi tiến trình

Lịch sử tiến trình  $(P_i)$  =  $h_i = \langle e_i^0, e_i^1, \dots \rangle$

Tiền sử tiến trình  $(P_i)$  =  $h_i^k = \langle e_i^0, e_i^1, \dots, e_i^k \rangle$

Trạng thái tiến trình  $S_i$ : là trạng thái của  $P_i$  ngay trước khi sự kiện thứ k



Hình 2.27 Phân loại lát cắt

Lát cắt mô tả sự kiện cuối cùng mà sự kiện này được ghi lại cho mỗi tiến trình.

Cho tập các tiến trình  $P_1, \dots, P_i, \dots$ :

### Các giải thuật loại trừ tương hỗ phân tán

vấn đề cơ bản của các hệ thống phân tán là sự tương tranh sự cộng tác giữa các tiến trình

cùng một lúc nhiều tiến trình cùng truy nhập đến một tài nguyên dẫn đến sự xung đột giải thuật loại trừ tương hỗ đã được đề xuất dựa trên phương pháp sử dụng thẻ bài và cấp quyền truy nhập.

Giải thuật tập trung

- mỗi tiến trình có một **số định danh duy nhất**
- tiến trình được bầu chọn làm điều phối là **tiến trình có số hiệu định danh cao nhất.**
- một tiến trình nào đó cần vào vùng giới hạn để truy nhập tài nguyên nó sẽ **gửi một thông điệp xin cấp quyền**
- 
- **Không có tiến trình** nào đang trong vùng giới hạn thì **tiến trình điều phối sẽ gửi phản hồi cấp phép**
- ngược lại sẽ **gửi thông điệp từ chối** và **chuyển yêu cầu này vào hàng đợi**
- tiến trình **một tiến trình rời khỏi vùng giới hạn** nó sẽ **gửi một thông điệp đến tiến trình điều phối** thông báo trả lại quyền truy nhập
- tiến trình điều phối **sẽ gửi quyền truy nhập** cho tiến trình đầu tiên trong hàng đợi truy nhập.
- Giải thuật này đảm bảo sẽ **chỉ có tối đa một tiến trình** trong vùng giới hạn và chỉ cần 3 thông điệp để thiết lập là: **Yêu cầu, cấp phép và giải phóng**

**Nhược điểm :**

- nếu tiến trình điều phối bị hỏng thì sẽ không có một tiến trình nào được cấp phép
- khi một tiến trình đang trong vùng giới hạn mà bị phong tỏa thì tài nguyên cũng không được giải phóng
- hiện tượng thắt cổ chai

### **Giải thuật không tập trung**

- Nếu gt tập chung : chỉ 1 tT điều phối , Tài nguyên nghèo nàn
- Khắc phục : **Nhân bản N tài nguyên**  
**mỗi bản sao sẽ có tiến trình điều phối riêng để kiểm soát truy nhập**
- ít lỗi hơn Gt tập trung
- Một tiến trình muốn truy nhập tài nguyên thì **phải gửi yêu cầu đến N tiến trình điều phối đang hoạt động** và sẽ **được cấp quyền truy nhập** khi và chỉ khi nhận **được số phiếu đồng ý lớn hơn N/2**  
chỉ khi nhận được số phiếu đồng ý lớn hơn N/2. So với giải thuật tập trung, giải thuật này ít bị lỗi hơn, gọi p là xác suất bị lỗi của mỗi tiến trình điều phối, xác suất k trong m tiến trình điều phối bị lỗi sẽ là:

$$P[k] = \binom{m}{k} p^k (1-p)^{m-k}$$

### **Giải thuật phân tán**

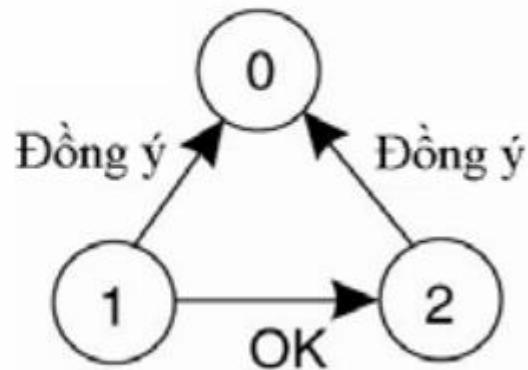
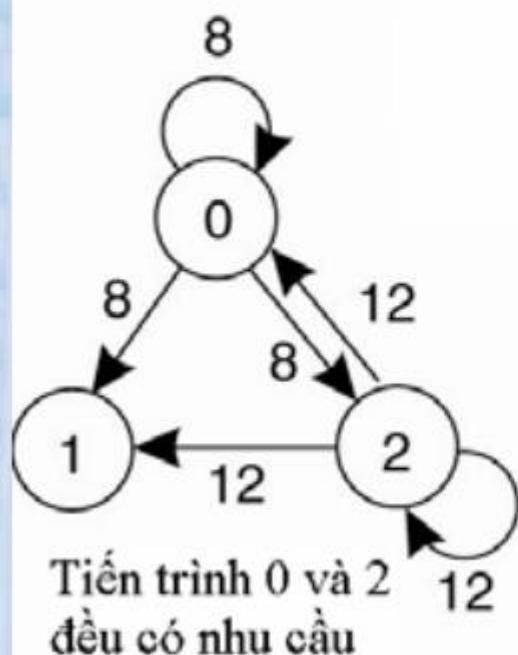
một tiến trình muốn vào vùng giới hạn

tạo ra một nhãn thời gian và gắn vào thông điệp gửi đến tất cả các tiến trình khác  
tiến trình khác sau khi nhận được thông điệp này sẽ xảy ra ba tình huống:

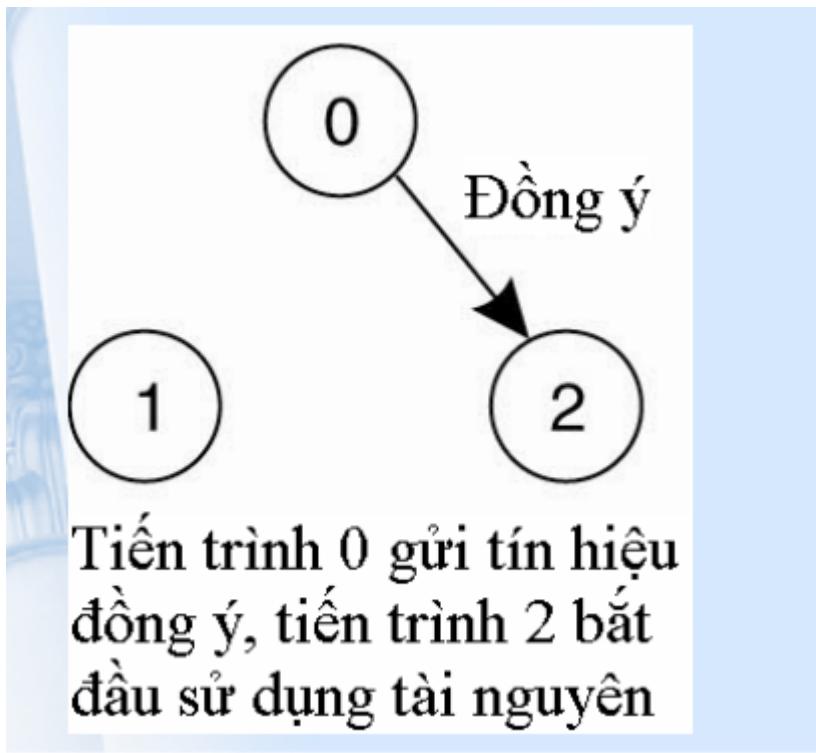
- **Nếu không ở trong vùng giới hạn** và **cũng không có nhu cầu vào vùng giới hạn** thì nó sẽ **gửi thông điệp chấp nhận**.

- Nếu đang ở trong vùng giới hạn thay vì trả lời nó sẽ đưa vào hàng đợi, chờ đến khi thoát khỏi vùng giới hạn sẽ phản hồi thông điệp chấp nhận và xóa yêu cầu đó trong hàng đợi.
- Nếu tiến trình cũng có nhu cầu vào vùng giới hạn thì nó sẽ so sánh nhãn thời gian của mình với nhãn thời gian gắn trong thông điệp nhận được. Nếu nhãn thời gian của tiến trình lớn hơn nhãn thời gian trong gán trong thông điệp thì phải phản hồi bằng thông điệp chấp nhận, ngược lại nếu nhãn thời gian của tiến trình **nhỏ hơn** nhãn thời gian gắn trong thông điệp nhận được thì nó chuyển yêu cầu vào hàng đợi, chờ đến khi không có nhu cầu vào vùng giới hạn thì sẽ lấy thông điệp từ hàng đợi, phản hồi đồng ý cho các tiến trình tương ứng và đồng thời xóa các thông điệp này trong hàng đợi.

## THUẬT TOÁN PHÂN TÁN



⋮



#### Giải thuật thẻ bài

- ✓ tất cả các tiến trình được sắp xếp trên một vòng tròn logic
- ✓ các tiến trình đều được đánh số và đều biết đến các tiến trình cạnh nó
- ✓ hệ thống bắt đầu khởi tạo
- ✓ tiến trình 0 sẽ được trao thẻ bài
- ✓ luân chuyển quanh vòng tròn logic
- ✓ Khi một tiến trình nhận được thẻ bài từ tiến trình liền trước
- ✓ kiểm tra xem có cần thiết vào vùng giới hạn hay không
- ✓ không cần thiết thì chuyển thẻ bài cho tiến trình kế tiếp
- ✓ ngược lại sẽ vào vùng giới hạn
- ✓ hoàn thành phần việc của mình nó sẽ trả thẻ bài cho tiến trình kế tiếp
- ✓ **Vấn đề :** Mất thẻ bài : khởi tạo lại từ đầu

#### So sánh các giải thuật loại trừ

dựa trên số lượng thông điệp cần lưu chuyền, độ trễ và những vấn đề tiềm ẩn trong giải thuật.

Giải thuật	Số lượng thông điệp	Độ trễ (SL thông điệp)	Nhược điểm
Tập trung	3	2	Tiến trình điều phối lỗi
Phi tập trung	$3mk$ , $k=1,2..$	$2m$	Kém hiệu quả
Phân tán	$2(n - 1)$	$2(n - 1)$	Lỗi của bất kỳ tiến trình nào
Thẻ bài	1 đến $\infty$	0 đến $n - 1$	Mất thẻ bài

Giải thuật tập trung đơn giản và hiệu quả nhất, nó chỉ cần ba thông điệp để vào, ra và giải phóng khỏi vùng tối hạn. Nếu mỗi tài nguyên được nhân bản m lần thì giải thuật phi tập trung cần tới  $3mk$  thông điệp, trong đó  $k=1,2,3....$  tương ứng với số lần gửi yêu cầu. Nếu hệ thống gồm  $n$  thành viên thì giải thuật phân tán cần  $2(n-1)$  thông điệp, số lượng thông điệp cần gửi trong giải thuật thẻ bài luôn thay đổi.

## Các giải thuật bầu chọn

Chọn tiến trình điều phối

### Giải thuật nổi bợt,nổi trội

- ✓ mỗi tiến trình đều có một định danh duy nhất
- ✓ tất cả các tiến trình khác đều có thể biết được định danh và địa chỉ của mỗi tiến trình trong hệ thống
- ✓ chọn một tiến trình có định danh cao nhất

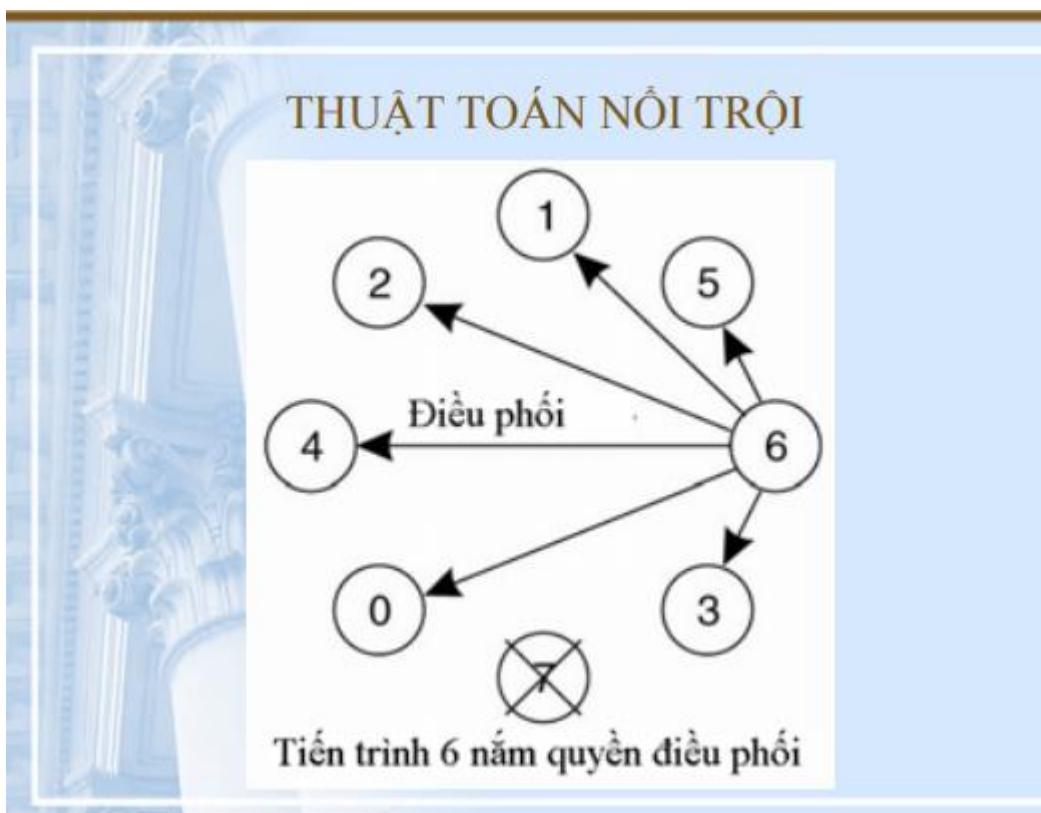
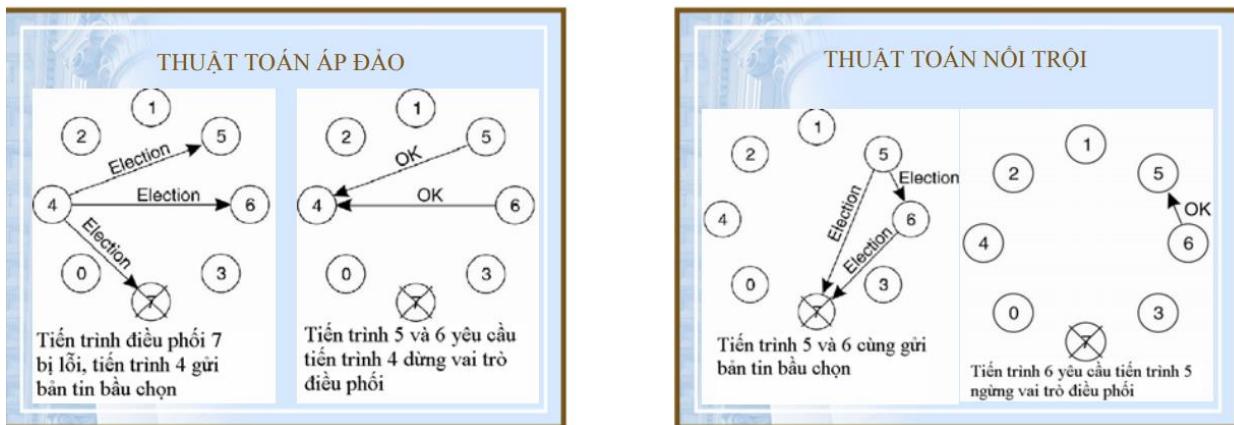
## THUẬT TOÁN ÁP ĐẢO

Mỗi một tiến trình đều có một ID duy nhất.Tất cả các tiến trình khác đều có thể biết được số ID và địa chỉ của mỗi tiến trình trong hệ thống.

Chọn một tiến trình có ID cao nhất làm khóa.Tiến trình sẽ khởi động việc bầu chọn nếu như nó khôi phục lại sau quá trình xảy ra lỗi hoặc tiến trình điều phối bị trực tiếp.

Các bước của giải thuật:

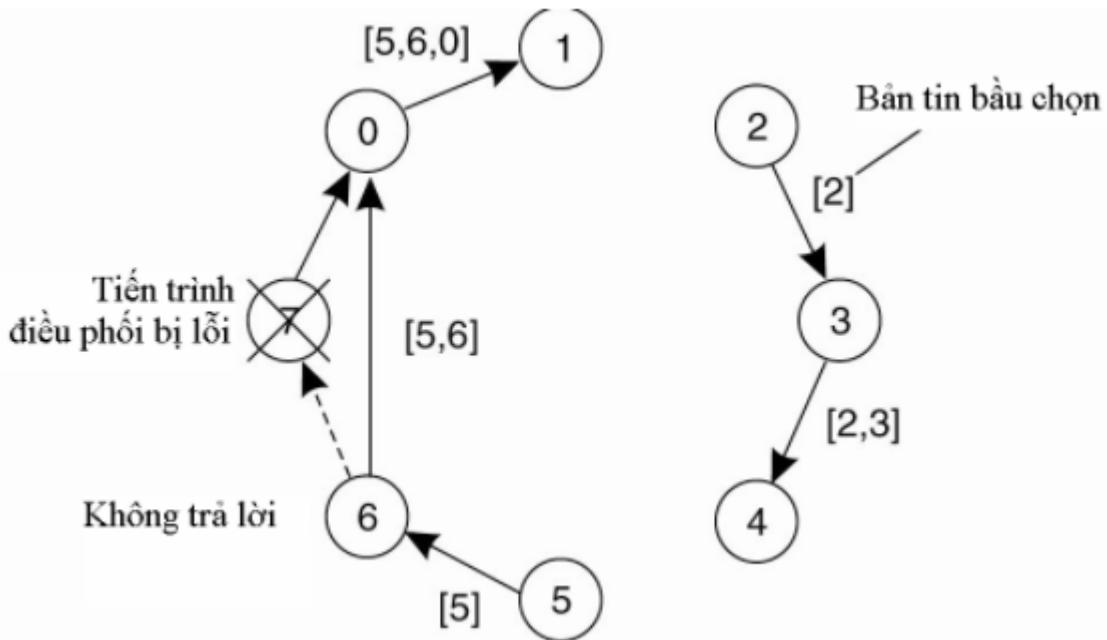
- Tiến trình P gửi thông điệp ELEC đến tất cả các tiến trình có ID cao hơn
- Nếu không có tiến trình nào phản hồi thì P sẽ trở thành tiến trình điều phối
- Nếu có một tiến trình có ID cao hơn phản hồi thì tiến trình đó sẽ đảm nhiệm vai trò điều phối.



## Thuật toán vòng

Các bước của giải thuật bao gồm:

1. Một tiến trình bất kỳ khởi sướng gửi thông điệp bầu chọn đến các tiến trình đang hoạt động gần nhất, quá trình gửi theo một hướng nhất định.
2. Thăm dò liên tiếp trên vòng cho đến khi tìm được một nút còn tồn tại.
3. Mỗi tiến trình sẽ gắn định danh của mình vào thông điệp gửi.
4. Thông điệp sẽ đi qua tất cả các tiến trình đang hoạt động và cuối cùng sẽ trở về tiến trình khởi sướng, khi đó sẽ chọn được một tiến trình có định danh cao nhất và gửi thông điệp đến tiến trình đã được chọn.



Bầu chọn môi tr khong dây

Bầu chọn trong các hệ thống qui mô lớn

## Chương 5 : tiến trình, ảo hóa , di trú mã trong các HTPT

### Tiến trình

một chương trình đang chạy thì được gọi là tiến trình

nhiệm vụ chính của hệ điều hành là quản lý và lập lịch cho các tiến trình

Tiến trình tạo nên các khối chức năng trong hệ thống phân tán

Ví dụ, trong hệ thống phân tán sẽ phải thường xuyên áp dụng kỹ thuật đa luồng (multithread) nhằm nâng cao hiệu suất hoạt động của hệ thống

với kỹ thuật đó quá trình trao đổi thông tin khách/chủ và các xử lý cục bộ được thực hiện đồng thời với nhau.

<Slide d\_Process

## Lời gọi HT

- Khái niệm: là tập lệnh mở rộng do hệ điều hành cung cấp xác định giao diện giữa hệ điều hành và các chương trình người sử dụng.
- Phân loại:
  - Phong tỏa (Blocking System call) là lời gọi hệ thống mà sau khi được gọi bởi tiến trình người sử dụng thì tiến trình này bị dừng lại cho đến khi thực hiện xong lời gọi hệ thống.
  - Không phong tỏa (Non – Blocking System call): sau khi gọi, điều khiển được trả lại cho tiến trình gọi và tiến trình này tiếp tục thực hiện song song với lời gọi hệ thống.

## Vấn đề khi SD tiến trình

### CÁC VẤN ĐỀ NÀY SINH KHI SỬ DỤNG TIẾN TRÌNH

- Thực hiện tính trong suốt tương tranh, hệ điều hành đã phải trả giá khá cao:
  - Tạo không gian địa chỉ hoàn toàn độc lập: Xóa bộ nhớ, sao chép dữ liệu mới của tiến trình vào bộ nhớ đó
  - Chuyển CPU giữa hai tiến trình đòi hỏi chi phí cao: Bên cạnh việc ghi nhớ ngữ cảnh CPU (giá trị các thanh ghi, biến đệm chương trình, con trỏ ngăn xếp...) hệ điều hành còn phải thay đổi các thanh ghi của đơn vị bộ nhớ (MMU) và bộ nhớ đệm (TBL)
  - Khi chạy nhiều tiến trình, có thể xảy ra quá trình swap: chuyển dữ liệu giữa ổ đĩa và bộ nhớ (RAM)
- Việc chia nhỏ tiến trình bởi hệ điều hành là chưa đủ, cần thiết phải chia nhỏ tiến trình hơn nữa để dễ dàng xây dựng phần mềm hơn và đồng thời đạt được hiệu năng cao hơn

## Khái niệm luồng

### KHÁI NIỆM LUỒNG (THREAD)

Giống như tiến trình, một luồng thực hiện đoạn mã của mình độc lập với các luồng khác.

Khác với tiến trình, một luồng không cố đạt được mức trong suốt cao, nếu điều đó làm suy giảm hiệu năng.

Nói chung, mỗi luồng chỉ duy trì thông tin tối thiểu để cho phép các luồng chia sẻ CPU.

Ngữ cảnh của luồng thường chỉ bao gồm ngữ cảnh của CPU, các thông tin khác bị bỏ qua. Lập trình viên phải tự xây dựng cơ chế bảo vệ dữ liệu cho mỗi luồng.

## Cài đặt luồng

### CÀI ĐẶT LUỒNG

Luồng thường được cài đặt dưới dạng gói luồng (thread package), các gói đó chứa các thao tác tạo/hủy luồng và thao tác trên các biến đồng bộ như mutex hoặc biến điều kiện

Hai cách tiếp cận cơ bản:

- Xây dựng thư viện luồng chạy hoàn toàn trong chế độ người dùng. Chi phí thấp trong việc tạo/hủy luồng, việc chuyển ngữ cảnh luồng chỉ cần thực hiện vài chi thị nhưng khi thực hiện lời gọi phong tỏa hệ thống sẽ phong tỏa ngay tiến trình chứa luồng
- Phần lõi (kernel) biết và lập lịch cho các luồng, chi phí cao khi chuyển ngữ cảnh luồng

Lai ghép mức lõi và mức người dùng (LWP-tiến trình hạng nhẹ): Là tiến trình hạng nặng (mức lõi) cung cấp gói luồng cho mức người dùng

## Luồng trong HTPT

### LUỒNG TRONG CÁC HỆ PHÂN TÁN

- Luồng cung cấp các phương tiện thuận lợi cho phép phong tỏa lời gọi hệ thống mà không phong tỏa tiến trình của nó
- Dựa trên đặc điểm này, có thể sử dụng luồng để duy trì đồng thời nhiều kênh truyền thông logic.
- Trong hệ thống phân tán cần phải xây dựng đa luồng cả ở hai phía khách và chủ

Đa luồng máy khách/chủ

## Ảo hóa

Nếu máy tính chỉ có một bộ vi xử lý thì việc chạy đồng thời nhiều tiến trình chỉ là ảo giác

một thời điểm bộ vi xử lý chỉ có thể thực thi được một lệnh

bằng cách chuyển rất nhanh từ tiến trình này sang tiến trình khác hoặc từ luồng này sang luồng khác tạo ra cảm giác các chúng chạy song song với nhau.

ảo hóa xuất phát từ nhu cầu thực tế, tài nguyên vật lý chỉ có một nhưng **nhiều thành phần** cùng sử dụng chung tài nguyên và **thành phần nào cũng muốn tài nguyên đó thuộc về mình**

**sinh vấn đề tương tranh giữa các tiến trình**

kỹ thuật ảo hóa đã được áp dụng vừa để đáp ứng nhu cầu sở hữu riêng tài nguyên của các tiến trình và đồng thời không để xảy ra tranh chấp trong hệ thống.

## Vai trò ảo hóa

Các **hệ thống máy tính** được tổ chức theo mô hình phân tầng, **tầng thấp hơn sẽ cung cấp giao diện lập trình** cho các **tầng mức cao hơn**

Sự ảo hóa ở đây thể hiện bằng cách mở rộng **hoặc thay thế** giao diện hiện hành để **bắt chước hành vi** của **hệ thống khác**.

## VAI TRÒ ẢO HÓA TRONG HỆ PHÂN TÁN

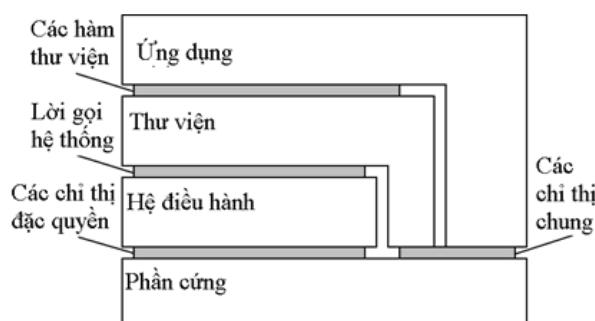


### Kiến trúc máy ảo

phân biệt bốn loại giao diện sau:

- Giao diện giữa phần cứng và phần mềm bao gồm các chỉ thị lệnh mà bất kỳ chương trình nào cũng có thể gọi.
- Giao diện giữa phần cứng và phần mềm bao gồm các chỉ thị lệnh mà chỉ một số chương trình có đặc quyền mới được phép gọi, ví dụ hệ điều hành.
- Giao diện gồm các lời gọi hệ thống do hệ điều hành cung cấp
- Giao diện gồm các lời gọi thư viện hình thành trong thư viện giao diện lập trình ứng dụng (API), trong nhiều trường hợp các lời gọi hệ thống ẩn trong thư viện này.

Bốn loại giao diện kể trên được thể hiện trên hình 5.5 và đó là những vấn đề cốt yếu của ảo hóa để bắt chước hành vi của các giao diện.



## Phương pháp ảo hóa

### CÁC PHƯƠNG PHÁP ẢO HÓA

Tiến trình máy ảo (Process Virtual Machine): cung cấp cho ứng dụng tập các chỉ thị đã được trừu tượng hóa. Về cơ bản, việc ảo hóa chỉ được thực hiện cho 01 tiến trình

Giám sát máy ảo (VMM- Virtual Machine Monitor):

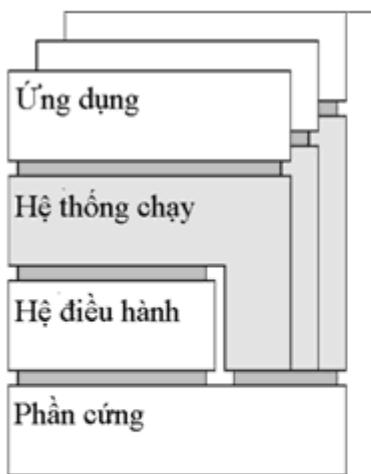
Tạo một lớp bao bọc toàn bộ phần cứng, cho phép nhiều ứng dụng có thể đồng thời chạy trên lớp này.

VMM ngày càng trở nên quan trọng trong các hệ thống phân tán, nó không những đảm bảo tính bảo mật mà còn cung cấp tính năng khả chuyển cho hệ thống

đặt ảo hóa bằng hai hình thức: **Máy ảo tiến trình** (Process Virtual Machine) và **Giám sát máy ảo** (Virtual Machine Monitor)

Hình thức thứ nhất xây dựng hệ thống thực hiện thao thời gian

Hình thức thứ hai dựa trên kiến trúc phân tầng, nó che đậy hoàn toàn phần cứng nhưng cung cấp đầy đủ các chỉ thị lệnh của phần cứng gọi là giao diện



Tiến trình máy ảo với nhiều thể hiện của các tổ hợp (ứng dụng, hệ thống chạy)



Giám sát máy ảo với nhiều thể hiện của các tổ hợp (ứng dụng, hệ điều hành)

Hình 5.6 Hai hình thức ảo hóa

## Máy khách

### MÁY KHÁCH

Nhiệm vụ chính của máy khách là cung cấp các phương tiện để người sử dụng truy nhập vào máy chủ  
Hai phương pháp thường sử dụng:

- Xây dựng giao thức tương ứng với từng dịch vụ
- Xây dựng lớp trung gian chung cho toàn bộ các dịch vụ

## Ứng dụng mạng vs Gt riêng, chung

### Máy chủ

### Di trú mã

đề cập đến vấn đề di chuyển các thành phần khác của tiến trình

Để tăng hiệu suất và độ linh hoạt của hệ thống, đôi khi phải di chuyển di chuyển các chương trình hoặc đoạn mã chương trình, quá trình đó gọi là di trú mã.

### DI TRÚ MÃ

Truyền thông trong hệ thống phân tán cho đến nay đang giới hạn ở việc truyền dữ liệu.

Trong một số tình huống, việc truyền chương trình (thậm chí đang chạy) sẽ đơn giản hóa việc thiết kế và đồng thời tăng hiệu năng và độ linh hoạt của hệ thống.

Thông thường, di trú mã trong hệ thống phân tán được thực hiện dưới dạng di trú tiến trình, toàn bộ tiến trình được chuyển sang máy khác. Việc di chuyển tiến trình đang chạy là công việc khá phức tạp.

Tư tưởng cơ bản của việc di trú mã dựa trên hiệu năng tổng thể, chuyển tiến trình từ máy tính đang có tài cao sang máy tính đang chịu tải thấp hơn.

## DI TRÚ MÃ

Tải của hệ thống dựa trên hàng đợi CPU, bộ nhớ RAM...

Thuật toán phân bổ tái xem xét tới việc định vị và phân bổ lại các nhiệm vụ nhưng phải xem xét tới việc sử dụng tài nguyên của mỗi máy tính (số lượng bộ xử lý, RAM...).

Trong các hệ thống phân tán, việc tối ưu hóa khả năng tính toán ít được coi trọng hơn việc giảm thiểu truyền thông.

Việc di trú mã có thể thực hiện bằng giải pháp chia sẻ xử lý giữa máy khách và máy chủ, máy khách có thể áp dụng các giải pháp xử lý song song. Điều này còn góp phần nâng cao tính linh hoạt của hệ thống phân tán.

## Các giải pháp di trú mã

### CÁC GIẢI PHÁP DI TRÚ MÃ

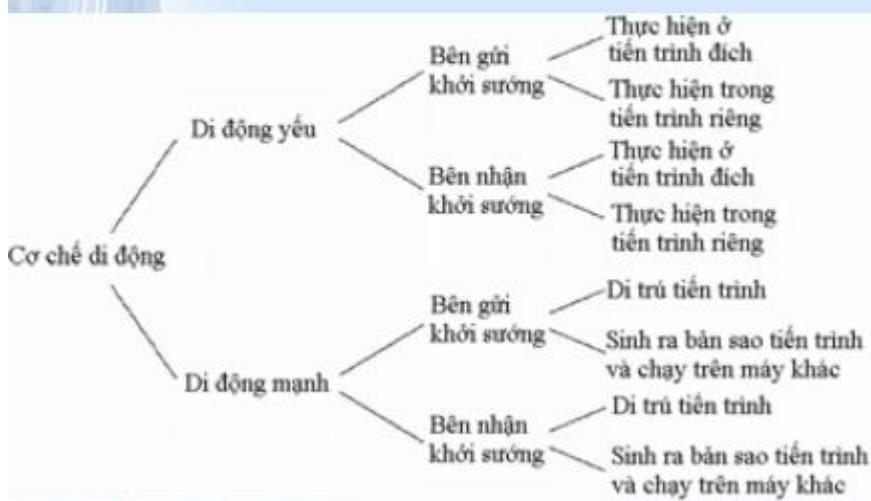
**Di động yếu:** chỉ truyền phần mã và một số dữ liệu khởi động của tiến trình. Chương trình được truyền đi luôn được bắt đầu từ trạng thái khởi động, chỉ yêu cầu máy đích có thể thực thi yêu cầu đó.

**Di động mạnh:** truyền cả phần mã và phần thực thi. Một tiến trình đang chạy có thể được dừng lại rồi chuyển đến một máy khác và tiếp tục thực hiện tiếp tiến trình đó, giải pháp này khó thực thi hơn.

**Di trú do bên gửi khởi sướng:** Di trú được khởi động từ máy mà phần mã của tiến trình được lưu trữ hoặc đang thực hiện trên máy gửi. Di trú này hoàn thành khi tải xong chương trình.

**Di trú do bên nhận khởi sướng :** Di trú mã ban đầu của máy gửi. Thực hiện đơn giản hơn so với di trú được khởi sướng từ bên gửi.

## CÁC PHƯƠNG ÁN DI TRÚ MÃ



## Di trú và tài nguyên cục bộ

### DI TRÚ VÀ TÀI NGUYÊN CỤC BỘ

- Khác với di trú phần mã và phần thực thi, việc di trú phần tài nguyên đòi hỏi những thay đổi nhất định.
- Phân biệt ba loại nhúng tiến trình với tài nguyên
  - Mạnh: Tiến trình tham chiếu đến tài nguyên bằng định danh (ví dụ URL)
  - Yêu: Nhúng bằng giá trị, tiến trình không bị ảnh hưởng nếu được cung cấp nguồn tài nguyên khác cùng giá trị (ví dụ thư viện lập trình)
  - Yêu nhất: Tiến trình chỉ cần kiểu tài nguyên (ví dụ thiết bị ngoại vi...)
- Phân biệt ba loại nhúng tài nguyên với máy tính
  - Tài nguyên không định kèm: Dễ dàng di chuyển giữa các máy (ví dụ các tập tin)
  - Tài nguyên gắn kèm: Khó di chuyển giữa các máy (ví dụ CSDL cục bộ, trang web..)
  - Tài nguyên cố định: Những tài nguyên gắn chặt với từng máy (ví dụ thiết bị, công...

.....continue

## Di trú trong hệ thống không đồng nhất

### DI TRÚ TRONG CÁC HỆ THỐNG KHÔNG ĐỒNG NHẤT

Đây các trang nhớ sang máy mới và gửi lại các trang đã thay đổi gần nhất trong quá trình di trú.

Dùng máy ảo hiện hành, di trú bộ nhớ và khởi tạo máy ảo mới

Cho phép máy ảo mới kéo các trang cần thiết (cho phép các tiến trình bắt đầu trên máy ảo mới ngay lập tức sau chép các trang bộ nhớ theo nhu cầu)

Quản lý cụm máy chủ

giải pháp này chỉ phù hợp với mô hình nhỏ,

Giả sử cụm có N nút và xác suất xảy ra lỗi trên mỗi nút là p, xác xuất m nút đồng thời xảy ra lỗi được tính theo công thức của lý thuyết xác suất thống kê như sau:

$$P(X = m) \binom{N}{m} p^m (1-p)^{N-m}$$

Ví dụ cụm máy chủ gồm N=1000 nút, xác suất xảy ra lỗi trên mỗi máy chủ là p=0.001, áp dụng công thức trên để tính xác suất không có máy nào bị lỗi sẽ cho kết quả như sau:

$$\begin{aligned} P(X = 0) & \binom{1000}{0} p^0 (1-p)^{1000} \\ & = \frac{1000!}{0!1000!} (0.001)^0 (0.999)^{1000} \\ & = 0.367695 \end{aligned}$$

## C6 QUẢN TRỊ GIAO TÁC VÀ ĐIỀU KHIỂN TƯƠNG TRANH

### Khái niệm giao tác

Giao tác gồm một hoặc nhiều câu lệnh truy xuất dữ liệu và chúng được thực hiện như một đơn vị thống nhất, nghĩa là không thể thêm hay bớt bất kỳ câu lệnh nào.

Giao tác được chia thành **giao tác phẳng, giao tác lồng ghép và giao tác phân tán**.

Giao tác có thể gồm nhiều câu lệnh đọc/ghi dữ liệu và phải đảm bảo tính chất sau:

- **Tính chất nguyên tử (Atomic):** Đối với thế giới bên ngoài thì giao tác không thể chia nhỏ hơn được, các lệnh trong giao tác đều được thực hiện hoặc

không có lệnh nào được thực hiện. Tính nguyên tử đòi hỏi nếu một giao tác bị hủy giữa chừng thì kết quả của các lệnh đã thực hiện sẽ bị hủy bỏ.

- **Tính chất nhất quán (Consistent):** giao tác không xâm phạm tính chất bất biến của hệ thống, nghĩa là luôn phải đảm bảo hệ thống toàn vẹn.
- **Tính chất cô lập (Isolated):** Khi có nhiều giao tác đồng thời thực hiện thì mỗi giao tác sẽ hoạt động độc lập và không làm ảnh hưởng đến các giao tác khác.
- **Tính chất bền vững (Durable):** Khi giao tác đã cam kết thì các thay đổi đối với nó không phải là tạm thời mà là những thay đổi bền vững, nếu không cam kết thì sẽ tự động phục hồi, coi như chưa từng thực hiện các lệnh trong giao tác.

### Giao tác phẳng

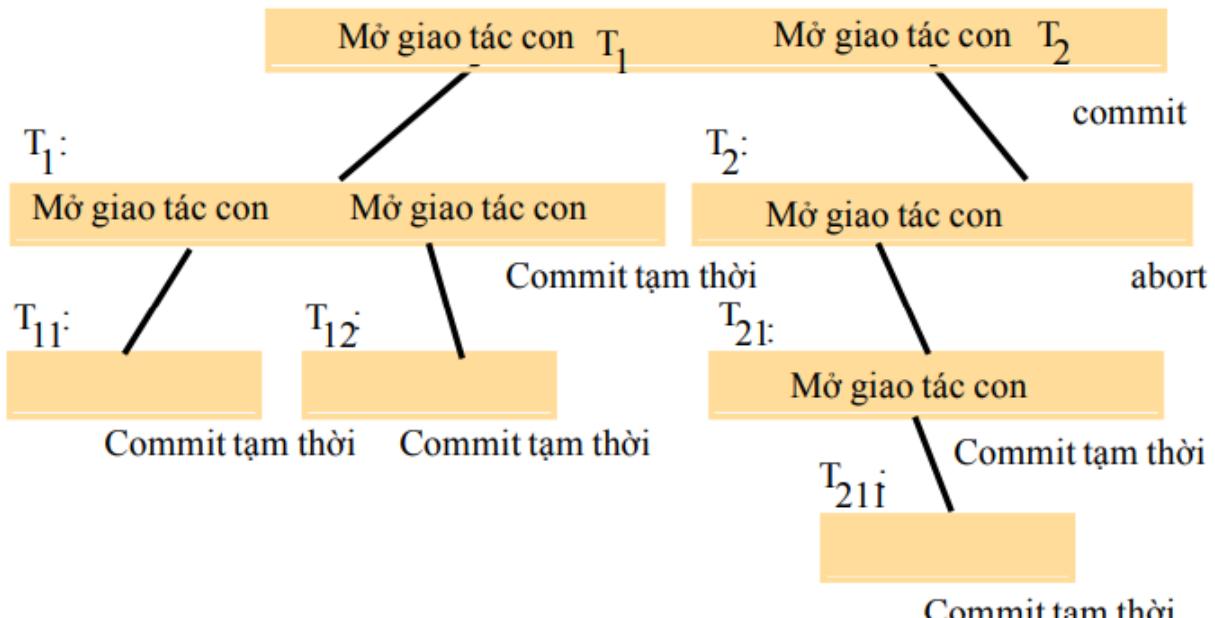
- ✓ Một giao tác thỏa mãn bốn tính trên gọi là giao tác phẳng
- ✓ **hạn chế chính** của giao tác phẳng là chúng **không cho phép tách riêng** các kết quả **được cam kết hay hủy bỏ**
- ✓ bắt đầu **bằng lệnh mở giao tác**
- ✓ tiếp theo là các chuỗi các lệnh thao tác với dữ liệu và kết thúc đóng giao tác bằng lệnh **cam kết (commit) hoặc hủy bỏ (abort, rollback)**.

### Các giao tác lồng nhau

cấu thành từ một số giao tác con

**mỗi giao tác con cũng có thể thực thi một hay nhiều giao tác con** của chính nó

T : Giao tác mức cao nhất



Hình 2.44 Giao tác lồng nhau

✓ **các giao tác con** trên cùng một mức có thể **chạy đồng thời** với nhau, các lệnh cam kết hoặc hủy bỏ của các giao tác con hoàn toàn độc lập với nhau.

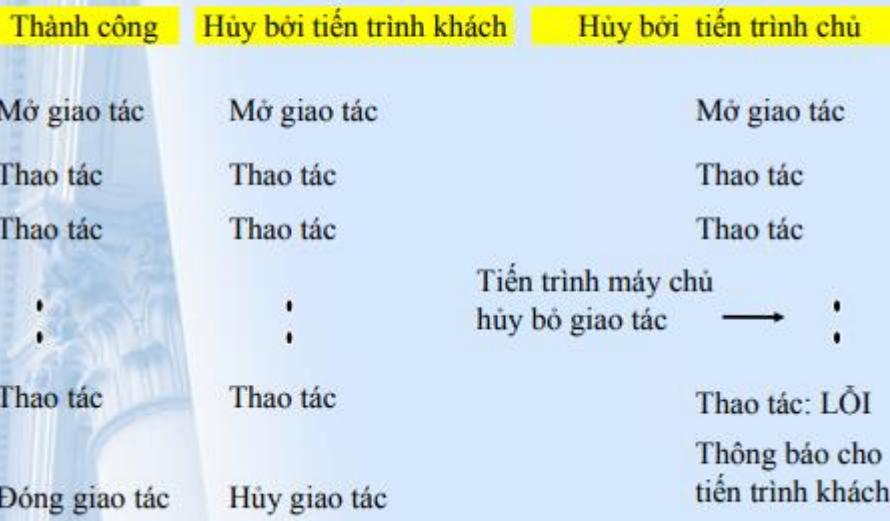
## Hoạt động của giao tác

### HOẠT ĐỘNG TRONG GIAO TÁC

- openTransaction() -> trans; Mở giao tác  
Bắt đầu giao tác và phân phát định danh của giao tác, định danh này sẽ được dùng trong các hoạt động khác của giao tác
- closeTransaction(trans) -> (commit, abort); Đóng giao tác //COMMIT, ABORT/ROLLBACK  
Kết thúc giao tác, nếu giá trị trả về là **commit** nghĩa là giao tác đã hoàn thành và dữ liệu đã được cập nhật theo đúng yêu cầu, nếu giá trị trả về là **abort** thì mọi hoạt động trong giao tác coi như chưa hề thực hiện
- abortTransaction(trans); Hủy giao tác  
Hủy bỏ giao tác //ABORT/ROLLBACK

## Quá trình sống giao tác

### QUÁ TRÌNH SỐNG CỦA GIAO TÁC



## Vấn đề tương tranh

- Mất mát dl

MẤT MÁT KHI CẬP NHẬT	
Tài khoản của A, B, C có giá trị lần lượt là \$100, \$200 và \$300. A và C cùng chuyển cho B số tiền bằng 10% giá trị tài khoản của B	
A chuyển cho B balance = b.getBalance(); b.setBalance(balance*1.1); a.withdraw(balance/10)	C chuyển cho B balance = b.getBalance(); b.setBalance(balance*1.1); c.withdraw(balance/10)
balance = b.getBalance(); \$200	balance = b.getBalance(); \$200
b.setBalance(balance*1.1); \$220	b.setBalance(balance*1.1); \$220
a.withdraw(balance/10) \$80	c.withdraw(balance/10) \$280
Số tiền trong tài khoản của B lẽ ra phải là \$242	
Do a và C cùng chuyển đồng thời cùng lấy số dư của b = 200 gây mất mát dl	

### 1. 1. Mất dữ liệu cập nhật (lost update)

➤ Ví dụ: Cho quan hệ HANGHOA (MaHH, Tên HH, ĐVT, SLTon)

➤ Giả sử: SLTon = 20

Transaction		Giá trị		
T <sub>1</sub> (bán hàng)	T <sub>2</sub> (bán hàng)	x <sub>1</sub>	x <sub>2</sub>	SLTon
Begin Transaction	Begin Transaction			20
Read(SLTon)→x <sub>1</sub>	Read(SLTon)→x <sub>2</sub>	20	20	20
x <sub>1</sub> - 10→x <sub>1</sub>	x <sub>2</sub> - 3→x <sub>2</sub>	10	20	20
Write x <sub>1</sub> →SLTon	.....	10	17	20
Commit T <sub>1</sub>	Write x <sub>2</sub> →SLTon	10	17	20
	Commit T <sub>2</sub>			

↳ Lost Update: Thao tác cập nhật của T<sub>1</sub> xem như bị mất, không được ghi nhận (do những thay đổi của các giao tác khác ghi đè lên).

- Kết quả k đồng nhất

thực hiện theo qui trình sau sẽ xảy ra hiện tượng đọc kết quả không đồng nhất:

## 1. Một số vấn đề điều khiển đồng thời

### 1.2. Đọc dữ liệu chưa commit (uncommitted data)

Ví dụ: Cho quan hệ HANGHOA ( MaHH, TenHH, ĐVT, SLton )

SLton = 20

Transaction		Giá trị		
T <sub>1</sub> (nhập hàng)	T <sub>2</sub> (bán hàng)	x <sub>1</sub>	x <sub>2</sub>	SLton
Begin transaction				20
Read (SLton) → x <sub>1</sub>		20		20
x <sub>1</sub> + 100 → x <sub>1</sub>		120		20
Write x <sub>1</sub> → SLton		120		
	Begin transaction	120		
	Read (SLton) → x <sub>2</sub>	120		
	x <sub>2</sub> - 5 → x <sub>2</sub>	120		
	Write x <sub>2</sub> → SLton	120		
	Commit T <sub>2</sub>	120		
Rollback T <sub>1</sub>				

↳ Transaction T<sub>1</sub> sửa đổi dòng X nhưng chưa commit, transaction T<sub>2</sub> đọc dòng X. Transaction T<sub>1</sub> rollback những gì thay đổi trên dòng X → dữ liệu mà Transaction T<sub>2</sub> đang đọc chưa hề tồn tại.

## 1. Một số vấn đề điều khiển đồng thời

### 1.4. Vấn đề bóng ma (phantom)

Ví dụ: T<sub>1</sub> thực hiện việc chuyển tiền từ tài khoản A sang tài khoản B. Khi T<sub>1</sub> mới chỉ thực hiện thao tác trừ số tiền ở tài khoản A (chưa cộng số tiền vào tài khoản B) thì T<sub>2</sub> muốn xem tổng số tiền ở 2 tài khoản → Tổng số tiền không chính xác.

Transaction	
T <sub>1</sub>	T <sub>2</sub>
Begin transaction	
Read (A)	
A = A - 50	
Write (A)	
	Begin transaction
	Read (A)
	Read (B)
	Print (A+B)
	Commit T <sub>2</sub>
Read (B)	
B = B+50	
Write (B)	
Commit T <sub>1</sub>	

### KẾT QUẢ KHÔNG ĐỒNG NHẤT

A chuyển \$100 cho B a.withdraw(100) b.deposit(100)	Kiểm tra số dư tài khoản aBranch.branchTotal()
---	---

**① a.withdraw(100);      \$100**

**rút tiền**

**② total = a.getBalance()      \$100**

**③ total = total+b.getBalance()      \$300**

**④ total = total+c.getBalance()**

b.deposit(100) **⑤ \$300**      :

**Init : A =100, B =200,  
C=300**

Nguyên nhân của hai lỗi trên được xác định là do trình tự thực hiện các thao tác không theo mong muốn và xung đột đã xảy ra trong khi thực hiện các thao tác.

## Biện pháp tuần tự hóa

### BIỆN PHÁP TUẦN TỰ HÓA THỰC HIỆN

Tổ hợp lại thứ tự thực hiện các yêu cầu

A chuyển cho B balance = b.getBalance(); b.setBalance(balance*1.1); a.withdraw(balance/10)	C chuyển cho B balance = b.getBalance(); b.setBalance(balance*1.1); c.withdraw(balance/10)
---	---

balance = b.getBalance(); \$200  
b.setBalance(balance\*1.1); \$220  
a.withdraw(balance/10)      \$80

balance = b.getBalance(); \$220  
b.setBalance(balance\*1.1); \$242  
c.withdraw(balance/10)      \$278

Số tiền trong tài khoản của B là \$242

Thực hiện a chuyển b  
trong khi chưa cập nhật lại số  
tiền B rồi mới chuyển

### BIỆN PHÁP TUẦN TỰ HÓA THỰC HIỆN

A chuyển \$100 cho B a.withdraw(100) b.deposit(100)	Kiểm tra số dư tài khoản aBranch.branchTotal()
---	---

a.withdraw(100);      \$100

b.deposit(100)      \$300

total = a.getBalance()      \$100  
total = total+b.getBalance()      \$400  
total = total+c.getBalance()  
:

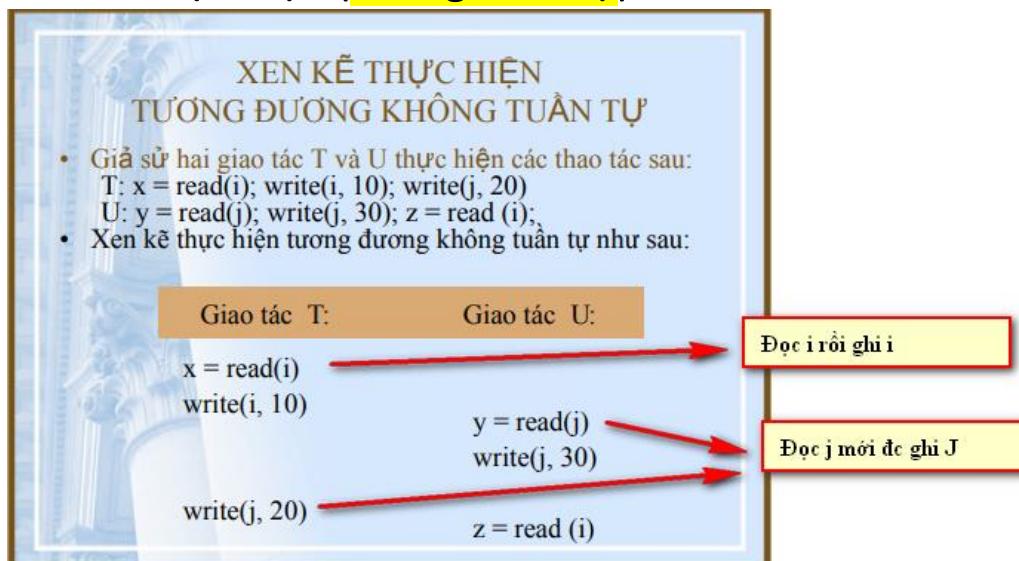
Thực hiện kiểm tra tk  
A trước rồi chuyển cho B

Tất cả cặp thao tác xung đột (kết quả tổ hợp phụ thuộc vào thứ tự thực hiện) phải được thực hiện theo 1 thứ tự nhất định

## Các luật xung đột

CÁC LUẬT XUNG ĐỘT CHO THAO TÁC ĐỌC VÀ GHI		
T1	T2	Xung đột
Đọc	Đọc	Không
Đọc	Ghi	Có
Ghi	Ghi	Có

## Xen kẽ thực hiện (không tuần tự)



## Phục hồi DL ban đầu khi hủy thực hiện

Nhận tất cả giao tác : ComMIT

Hủy toàn bộ : AboRT

## cÁc vấn đề phục hồi

- Đọc giá trị bẩn (Dirty reads)
- Hủy phân tầng
- Ghi trước
- Thực hiện giao tác chặt chẽ
- Sử dụng phiên bản tạm

- Đọc giá trị bẩn

## ĐỌC GIÁ TRỊ BẢN (Dirty Read)

Giao tác T	Giao tác U
a.getBalance() a.setBalance(balance + 10)	a.getBalance() a.setBalance(balance + 20)
balance = a.getBalance() \$100	
a.setBalance(balance + 10) \$110	balance = a.getBalance() \$110
	a.setBalance(balance + 20) \$130
	commit transaction
abort transaction	

- Ghi đè giá trị chưa commit

## GHI ĐÈ GIÁ TRỊ CHƯA COMMIT

Giao tác T	Giao tác U
a.setBalance(105)	a.setBalance(110)
	\$100
a.setBalance(105)	\$105
	a.setBalance(110) \$110

## Các phương pháp điều khiển tương tranh

- Điều khiển tương tranh bí quan,
- điều khiển tương tranh lạc quan
- điều khiển tương tranh dựa trên nhãn thời gian

## CÁC PHƯƠNG PHÁP ĐIỀU KHIỂN TƯƠNG TRANH

- Sử dụng khóa (Locking):** Khóa được dùng để sắp xếp thứ tự các giao tác theo thứ tự đi đến của chúng trên cùng một khoản mục dữ liệu.
- Điều khiển tương tranh lạc quan (Optimistic concurrency control):** Cho phép các thao tác được tiếp tục thực hiện cho đến khi sẵn sàng COMMIT và sau đó thực hiện kiểm tra xem có thao tác nào bị xung đột hay không.
- Thứ tự nhãn thời gian:** Sử dụng nhãn thời gian để sắp thứ tự giao tác theo thời gian bắt đầu của chúng.

Khóa loại trừ

Là cơ chế đơn giản để tuân tự hóa

Phân loại :

Khóa 2 pha : Pha 1 giữ khóa, pha 2 mở khóa

Khóa 2 pha nghiêm ngặt : Giữ khóa cho tới khi COMMIT hoặc ABORT

### KHÓA LOẠI TRỪ

Giao tác T	Giao tác U		
Thao tác	Khóa	Thao tác	Khóa
$balance = b.getBalance()$		$balance = b.getBalance()$	
$b.setBalance(bal * 1.1)$		$b.setBalance(bal * 1.1)$	
$a.withdraw(bal / 10)$		$c.withdraw(bal / 10)$	
$openTransaction$		$openTransaction$	
$bal = b.getBalance()$	lock B		
$b.setBalance(bal * 1.1)$			
$a.withdraw(bal / 10)$	lock A	$bal = b.getBalance()$	waits for T's lock on B
$closeTransaction$	unlock A, B	...	
			lock B
		$b.setBalance(bal * 1.1)$	
		$c.withdraw(bal / 10)$	lock C
		$closeTransaction$	unlock B, C

## Điều khiển tương tranh bằng khóa

### Định nghĩa khóa

#### KHÓA

- Các thao tác đọc trên cùng một khoản mục dữ liệu không gây nén xung đột.
- Khóa loại trừ giảm tính tương tranh hơn mức cần thiết
- Mô hình đọc nhiều/Ghi một (many reader/single write) phân biệt hai loại khóa: khóa chia sẻ và khóa ghi.
- Khóa hai pha hoặc hai pha chặt chẽ vẫn thường được dùng để đảm bảo tính tuần tự thực hiện

#### Khái niệm khóa:

Lock là một đặc quyền truy xuất (access privilege) lên các đơn vị dữ liệu của các giao tác mà bộ quản lý khóa có thể trao cho một giao tác hay thu hồi lại. Khi một giao tác đã khóa (lock) trên một đơn vị dữ liệu nào đó thì các giao tác khác không được phép truy cập đến đơn vị dữ liệu đó cho đến khi nó nhả khóa(unlock).

Khi một giao tác T thực hiện việc lock đơn vị dữ liệu A, ta nói T đang giữ lock A.

Thông thường tại mỗi thời điểm, chỉ có một tập con các đơn vị dữ liệu bị khóa, vì vậy bộ quản lý khóa có thể lưu các khóa hiện hành trong một bảng (lock table) với các mẫu tin có dạng: (A, L, T) ( giao tác T có một khóa kiểu L trên đơn vị dữ liệu A);

## 2. Kỹ thuật khóa đơn giản

# 5. Điều khiển tương tranh bằng cơ chế khóa

## 5.2. Kỹ thuật khóa đơn giản

➤ Một giao tác khi có yêu cầu truy xuất đến đơn vị dữ liệu thì phải phát ra yêu cầu xin khóa (lock) trên đơn vị dữ liệu đó, nếu yêu cầu này được chấp thuận thì được quyền thao tác và như vậy các giao tác khác sẽ không được phép truy cập đến đơn vị dữ liệu đó cho đến khi giao tác giữ khóa được unlock

### Không khóa

T <sub>1</sub>	T <sub>2</sub>	A (5)
Read A		5
A=A+1	Read A	5
	A=A+1	5
	Write A	6 6
Write A		

T <sub>1</sub>	T <sub>2</sub>	Nhận xét
Lock A Read A  A=A+1  Write A Unlock A	Lock A Read A  A=A+1 Write A  Unlock A	T <sub>2</sub> chờ T <sub>1</sub> Unlock  T <sub>1</sub> sẽ hoàn tất trước khi T <sub>2</sub> bắt đầu. Khi đó A=7

### Kỹ thuật khóa

a. Giới thiệu: - Một giao tác khi có yêu cầu truy xuất đến đơn vị dữ liệu thì phải phát ra yêu cầu xin khóa (lock) trên đơn vị dữ liệu đó, nếu yêu cầu này được chấp nhận thì được quyền thao tác, và như vậy các giao tác khác sẽ không được phép truy cập đến đơn vị dữ liệu đó cho đến khi giao tác giữ khóa được unlock.

b. Ví dụ: Xét 2 giao tác T1 và T2. Mỗi giao tác truy xuất 1 đơn vị dữ liệu A được giả sử là mang giá trị số nguyên, rồi cộng thêm 1 vào A. Hai giao tác này là các thực hiện của chương trình P dưới đây: P: Lock (A) Read( A) A=A+1 Write (A) Unlock (A) Nếu T1 → bắt đầu trước, nó yêu cầu khóa trên A. Giả sử rằng không có giao tác nào đang khóa A, bộ quản lý khóa sẽ cho nó khóa này. Như vậy bây giờ chỉ có T1 mới có thể truy xuất A. Nếu T2 bắt đầu trước khi T1 chấm dứt thì T2 phải đợi. Chỉ khi T1 thực hiện lệnh unlock(A), hệ thống mới cho phép T2 tiến hành. Kết quả là T1 hoặc T2 sẽ hoàn tất trước khi giao tác kia bắt đầu, và tác dụng là cộng 2 vào A.

### 3. Kỹ thuật khóa đọc/ viết ( readlock/ writelock)

## 5. Điều khiển tương tranh bằng cơ chế khóa

### 5.2. Kỹ thuật khóa đơn giản

➤ Ví dụ

T <sub>1</sub>	T <sub>2</sub>
Lock A Read A A=A+1	Lock A Read A Unlock A
Write A Lock B Read B B=B+1 Write B Unlock B Unlock A	

➤ Nhận xét

✓ Nếu không phân biệt khóa cho thao tác đọc hay viết thì rõ ràng sẽ có nhiều giao tác phải chờ để được quyền khóa trên 1 đơn vị dữ liệu.

✓ Thực tế là nhiều khi một giao tác chỉ cần lấy giá trị của 1 đơn vị dữ liệu nhưng không thay đổi giá trị đó.

✓ Vì vậy để giảm bớt tình huống phải chờ khi các giao tác cùng đọc dữ liệu, người ta đề nghị tách yêu cầu khóa thành 2 yêu cầu khóa riêng biệt.

### 5.3. Kỹ thuật khóa đọc/viết (Readlock/Writelock)

\* Khóa để đọc (hay Shared lock): một giao tác T chỉ muốn đọc 1 đơn vị dữ liệu A sẽ thực hiện lệnh RLOCK(A), ngăn không cho bất kỳ giao tác khác ghi giá trị mới của A trong khi T đã khóa A. Tuy nhiên các giao tác khác vẫn có thể giữ 1 khóa đọc trên A cùng lúc với T.

\* Khóa để ghi (Exclusive lock): một giao tác T muốn thay đổi giá trị của 1 đơn vị dữ liệu A đầu tiên sẽ lấy khóa ghi bằng cách thực hiện lệnh WLOCK(A). Khi 1 giao tác đang giữ 1 khóa ghi trên 1 đơn vị dữ liệu, các giao tác khác không thể lấy được khóa đọc hay khóa ghi trên A cùng một lúc với T.

- Cả hai khóa đọc và khóa ghi đều được loại bỏ bằng lệnh UNLOCK

## 5. Điều khiển tương tranh bằng cơ chế khóa

### 5.3. Kỹ thuật khóa đọc/viết (Readlock/Writelock)

#### Điều kiện để xin khóa đọc/viết

➤ Một yêu cầu xin RLOCK(A) chỉ được chấp thuận nếu A chưa bị khóa bởi 1 WLOCK trước đó.

➤ Một yêu cầu xin WLOCK(A) chỉ được chấp thuận nếu A được tự do.

#### Bảng tương thích các loại khóa

		Lock đang được giữ	
		R-lock	W-lock
Giao tác yêu cầu lock	R-lock	Yes	No
	W-lock	No	No

### 5.3. Kỹ thuật khóa đọc/viết (Readlock/Writelock)

Ví dụ (Với đơn vị dữ liệu B=2)

Nhân xét

T <sub>1</sub>	T <sub>2</sub>	Ghi chú
Rlock B Read B → a <sub>1</sub> Unlock B		a <sub>1</sub> =B=2
	Rlock B Read B → a <sub>2</sub> Unlock B a <sub>2</sub> + 1 → a <sub>2</sub> Wlock B Write a <sub>2</sub> → B Unlock B	a <sub>2</sub> =B=2 a <sub>2</sub> =3 B=a <sub>2</sub> =3 a <sub>1</sub> =3 B=a <sub>1</sub> =3
a <sub>1</sub> + 1 → a <sub>1</sub> Wlock B Write a <sub>1</sub> → B Unlock B		

➤ Lịch thao tác không khả tuần tự nên xảy ra lost update.

➤ Việc sử dụng cơ chế lock không đủ để bảo đảm tính khả tuần tự cho lịch thao tác

➤ Để giải quyết tính không khả tuần tự. Dùng chiến lược lock 2 pha, hoặc yêu cầu các giao tác lock các đơn vị dữ liệu theo 1 thứ tự cố định nào đó.

- Nếu không phân biệt khóa cho thao tác đọc hay viết thì rõ ràng sẽ có nhiều giao tác phải chờ để được quyền khóa trên một đơn vị dữ liệu. Thực tế là nhiều khi một giao tác chỉ cần lấy giá trị của một đơn vị dữ liệu nhưng không thay đổi giá trị đó. Vì vậy để giảm bớt tình huống phải chờ khi các giao tác cùng đọc dữ liệu, người ta đề nghị tách yêu cầu khóa thành 2 yêu cầu khóa riêng biệt: + **Khóa để đọc (hay Shared lock)**: một giao tác T chỉ muốn đọc một đơn vị dữ liệu A sẽ **thực hiện lệnh RLOCK(A)**, ngăn không cho bất kỳ giao tác khác ghi giá trị mới của A trong khi T đã khóa A. Tuy nhiên các giao tác khác vẫn có thể giữ 1 khóa đọc trên A cùng lúc với T.
- + **Khóa để ghi (hay Exclusive lock)**: một giao tác muốn **thay đổi giá trị** của một đơn vị dữ liệu A đầu tiên sẽ phải lấy **khóa ghi** bằng cách thực hiện lệnh **WLOCK(A)**. Khi một giao tác đang giữ một khóa ghi trên một đơn vị dữ liệu, các giao tác khác **không thể** lấy được khóa đọc hoặc khóa ghi trên A cùng lúc với T. Cả hai khóa đọc và khóa ghi đều được loại bỏ bằng lệnh **UNLOCK**.
  - Điều kiện để xin khóa đọc/ viết: + Một yêu cầu xin **RLOCK(A)** chỉ được chấp nhận nếu A chưa bị khóa bởi một **WLOCK** trước đó.
  - + Một yêu cầu xin **WLOCK(A)** chỉ được chấp thuận nếu A được tự do.
  - Ma trận tương thích các loại khóa:  
Lock đang được giữ R-lock W-lock Giao tác yêu cầu lock R-lock Yes No W-lock No No - Từ ma trận tương thích, chúng ta thấy: một đơn vị dữ liệu tại một thời điểm có thể có nhiều transaction giữ Rlock nhưng chỉ có tối đa một transaction giữ Wlock.
  - Việc sử dụng cơ chế lock không đủ để đảm bảo tính khả tuần tự cho lịch giao tác.

KHẢ NĂNG TƯƠNG THÍCH KHÓA		
Cho một đối tượng		Khóa yêu cầu
Đã thiết lập khóa		read      write
	none	OK      OK
	read	OK      wait
	write	wait      wait

4. Một số vấn đề trong kỹ thuật khóa a.

## Livelock(Khóa chờ)

- Hệ thống trao và buộc khóa các đơn vị dữ liệu không thể hoạt động một cách thát thường, nếu không thì các hệ thống không mong muốn có thể xảy ra. Giả sử như ví dụ trên, khi T1 giải phóng khóa trên A, khóa này lại trao lại cho T2. Điều gì sẽ xảy ra nếu như trong khi T2 đang đợi nhận khóa, một giao tác T3 khác cũng xin khóa trên A, và T3 lại được trao khóa này trước T2. Rồi sau khi T3 được trao khóa trên A thì lại có một giao tác T4 xin khóa trên A...và rất có thể T2 sẽ phải đợi mãi và chẳng bao giờ nhận được khóa trong khi luôn có một giao

## 5. Điều khiển tương tranh bằng cơ chế khóa

### 5.4. Các vấn đề trong kỹ thuật khóa

#### Livelock

➤ Giả sử khi  $T_1$  giải phóng khóa trên A, khóa này được trao lại cho  $T_2$ . Điều gì sẽ xảy ra nếu như trong khi  $T_2$  đang đợi nhận khóa, một giao tác  $T_3$  khác cũng xin một khóa trên A, và  $T_3$  lại được trao khóa này trước  $T_2$ . Rồi sau khi  $T_3$  được trao khóa trên A thì lại có 1 giao tác  $T_4$  xin khóa trên A,... Và rất có thể  $T_2$  phải đợi mãi và chẳng bao giờ nhận được khóa trong khi luôn có 1 giao tác khác giữ khóa trên A, dù rằng có một số lần  $T_2$  có cơ hội nhận được khóa trên A.

**Như vậy,** Livelock là trường hợp 1 giao tác chờ được cấp quyền lock trên 1 đơn vị dữ liệu nào đó mà không xác định được thời điểm được đáp ứng yêu cầu.



**Yêu cầu** hệ thống khi trao khóa phải ghi nhận tất cả các thỉnh cầu chưa được đáp ứng, và khi đơn vị dữ liệu A được mở khóa thì trao cho các giao tác đã xin đầu tiên trong số những giao tác đang đợi khóa A. Và khi đó bộ lập lịch (schedulers) sẽ tiến hành thực hiện cơ chế: giao tác nào yêu cầu trước thì được đáp ứng trước (FIFO).

## Quản lý khóa

### QUẢN LÝ KHÓA

- Khi một thao tác truy nhập đối tượng bên trong giao tác:
  - a) Nếu đối tượng chưa bị khóa, đối tượng sẽ được khóa và thao tác tiếp tục
  - b) Nếu đã có giao tác khác khóa đối tượng và khóa đó xung đột với thao tác thì thao tác đó phải chờ cho đến khi mở khóa
  - c) Nếu đã có giao tác khác khóa đối tượng và khóa đó không xung đột với thao tác thì khóa ở chế độ chia sẻ và thao tác tiếp tục
  - d) Nếu đối tượng được khóa trong cùng một giao tác thì nâng mức khóa nếu cần thiết và thao tác tiếp tục (Ngăn chặn nâng mức khóa đối với khóa xung đột, sử dụng luật b)
- Khi thực hiện COMMIT hoặc ABORT, máy chủ sẽ mở khóa tất cả các đối tượng đã khóa cho giao tác

# Khóa Chết

## KHÓA CHẾT (Deadlocks)

- Sử dụng khóa có thể dẫn đến trạng thái *deadlock*.
- Deadlock là trạng thái, trong đó mỗi thành viên của nhóm các giao tác đang chờ thành viên khác giải phóng khóa
- Có thể sử dụng đồ thị *wait-for* để thể hiện các quan hệ chờ giữa các giao tác tương tranh tại máy chủ

### Deadlock

T <sub>1</sub>	T <sub>2</sub>	Nhận xét
Lock A	Lock B	- T <sub>1</sub> chờ T <sub>2</sub> unlock B
Lock B	Lock A	- T <sub>2</sub> chờ T <sub>1</sub> unlock A → Deadlock

➤ T<sub>1</sub> yêu cầu và được trao khóa trên A, còn T<sub>2</sub> yêu cầu và được trao khóa trên B. Do đó khi T<sub>1</sub> yêu cầu khóa trên B, nó sẽ phải đợi vì T<sub>2</sub> đã khóa B. Tương tự khi T<sub>2</sub> yêu cầu khóa trên A, nó cũng buộc phải đợi vì T<sub>1</sub> đã khóa A. Kết quả là không một giao tác nào tiếp tục hoạt động được: mỗi giao tác đều phải đợi cho giao tác kia mở khóa, và chúng đều phải đợi nhưng chẳng bao giờ nhận được khóa yêu cầu.

**Deadlock** là tình trạng trong đó những giao tác có liên quan không thể thực hiện tiếp các thao tác của nó mà phải chờ nhau mãi



Bỏ qua (Hủy, làm lại)

Ngăn ngừa (Chờ theo chiều nhất định)

Phát hiện (Đồ thị chờ)

## Deadlock với các khóa ghi (WRITE)

Giao tác T		Giao tác U	
Thao tác	Khóa	Thao tác	Khóa
a.deposit(100);	Khóa ghi A	b.deposit(200)	Khóa ghi B
b.withdraw(100)			
...	Chờ khóa U's trên B	a.withdraw(200);	Chờ khóa T's trên A
...		...	
...		...	

Các kỹ thuật phòng ngừa deadlock

### CÁC KỸ THUẬT PHÒNG NGỪA Deadlock

- Khi bắt đầu giao tác sử dụng khóa tất cả các mục dữ liệu
- Mỗi giao tác yêu cầu khóa trên các mục dữ liệu theo thứ tự đã định nghĩa trước
- Mỗi khóa có khoảng thời gian giới hạn, sau thời gian đó sẽ không được bảo vệ

### PHÁT HIỆN Deadlock

- Có thể phát hiện Deadlock bằng cách tìm các chu kỳ trong đồ thị chờ khóa
- Hai vấn đề thiết kế:
  - Thường xuyên kiểm tra sự tồn tại của chu kỳ khóa
  - Chọn giao tác để hủy bỏ

## Tăng tính tương tranh

**TĂNG TÍNH TƯƠNG TRANH**

- Khóa hai phiên bản:** Cho phép một giao tác tạm thời ghi dữ liệu trong khi các giao tác khác đọc dữ liệu chưa được COMMIT. Các thao tác đọc chỉ chờ nếu có giao tác khác đang COMMIT dữ liệu đó
- Khóa có thứ bậc:** Ở mỗi mức, việc thiết lập khóa cha có kết quả như đặt tất cả các khóa con tương đương

**TƯƠNG THÍCH KHÓA**

Cho một đối tượng	Khóa cần đặt		
	read	write	commit
Đã đặt khóa	none	OK	OK
	read	OK	OK
	write	OK	wait
	commit	wait	wait

## Khóa phân cấp

**KHÓA PHÂN CẤP**

- Trong thực tế tồn tại các quan hệ cha con
- Ví dụ đơn giản: thời gian trong tuần

Week				
Monday	Tuesday	Wednesday	Thursday	Friday
9:00–10:00	10:00–11:00	11:00–12:00	12:00–13:00	13:00–14:00
14:00–15:00	15:00–16:00			

**TƯƠNG THÍCH KHÓA PHÂN CẤP**

Cho một đối tượng	Khóa cần thiết lập			
	read	write	I-read	I-write
Thiết lập khóa	none	OK	OK	OK
	read	OK	wait	OK
	write	wait	wait	wait
	I-read	OK	wait	OK
	I-write	wait	wait	OK

## Nhược điểm cơ chế khóa

**NHƯỢC ĐIỂM CỦA CƠ CHẾ KHÓA**

- Việc duy trì khóa tăng thêm tải xử lý, một số khóa có thể không cần thiết
- Giảm tính tương tranh để tránh deadlock hoặc giữ khóa cho tới khi kết thúc giao tác (tránh hủy bỏ theo tầng)

## Điều khiển tương tranh lạc quan

Điều khiển tương tranh lạc quan cho phép các giao tác được phép tiếp tục nếu không có xung đột với các giao tác khác, nếu phát hiện xung đột thì sẽ hủy bỏ giao tác nào đó. Mỗi giao tác gồm ba pha:

- Pha đọc: sử dụng bản tạm thời cho mỗi mục dữ liệu được cập nhật
- Pha phê chuẩn: Kiểm tra xem có xung đột hay không
- Pha ghi: Nếu được phê chuẩn không có xung đột thì chuyển bản dữ liệu tạm thời thành vĩnh viễn

## Các luật đọc ghi

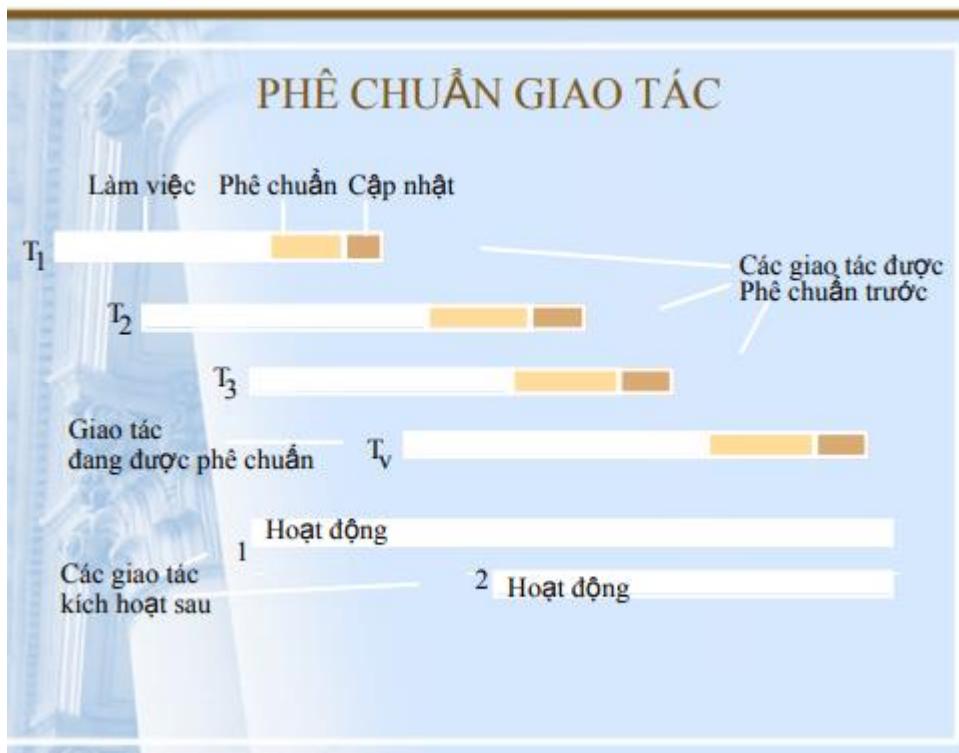
### CÁC LUẬT ĐỌC/GHI

T <sub>v</sub>	T <sub>i</sub>	Luật
write	read	1. T <sub>i</sub> không được đọc đối tượng được T <sub>v</sub> ghi
read	write	2. T <sub>i</sub> không được ghi đối tượng được T <sub>v</sub> đọc
write	write	3. T <sub>i</sub> không được ghi đối tượng được T <sub>v</sub> ghi T <sub>v</sub> không được ghi đối tượng được T <sub>i</sub> ghi

## Phê chuẩn giao tác

### PHÊ CHUẨN GIAO TÁC

- Để được phê chuẩn, mỗi giao tác được gán số thứ tự (tăng dần) khi bước vào pha phê chuẩn
- Giao tác luôn luôn kết thúc pha đọc của mình sau tất cả các giao tác có số thứ tự thấp hơn.
- Các pha phê chuẩn có thể chồng nhau nhưng số hiệu phải được gán tuần tự.
- Tất cả các pha ghi được thực hiện tuần tự theo số hiệu đã được gán, như vậy không cần kiểm tra xung đột ghi-ghi
- Không nên tái sử dụng số thứ tự đã gán cho giao tác
- Hai dạng phê chuẩn:
  - Phê chuẩn ngược: Kiểm tra với các giao tác đã bước vào giai đoạn phê chuẩn trước nó.
  - Phê chuẩn xuôi: Kiểm tra với các giao tác sau nhưng vẫn đang hoạt động



Phê chuẩn ngc

## PHÊ CHUẨN GIAO TÁC NGUỐC

- Tất cả các thao tác đọc của các giao tác đã được thực hiện trước khi giao tác  $T_v$  bắt đầu không thể bị ảnh hưởng bởi thao tác ghi của  $T_v$ .
- Phê chuẩn giao tác của  $T_v$  chỉ cần kiểm tra tập đọc của  $T_v$  với tập ghi của các giao tác trước.
- $\text{startT}_{n+1}$  là số hiệu giao tác lớn nhất đã được phê chuẩn,  $\text{finishT}_n$  là số hiệu giao tác lớn nhất đã được gán tại thời điểm  $T_v$  bước vào giai đoạn phê chuẩn

```
boolean valid = true;
for (int Ti=startTn+1;Ti>=finishTn;Ti--)
{
    if (Tập đọc của Tv giao với tập ghi của Ti)
    {
        valid = false;
    }
}
```

## Phê chuẩn xuôi

### PHÊ CHUẨN GIAO TÁC XUÔI

- Luật Ghi-Đọc đương nhiên thỏa mãn vì các giao tác được gán số hiệu sau chỉ được thực hiện sau khi giao tác trước đã kết thúc thao tác đọc.
- Tập ghi của Tv được so sánh với tập đọc của tất cả các giao tác vẫn đang hoạt động
- Giả sử sau giao thắc Tv có các giao tác active1... activeN đang hoạt động

```
boolean valid = true;
for (int T_id = active1; T_id <= activeN; T_id++)
{
    if (tập ghi của T_v giao với tập đọc của T_id)
    {
        valid = false;
    }
}
```

## So sánh 2 giao tác phê chuẩn

### SO SÁNH PHÊ CHUẨN XUÔI VÀ NGƯỢC

- Phê chuẩn ngược:
  - Bỏ qua các giao tác đang được phê chuẩn
  - Phải nhớ tập ghi của các giao tác đã COMMIT có thể xung đột với các giao tác đang hoạt động
  - So sánh tập đọc có thể rất lớn so với tập ghi cũ
- Phê chuẩn xuôi:
  - Ba lựa chọn:
    - Hoãn phê chuẩn cho đến khi giao tác xung đột (hoạt động) kết thúc
    - Hủy bỏ tất cả các giao tác đang xung đột và COMMIT giao tác đang được phê chuẩn
    - Hủy bỏ giao tác đang được phê chuẩn
  - Trong thời gian phê chuẩn, phải cho phép giao tác mới được phép bắt đầu
  - So sánh tập ghi nhỏ với tập đọc của các giao tác đang hoạt động

## Điều khiển tương tranh dựa trên nhãn thời gian

Sử dụng giải thuật gán nhãn thời gian Lamport sẽ đảm bảo cho mỗi thao tác được gán một nhãn thời gian duy nhất và như vậy sẽ không còn hiện tượng tương tranh, tuy nhiên cần phải có một thành phần điều phối để ưu tiên những giao tác có thời gian thực hiện nhanh và thường xuyên hơn.

## THÚ TỰ NHÃN THỜI GIAN

Mỗi giao tác được gán nhãn thời gian duy nhất khi bắt đầu

Mỗi thao tác mang nhãn thời gian của giao tác đã được cấp phát và được phê chuẩn khi thực hiện.

Nếu thao tác không được thực hiện thì giao tác sẽ bị hủy bỏ ngay lập tức.

## LUẬT ĐỌC/GHI

Luật  $T_c = T_i$

1. write read  $T_c$  không được ghi đổi tượng đang được  $T_i$  đọc trong đó  $T_i > T_c$   
điều này yêu cầu  $T_c \geq$  nhãn thời gian đọc lớn nhất của đổi tượng.
2. write write  $T_c$  không được ghi đổi tượng đang được  $T_i$  ghi trong đó  $T_i > T_c$   
điều này yêu cầu  $T_c \geq$  Nhãn thời gian ghi của đổi tượng đã COMMIT
3. read write  $T_c$  không được đọc đổi tượng đang được  $T_i$  ghi trong đó  $T_i > T_c$   
điều này yêu cầu  $T_c >$  Nhãn thời gian ghi của đổi tượng đã COMMIT

## 4. Sắp xếp các giao tác bằng nhãn thời gian

### 4.1. Khái niệm nhãn thời gian (timestamp)

➤ Là 1 con số được phát sinh bởi bộ lập lịch, được gán cho mỗi giao tác để chỉ định thời điểm bắt đầu thực hiện giao tác. **Nhãn thời gian có tính chất duy nhất và tăng dần** ( $T_i < T_j \leftrightarrow t_{Ti} < t_{Tj}$ )

➤ Nhãn thời gian của đơn vị dữ liệu: là nhãn thời gian của giao tác cuối cùng có truy cập đến đơn vị dữ liệu đó thành công, hay là nhãn thời gian cao nhất trong số các giao tác có truy cập thành công đến đơn vị dữ liệu đó.

$T_1$	$T_2$	$T_3$	$t_A$
<b>Read A</b>	<b>Read A</b>	<b>Read A</b>	$t_A = t_{T1}$ $t_A = t_{T2}$ $t_A = t_{T3}$

## 4. Sắp xếp các giao tác bằng nhãn thời gian

### 4.2. Thuật toán sắp xếp toàn phần

```
Procedure Read (Ti, A)
Begin
If tA ≤ tTi then
    Thực hiện thao tác đọc
    tA := tTi
Else
    Rollback Ti và bắt đầu lại với
    nhãn thời gian mới
End Proc
```

```
Procedure Write (Ti, A)
Begin
If tA ≤ tTi then
    Thực hiện thao tác ghi
    tA := tTi
Else
    Rollback Ti và bắt đầu lại với
    nhãn thời gian mới
End Proc
```

Nhận xét

#### Ghi chú:

- t<sub>A</sub>: nhãn thời gian của đơn vị dữ liệu A
- t<sub>T<sub>i</sub></sub>: nhãn thời gian của giao tác T<sub>i</sub>
- Ban đầu t<sub>A</sub> = 0 khi chưa có giao tác truy cập

➤ Ví dụ 1: t<sub>T<sub>1</sub></sub> = 100, t<sub>T<sub>2</sub></sub> = 120, t<sub>A</sub> = 0

T <sub>1</sub>	T <sub>2</sub>	t <sub>A</sub>
Read A		t <sub>A</sub> = 100
A = A + 1	Read A	t <sub>A</sub> = 120
	A = A + 1	t <sub>A</sub> > t <sub>T<sub>1</sub></sub> nên T <sub>1</sub>
	Write A	phải rollback và
		bắt đầu lại với
		timestamp mới

➤ Thuật toán chỉ sắp xếp thứ tự các giao tác (thời gian bắt đầu) không nhằm sắp xếp trình tự thực hiện các thao tác trong mỗi giao tác.

## 4. Sắp xếp các giao tác bằng nhãn thời gian

### 4.2. Thuật toán sắp xếp toàn phần

➤ Ví dụ 2:  $t_{T_1} = 100$ ,  $t_{T_2} = 120$ ,  $t_A = 0$

$T_1$	$T_2$	$t_A$
Read A	Read A	$t_A = 100$ $t_A = 120$ $t_A = 120$
Read A	Read A	$t_A > t_{T_1}$ nên $T_1$ phải rollback và bắt đầu lại với timestamp mới

Nhận xét



➤ Trong trường hợp này rõ ràng không xảy ra đụng độ  $T_1$  không cần phải rollback và bắt đầu lại với timestamp mới. Tuy nhiên do thuật toán sắp xếp toàn phần không phân biệt tính chất của thao tác dữ liệu là Read hay Write nên  $T_1$  vẫn bị rollback và bắt đầu lại.

**Như vậy:** Thuật toán sắp xếp toàn phần : không quan tâm đến tính chất của thao tác dữ liệu (Read/Write) nên chỉ có 1 nhãn thời gian duy nhất cho 1 đơn vị dữ liệu. Nếu quan tâm đến tính chất của thao tác dữ liệu thì cần 2 nhãn thời gian cho 1 đơn vị dữ liệu tương ứng với thao tác đọc và ghi trên đơn vị dữ liệu đó.

## 4. Sắp xếp các giao tác bằng nhãn thời gian

### 4.3. Thuật toán sắp xếp từng phần

- Mỗi đơn vị dữ liệu A có 2 nhãn thời gian RTS và WTS. Ban đầu, RTS = WTS = 0
- RTS(A) là nhãn thời gian của giao tác có timestamp lớn nhất truy cập (Read) thành công lên A
- WTS(A) là nhãn thời gian của giao tác có timestamp lớn nhất truy cập (Write) thành công lên A

$T_1(t_{T_1} = 100)$	$T_2(t_{T_2} = 120)$	RTS(A)	WTS(A)
Read A	Read A $A=A+1$	100 120	
$A=A+1$	Write A	<u>T1 rollback</u>	
Write A			120

```

Procedure Read ( $T_i, A$ )
Begin
If WTS(A) <=  $t_{T_i}$  then
    Thực hiện thao tác đọc
    RTS(A) := Max(RTS(A),  $t_{T_i}$ )
Else
    Rollback  $T_i$  và bắt đầu lại với nhãn thời gian mới
End Proc

```

```

Procedure Write ( $T_i, A$ )
Begin
If (RTS(A) <=  $t_{T_i}$ ) and (WTS(A) <=  $t_{T_i}$ ) then
    Thực hiện thao tác ghi
    WTS(A) :=  $t_{T_i}$ 
Else
    Rollback  $T_i$  và bắt đầu lại với nhãn thời gian mới
End Proc

```

## 4. Sắp xếp các giao tác bằng nhãn thời gian

### 4.3. Thuật toán sắp xếp từng phần

- Nhận xét : Trong thuật toán sắp xếp từng phần, số lượng giao tác bị rollback lại ít hơn trong thuật toán sắp xếp toàn phần (do nếu 2 giao tác chỉ thực hiện «Đọc» thì không gây ra đụng độ)

$T_1$	$T_2$	Nhận xét
(1) Read A	(2) Read A	Thuật toán sắp xếp toàn phần : $T_1$ bị rollback ở (3) Thuật toán sắp xếp từng phần : không có rollback
(3) Read A		

$T_1$	$T_2$	Nhận xét
(1) Read A	(2) Write A	Thuật toán sắp xếp toàn phần : $T_1$ bị rollback ở (3) Thuật toán sắp xếp từng phần : $T_1$ bị rollback ở (3)
(3) Read A		

## So sánh các pp dk tuong tranh

### SO SÁNH ĐIỀU KHIỂN TƯƠNG TRANH

Mục	Khóa	Nhân thời gian
Tính chất	Bí quan	Lạc quan
Thứ tự quyết định	Động	Tĩnh
Giao tác được lợi	Ghi nhiều hơn đọc	Chi đọc
Giải quyết xung đột	Chờ, hủy bỏ	Hủy bỏ

## Chương 7 :PHỤC HỒI VÀ TÍNH CHỊU LỖI

### Khái niệm tính chịu lỗi

Tính chịu lỗi của một hệ thống liên quan mật thiết **tới khái niệm tính tin cậy** của hệ thống, một hệ thống được coi là có khả năng chịu lỗi nếu đáp ứng được các tiêu chí sau:

- Khả năng sẵn sàng phục vụ của hệ thống
- Độ tin cậy của hệ thống
- Độ an toàn của hệ thống
- Khả năng bảo trì hệ thống

### QUAN ĐIÈM CƠ BẢN VỀ TÍNH CHỊU LỖI

- Tính chịu lỗi liên quan tới hệ thống đáng tin cậy.
- Hệ thống đáng tin cậy thể hiện trong các khía cạnh sau:
  - **Tính sẵn sàng** (Availability): luôn sẵn sàng thực thi tốt khi có yêu cầu gửi đến.
  - **Tính tin cậy** (Reliability): có khả năng hoạt động trong một thời gian dài mà không bị gián đoạn, không xảy ra lỗi.
  - **Tính an toàn** (Safety): khi xảy ra lỗi cũng không dẫn tới sai lệch (thông tin, điều khiển).
  - **Khả năng bảo trì** (Maintainability): Dễ dàng sửa chữa khi có lỗi, có khả năng phục hồi lại được sau khi có lỗi. Nếu sự phục hồi này diễn ra tự động thì có thể nói hệ thống này cũng có tính sẵn sàng cao.
- Tính chịu lỗi còn có liên quan tới khái niệm **điều khiển lỗi** (*Fault control*). Điều khiển lỗi bao gồm ngăn ngừa, loại bỏ và dự báo lỗi.

### Phân loại lỗi

Xét về tần suất, lỗi được phân chia thành ba loại sau:

- **Lỗi nhất thời:** Là loại lỗi xảy ra một lần sau đó không xuất hiện lại.
- **Lỗi lặp:** Là loại lỗi mà chúng xuất hiện nhiều lần, có thể theo chu kỳ hoặc không. Lỗi này thường gây ra các hậu quả nghiêm trọng vì chúng rất khó xác định được.
- **Lỗi lâu dài:** Là loại lỗi vẫn tồn tại ngay cả khi thành phần gây lỗi đó đã được sửa chữa

**PHÂN LOẠI LỖI**

- **Lỗi nhất thời (Transient faults):** lỗi xuất hiện một lần rồi biến mất. Để khắc phục thì chỉ cần thực hiện lại hoạt động gây nên lỗi này.
- **Lỗi chập chờn (Intermittent faults):** lỗi xuất hiện rồi biến mất, sau đó lại xuất hiện lại và cứ tiếp tục như vậy. Lỗi này thường gây ra các hậu quả nghiêm trọng vì rất khó xác định nguyên nhân gây lỗi. Để khắc phục cần phải dò tìm và phân tích.
- **Lỗi thường xuyên (Permanent faults):** Là loại lỗi vẫn tồn tại ngay cả khi đã xác định được nguyên nhân và sửa thành phần gây lỗi.

Xét về mức độ ảnh hưởng, lỗi được phân thành năm loại sau:

- **Máy chủ bị lỗi nghiêm trọng :** Máy chủ có thể bị treo và dừng mọi hoạt động cung cấp dịch vụ, cách duy nhất để giải quyết vấn đề này là khởi động lại máy chủ.
- **Máy chủ xử lý lỗi:** Máy chủ không nhận được yêu cầu từ máy khách, nhận được yêu cầu nhưng không thể trả lời hoặc không thể trả về kết quả xử lý cho máy khách. Trong trường hợp này cần phải kiểm tra kết nối giữa máy khách và máy chủ, thường thường đó là các lỗi mạng.
- **Lỗi thời gian:** Máy chủ không đáp ứng được thời gian xử lý yêu cầu của máy khách.
- **Lỗi kết quả xử lý:** Thông tin trả về của máy chủ không chính xác, có thể là giá trị sai.
- **Lỗi không xác định:** Máy chủ trả về các giá trị không mong muốn và những giá trị đó chưa từng xảy ra, việc xác định nguyên nhân và biện pháp xử lý các lỗi này rất khó.

## CÁC MÔ HÌNH LỖI

- Lỗi sụp đổ (crash failure):** máy chủ bất ngờ bị treo mặc dù trước đó vẫn hoạt động bình thường, cách duy nhất là khởi động lại.
- Lỗi bỏ sót (omission failure):** máy chủ không thể đáp ứng được yêu cầu gửi tới, gồm hai loại:
  - Lỗi nhận thông điệp: máy chủ không nhận được thông điệp từ máy khách. Lỗi này không ảnh hưởng đến trạng thái của máy chủ.
  - Lỗi gửi thông điệp: máy chủ vẫn nhận được yêu cầu và hoàn thành yêu cầu đó nhưng không thể gửi kết quả tới máy khách. Thường máy khách sẽ gửi lại thông điệp, do đó máy chủ cần phải xử lý tránh trùng lặp.
- Lỗi thời gian (timing failure):** máy chủ phản hồi kết quả chậm hơn thời gian cho phép.

## CÁC MÔ HÌNH LỖI

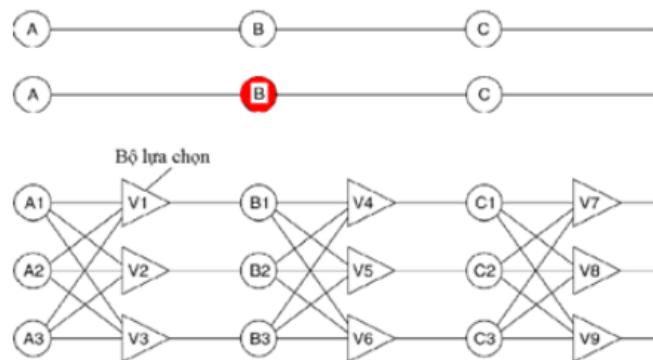
- Lỗi đáp ứng (Response failure):** Máy chủ trả về giá trị không chính xác. Đây là kiểu lỗi rất nghiêm trọng, phân thành hai loại:
  - Lỗi giá trị: Giá trị trả về không chính xác.
  - Lỗi chuyển trạng thái: Tiến trình máy chủ hoạt động không đúng với luồng điều khiển, kết quả trả về ngoại ý.
- Lỗi phức tạp (Arbitrary failure, Byzantine):** có thể xảy ra ở bất kì thời gian nào. Đây là loại lỗi nguy hiểm nhất, tiến trình máy chủ tạo ra kết quả sai mà không thể phát hiện ra được hoặc kết quả sai khi tiến trình máy chủ liên kết với các tiến trình máy chủ khác. Thường thể hiện ở ba dạng:
  - fail-stop: Máy chủ không đưa ra kết quả và các tiến trình khác có thể phát hiện ra điều này. Trường hợp tốt nhất tiến trình máy chủ sẽ thông báo về sự cố, nếu không sẽ chỉ đơn giản ngừng hoạt động.
  - fail-silent: máy chủ đột ngột hoạt động chậm lại các tiến trình không thể biết được máy chủ cộn hoạt động hay không, ảnh hưởng đến hiệu năng của hệ thống.
  - fail-safe: Máy chủ trả về kết quả ngẫu nhiên để các tiến trình khác nhận dạng nhưng không có giá trị sử dụng.

## Các biện pháp đảm bảo tính chịu lỗi

Che giấu lỗi bằng biện pháp dư thừa

- người sử dụng **không hề biết hoặc ít biết** về sự cố đối với mạng dịch vụ

- Dư thừa thông tin:** bổ sung thêm các bit dư thừa để phát hiện lỗi và phục hồi lỗi. Ví dụ trong việc truyền dữ liệu thường thêm vào các bit kiểm tra chẵn lẻ, mã Haming, CRC... để phát hiện lỗi và phục hồi lỗi.
- Dư thừa thời gian:** khi một hoạt động đã được thực hiện, nếu dư thừa thời gian nó có thể được thực hiện lại. Kỹ thuật dư thừa thời gian phù hợp khi lỗi là nhất thời và lỗi chập chờn. Có thể sửa lỗi bằng cách thực hiện lại các giao tác bị lỗi, **Tuy nhiên phải chú ý tính đúng đắn đối với các thao tác thực hiện theo thời gian thực.**
- Dư thừa vật lý:** bổ sung thêm tài nguyên vật lý, thay vì sử dụng một thiết bị thì sử dụng thêm nhiều thiết bị khác để dự phòng



Hình 7.1 Xây dựng hệ thống dư thừa theo kiểu chuyển mạch

Hình 7.1 minh họa kỹ thuật dư thừa theo tương tự như hệ thống chuyển mạch, bình thường hệ thống chỉ bao gồm 3 thành phần chính A, B, C. Để đảm bảo khả năng chịu lỗi, chúng ta thêm các thành phần  $A_i$ ,  $B_i$ ,  $C_i$  tương ứng với A, B, C và duy trì ở trạng thái dự phòng. Khi có lỗi xảy ra, hệ thống sẽ tự động chuyển đến một trong các thành phần dư thừa tương ứng, sau khi khắc phục xong lỗi, hệ thống sẽ chuyển về xử lý trên các thành phần chính của hệ thống.

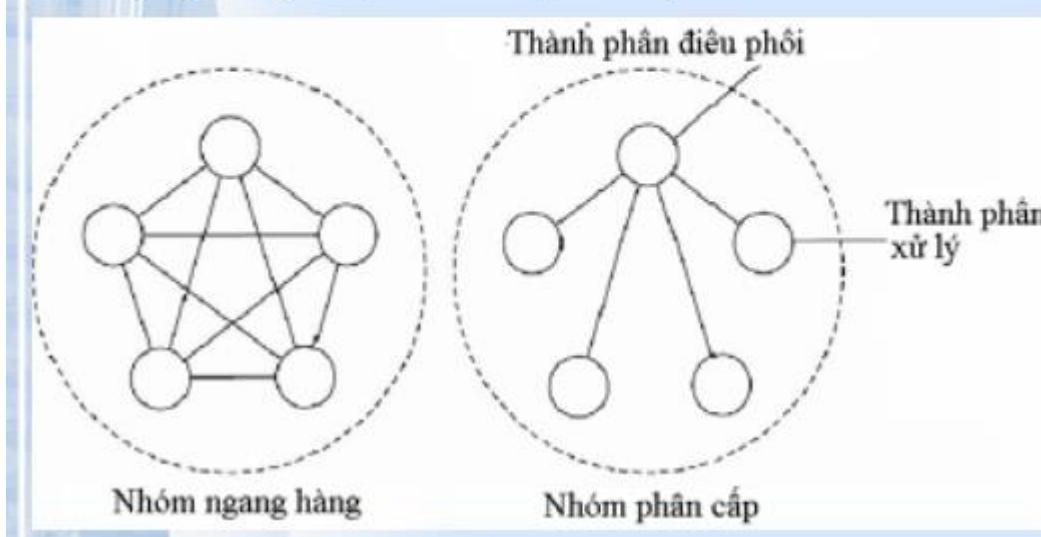
## Nhóm tiến trình bền bỉ

### TIẾN TRÌNH BỀN BỈ (Resilient)

- Cách tiếp cận mấu chốt là gộp các tiến trình giống nhau vào cùng một nhóm.
- Thuộc tính mấu chốt của nhóm là khi nhóm nhận được thông báo thì thông báo này sẽ được gửi tới tất cả các thành viên trong nhóm. Nếu có tiến trình nào trong nhóm bị lỗi thì hy vọng sẽ có các tiến trình khác thay thế.
- Nhóm các tiến trình có thể động, tính động thể hiện ở các mặt sau:
  - Có thể tạo thêm hay hủy bỏ một nhóm.
  - Một tiến trình có thể gia nhập hay rời khỏi nhóm.
  - Một tiến trình có thể là thành viên của nhiều nhóm trong cùng thời một điểm.
- Do tính động của nhóm nên cần phải đưa ra các cơ chế quản lý mối quan hệ giữa các nhóm và các thành viên trong nhóm.

### PHÂN LOẠI NHÓM TIẾN TRÌNH BỀN BỈ

Dựa trên tổ chức bên trong, nhóm phân làm hai loại: Nhóm ngang hàng (flat) và nhóm phân cấp



## NHÓM TIỀN TRÌNH BỀN BỈ PHÂN CẤP

- Trong mỗi nhóm sẽ có một tiến trình giữ vai trò điều phối, còn các tiến trình khác đóng vai trò xử lý. Các tiến trình xử lý chịu sự điều khiển của tiến trình điều phối.
- Khi có yêu cầu gửi đến nhóm, yêu cầu này sẽ được gửi tới tiến trình điều phối. Tiến trình điều phối sẽ quyết định xem tiến trình nào trong nhóm đảm nhiệm công việc đó một cách phù hợp nhất và chuyên yêu cầu nhận được đến tiến trình đó.
- **Ưu điểm:** không bị trễ như kiến trúc ngang hàng.
- **Nhược điểm:** khi tiến trình điều phối gặp sự cố thì toàn bộ hoạt động của nhóm sẽ bị dừng lại.

## NHÓM TIỀN TRÌNH BỀN BỈ NGANG HÀNG

- Tất cả các tiến trình trong nhóm là ngang hàng nhau.
- Khi thực hiện một công việc nào đó sẽ phải có một quá trình bầu cử để xác định xem tiến trình nào phù hợp để thực hiện công việc đó.
- **Ưu điểm:** khi một tiến trình bị lỗi thì chỉ làm cho kích thước của nhóm giảm đi chứ không ảnh hưởng đến hoạt động của cả nhóm.
- **Nhược điểm:** do phải có quá trình bầu cử nên tốn thời gian.

## QUẢN LÝ THÀNH VIÊN TRONG NHÓM

Phương pháp 1 (dùng máy chủ nhóm): Máy chủ này chứa tất cả các thông tin về các nhóm và các thành viên của từng nhóm.

- **Ưu điểm:** hiệu quả, dễ sử dụng
- **Nhược điểm:** nếu máy chủ bị lỗi thì không thể quản lý được toàn bộ hệ thống và các nhóm có thể phải xây dựng lại từ đầu các công việc mình đã thực hiện.

Phương pháp 2 (phương pháp phân tán): Khi tiến trình muốn gia nhập hay rời khỏi nhóm thì nó phải gửi bản tin thông báo tới tất cả các tiến trình khác.

- Phương pháp này không phù hợp với dạng Fail-Stop.
- Vấn đề phức tạp khác là việc gia nhập hoặc rời khỏi nhóm được đồng bộ với thông điệp dữ liệu đang gửi.
  - Khi một tiến trình gia nhập nhóm nó sẽ nhận tất cả các thông điệp từ nhóm đó.
  - Khi một tiến trình rời khỏi nhóm thì nó sẽ không được nhận bất kì thông điệp nào từ nhóm đó nữa và không một thành viên nào của nhóm cũ nhận được các thông điệp từ nó.
- Để giải quyết vấn đề trên cần phải lập thứ tự các thông điệp

### Che giấu lỗi và nhân bản

Che giấu lỗi trong hệ thống phân tán tập trung vào trường hợp có tiến trình bị lỗi, tuy nhiên trước hết cần phải xem xét các trường hợp lỗi truyền tin. Sử dụng nhóm tiến trình cũng là một trong các biện pháp che giấu lỗi, nhóm các tiến trình giống nhau sẽ che giấu một tiến trình nào đó bị lỗi. Nhân bản một tiến trình sau đó gộp chúng thành một nhóm, có hai phương pháp nhân bản: giao thức dựa trên thành phần chính (primary-based protocol) và giao thức dựa trên ghi nhân bản (replicated-write protocol).

Trong phương pháp che giấu lỗi dựa trên thành phần chính, các tiến trình trong nhóm tổ chức theo mô hình phân cấp. Một tiến trình đóng vai trò tiến trình chính có nhiệm vụ điều phối tất cả các thao tác ghi. Nếu tiến trình chính của nhóm dừng hoạt động thì các tiến trình sao lưu sẽ thực hiện giải thuật bầu chọn để lựa chọn tiến trình chính mới. Nếu dựa trên ghi nhân bản thì các tiến trình trong nhóm tổ chức theo mô

hình nhóm ngang hàng, vấn đề nằm ở câu hỏi cần nhân bản với số lượng là bao

## CHE GIẤU LỖI VÀ NHÂN BẢN

- Sử dụng nhóm tiền trình cũng là một trong các biện pháp che giấu lỗi. Nhóm các tiền trình giống nhau sẽ che giấu một tiền trình nào đó bị lỗi.
- Nhân bản một tiền trình sau đó gộp chúng thành một nhóm. Có hai phương pháp nhân bản: giao thức dựa trên thành phần chính (primary-based protocol) và giao thức dựa trên ghi nhân bản (replicated-write protocol)
- **Dựa trên thành phần chính**: Các tiền trình trong nhóm tổ chức theo mô hình phân cấp. Một tiền trình đóng vai trò tiền trình chính có nhiệm vụ điều phối tất cả các thao tác ghi. Nếu tiền trình chính của nhóm dừng hoạt động thì các tiền trình sao lưu sẽ thực hiện giải thuật bầu chọn để lựa chọn tiền trình chính mới.
- **Dựa trên ghi nhân bản**: Các tiền trình trong nhóm tổ chức theo mô hình nhóm ngang hàng. Vấn đề là cần nhân bản với số lượng là bao nhiêu?

nhiêu

### Đồng thuận trong các hệ thống lỗi

- . Vấn đề trở nên phức tạp nếu muốn các tiền trình trong nhóm đạt được thỏa thuận trong một số trường hợp: bầu cử, COMMIT, phân chia nhiệm vụ xử lý...
- . Mục đích chung của thuật toán thỏa thuận phân tán là để tất cả các tiền trình lỗi đạt được sự đồng thuận trên một số vấn đề và thiết lập sự đồng thuận đó trong một số bước hữu hạn, tuy nhiên thực tế rất phức tạp và cần phải phân biệt các trường hợp sau:
  - Các hệ thống đồng bộ đối với các hệ thống không đồng bộ
  - Có ràng buộc độ trễ truyền thông hay không, chỉ có thể ràng buộc nếu thông điệp được phân phát trong một khoảng thời gian xác định trước.
  - Phân phát thông điệp theo thứ tự hay không? Chỉ có thể coi là tuần tự nếu thứ tự bên gửi và bên nhận giống nhau.
  - Truyền thông điệp được thực hiện qua cơ chế Unicast hay Multicast

## THỎA THUẬN TRONG CÁC HỆ THỐNG LỖI

- Vấn đề trở nên phức tạp nếu muốn các tiến trình trong nhóm đạt được thỏa thuận trong một số trường hợp: bầu cử, COMMIT, phân chia nhiệm vụ xử lý...
- Mục đích chung của thuật toán thỏa thuận phân tán là để tất cả các tiến trình lỗi đạt được sự đồng thuận trên một số vấn đề và thiết lập sự đồng thuận đó trong một số bước hữu hạn, tuy nhiên thực tế rất phức tạp và cần phải phân biệt các trường hợp sau:
  - Các hệ thống đồng bộ đối với các hệ thống không đồng bộ
  - Có ràng buộc độ trễ truyền thông hay không, chỉ có thể ràng buộc nếu thông điệp được phân phát trong một khoảng thời gian xác định trước.
  - Phân phát thông điệp theo thứ tự hay không? Chỉ có thể coi là tuân tự nếu thứ tự bên gửi và bên nhận giống nhau.
  - Truyền thông điệp được thực hiện qua cơ chế Unicast hay Multicast

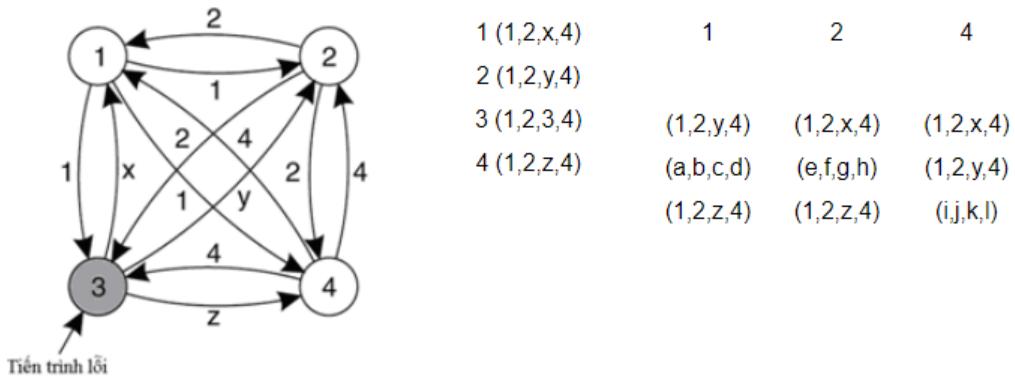
CÁC TÌNH HUỐNG CÓ THỂ ĐẠT ĐƯỢC  
THỎA THUẬN PHÂN TÁN

		Thứ tự thông điệp					
		Không		Có			
Hành vi tiến trình xử lý	Đồng bộ			X		Ràng buộc	Độ trễ
	Không đồng bộ	X	X	X	X	Không ràng buộc	không truyền thông
				X	X	không ràng buộc	
		Unicast	Multicast	Unicast	Multicast		
	Loại truyền thông điệp						

VẤN ĐỀ THỎA THUẬN Byzantine

- Hệ thống gồm N tiến trình, mỗi tiến trình i cung cấp giá trị  $V_i$  cho các tiến trình khác
- Mục tiêu cần đạt: Cho phép mỗi tiến trình xây dựng vector  $V[N]$  với  $V[i]=V_i$  nếu tiến trình i không lỗi và  $V[i]$  không xác định nếu tiến trình i bị lỗi.
- Bước 1: Mỗi tiến trình i không lỗi gửi giá trị  $V_i$  cho các tiến trình khác, tiến trình lỗi gửi giá trị bất kỳ
- Bước 2: Kết quả nhận được từ bước 1 sẽ tập hợp lại thành vector
- Bước 3: Mỗi tiến trình gửi Vector bước 2 cho các tiến trình khác
- Bước 4: Mỗi tiến trình kiểm tra phần tử thứ I trong các Vector nhận được ở bước 3, nếu kết quả kiểm tra chiếm đa số thì đặt giá trị vào Vector kết quả thỏa thuận, nếu không thì đặt giá trị UNKNOWN.

Ví dụ, hệ thống gồm bốn thành viên như trên hình 7.4 (a), các tiến trình 1, 2, 4 đều gửi giá trị tương ứng 1, 2 và 4 cho các tiến trình khác, tiến trình số 3 bị lỗi nên gửi các giá trị bất kỳ x, y và z cho các tiến trình 1, 2 và 4. Các thành viên khác trong hệ thống cần phải đảm bảo sự đồng thuận về việc tiến trình 3 có thực sự bị lỗi hay không. Kết quả thực hiện ở bước 2, mỗi tiến trình sẽ lưu giữ giá trị vector lần lượt là  $(1,2,x,4)$ ,  $(1,2,y,4)$ ,  $(1,2,3,4)$ ,  $(1,2,z,4)$ . Thực hiện bước 3, các tiến trình 1, 2 và 4 gửi giá trị vector đã nhận được ở bước 2 cho các tiến trình khác, trong khi đó tiến trình 3 tiếp tục gửi những giá trị không xác định, kết quả là các tiến trình 1, 2 và 4 sẽ nhận được các vector như hình 7.4 (c).



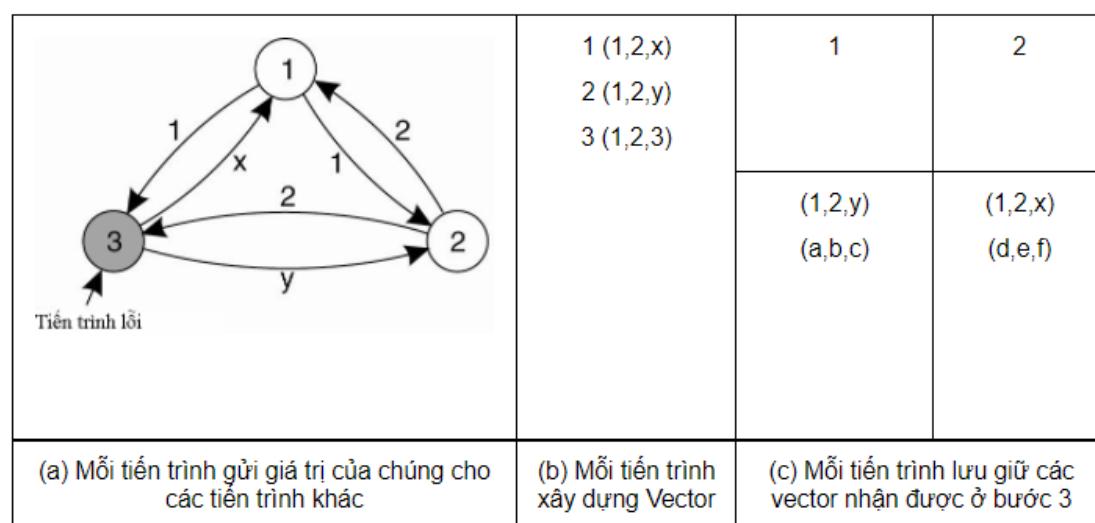
(a) Mỗi tiến trình gửi giá trị của chúng cho các tiến trình khác

(b) Mỗi tiến trình xây dựng Vector

(c) Mỗi tiến trình lưu giữ các vector nhận được ở bước 3

Hình 7.4 Thuật toán đồng thuận phân tán

Tại bước 4, mỗi tiến trình sẽ tự tổng hợp cho kết quả cuối cùng cho mỗi phần tử thứ  $i$  của các vector nhận được trong bước thứ ba, nếu giá trị nào chiếm đa số thì sẽ đánh dấu giá trị đó trong phần tử tương ứng, ngược lại sẽ điền giá trị không xác định. Giá trị cuối cùng trong các tiến trình 1, 2 và 4 đều có giá trị là  $(1,2,\text{Unknown},4)$ , như vậy các tiến trình 1, 2 và 4 có thể đảm bảo chắc chắn các tiến trình số 1, 2, 4 không bị lỗi nhưng lại không thể đảm bảo tiến trình số 3 có bị lỗi hay không.



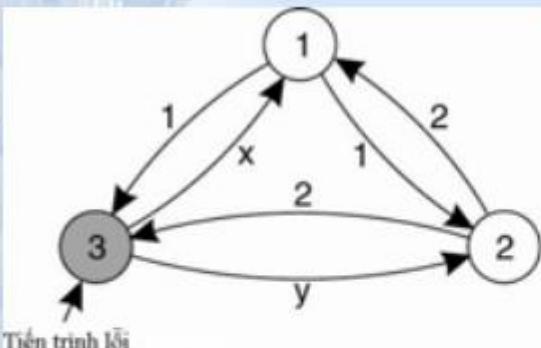
(a) Mỗi tiến trình gửi giá trị của chúng cho các tiến trình khác

(b) Mỗi tiến trình xây dựng Vector

(c) Mỗi tiến trình lưu giữ các vector nhận được ở bước 3

Hình 7.5 Không xác định được sự đồng thuận

## VĂN ĐỀ THỎA THUẬN Byzantine



Tiến trình	Vector
1	(1,2,x)
2	(1,2,y)
3	(1,2,3)

Tiến trình 1	Tiến trình 2
(1,2,y)	(1,2,x)
(a,b,c)	(d,e,f)

**Không có tiến trình chiếm đa số, thất bại trong thỏa thuận!**

Tiến trình 1	Tiến trình 2
(Unknown,Unknown,Unknown)	(Unknown,Unknown,Unknown)

### Phát hiện lỗi

Phát hiện lỗi là một trong những nhiệm vụ nền tảng để đảm bảo khả năng chịu lỗi trong các hệ thống phân tán, các tiến trình trong nhòm phải có khả năng phát hiện kịp thời những tiến trình nào đang bị lỗi hoặc có khả năng gây ra lỗi, từ đó thông báo đến các thành phần khác có liên quan hoặc thông báo cho người quản trị hệ thống. Việc phát hiện lỗi có thể dựa trên các thông tin trạng thái, ngưỡng tối hạn hoặc các tiến trình sử dụng kỹ thuật trí tuệ nhân tạo, khi có lỗi xảy ra cần phải thực hiện các thao tác trong qui trình sửa lỗi. Để phát hiện lỗi của các tiến trình, chỉ cần sử dụng hai cơ chế chủ động hoặc thụ động, trong cơ chế chủ động các tiến trình trong nhóm gửi cho nhau **thông điệp “IS ALIVE”** và tất nhiên sẽ chờ đợi phản hồi từ các tiến trình khác. Với cơ chế thụ động, mỗi tiến trình sẽ chờ nhận thông điệp từ các tiến trình khác, trong thực tế thường dùng cơ chế chủ động. Đã có nhiều công trình nghiên cứu lý thuyết phát hiện lỗi, tựu trung đều sử dụng cơ chế thời gian quá hạn để kiểm tra xem một tiến trình có bị lỗi hay không. Cơ chế này có hai điểm yếu cơ bản, thứ nhất mạng không đáng tin cậy và thứ hai thời gian quá hạn chỉ là dữ liệu thô. Khi gửi thông điệp đến tiến trình khác, không thể vội vàng kết luận tiến trình đó đã bị lỗi chỉ vì lý do không nhận được phản hồi. Mỗi tiến trình cần thường xuyên trao đổi thông tin với các tiến trình có liên quan, phải phân biệt được lỗi mạng hay lỗi trên mỗi nút mạng, bất kỳ thành viên nào phát hiện được lỗi thì phải thông báo tới các nút có liên quan.

### Truyền thông khách chủ tin cậy

Trong nhiều trường hợp tính chịu lỗi trong hệ thống phân tán tập trung vào các tiến trình lỗi, tuy nhiên cũng cần phải chú ý đến các lỗi truyền thông. Lỗi truyền thông có thể do mất kênh truyền, thất lạc thông tin, quá thời gian, lặp thông điệp..., trong thực tế khi xây dựng các kênh truyền thông tin cậy tiêu điểm tập trung vào việc che

giáu lỗi mất kênh truyền và thất lạc thông tin. Để khắc phục lỗi truyền thông, người ta thường sử dụng phương pháp truyền thông tin điểm-điểm hoặc truyền thông theo nhóm, truyền thông tin cậy điểm-điểm dựa trên các giao thức truyền tin cậy, ví dụ sử dụng giao thức TCP.

### Truyền thông điểm – điểm

Trong hệ phân tán, truyền thông điểm - điểm tin cậy được thiết lập bằng cách sử dụng các giao thức truyền thông tầng vận tải tin cậy, ví dụ giao thức TCP. Giao thức TCP che giấu được lỗi bở sót bằng cơ chế số tuần tự của đoạn tin và thực hiện truyền lại, những lỗi như vậy hoàn toàn trong suốt đối với máy khách. Tuy nhiên, sử dụng giao thức truyền thông tin cậy không khắc phục được lỗi sụp đổ, lỗi này xuất hiện khi liên kết TCP đột ngột bị ngắt và các thông điệp không thể tiếp tục gửi qua kênh truyền, thông thường máy khách sẽ nhận được thông báo lỗi kênh truyền. Khi hệ thống gặp lỗi sụp đổ thì liên kết TCP sẽ bị hủy bỏ, cách duy nhất là hệ thống tự động tạo một liên kết mới bằng cách gửi yêu cầu thiết lập lại liên kết.

#### Các tình huống lỗi khi gọi thủ tục từ xa

Gọi thủ tục từ xa là một trong những phương pháp sử dụng phổ biến trong mô hình truyền thông khách/chủ, nó che giấu quá trình truyền thông bằng cách làm cho việc gọi thủ tục từ xa giống như gọi trên máy cục bộ. Nếu máy khách, máy chủ và môi trường mạng vận hành bình thường thì gọi thủ tục từ xa thực hiện rất tốt vai trò của nó, tuy nhiên vẫn đề sẽ phát sinh khi một trong ba thành phần trên gặp lỗi và tất nhiên việc che giấu lỗi sẽ không thể thực hiện được nữa, gọi thủ tục từ xa gồm nhiều công đoạn và có thể xảy ra các lỗi sau:



Hình 7.6 Những khả năng lỗi khi gọi thủ tục từ xa

- Máy khách không thể xác định được máy chủ: Nguyên nhân gây lỗi có thể do máy chủ và máy khách dùng các phiên bản khác nhau hoặc do chính máy chủ bị lỗi. Khắc phục bằng cách sử dụng tính năng phát hiện lỗi trong ngôn ngữ lập trình ứng dụng, tuy nhiên không phải ngôn ngữ nào cũng hỗ trợ phát hiện lỗi, nếu tự viết một ngoại lệ hay điều khiển tín hiệu thì sẽ phá hủy tính trong suốt.
- Mất thông điệp yêu cầu từ máy khách gửi đến máy chủ: Đây là loại lỗi dễ xử lý nhất, hệ điều hành hay Stub của máy khách kích hoạt một bộ đếm thời gian khi gửi đi một yêu cầu, nếu bộ đếm thời gian đã trở về giá trị 0 mà vẫn không nhận được thông điệp phản hồi từ máy chủ thì nó sẽ gửi lại yêu cầu đó. Nếu máy khách nhận thấy có quá nhiều yêu cầu phải gửi lại thì nó sẽ xác nhận rằng máy chủ không hoạt động và sẽ quay lại thành kiểu lỗi "không xác định được máy chủ".

- Máy chủ bị lỗi: sau khi nhận được yêu cầu từ máy khách, máy chủ sẽ thực hiện yêu cầu và trả kết quả thực hiện cho máy khách, như vậy có thể xảy ra ba tình huống sau:
  - Máy chủ đã nhận được yêu cầu nhưng chưa thực hiện thì gặp lỗi
  - Máy chủ đang thực hiện yêu cầu thì gặp lỗi
  - Máy chủ thực hiện xong yêu cầu nhưng chưa gửi kết quả về cho máy khách thì gặp lỗi.

Khi gặp lỗi kiểu này, ở phía máy chủ sẽ thực hiện theo ba cách sau:

Cách 1: đợi đến khi nào máy chủ hoạt động trở lại, nó sẽ cố thực hiện yêu cầu đã nhận được trước khi lỗi đó, như thế RPC thực hiện ít nhất một lần.

Cách 2: máy chủ sau khi được khôi phục nó sẽ không thực hiện yêu cầu nhận được trước khi bị lỗi mà sẽ gửi lại thông báo hỏng cho máy khách, với cách này thì RPC thực hiện nhiều nhất một lần.

Cách 3: không thực hiện gì để đảm bảo cả, nếu máy chủ bị lỗi thì máy khách không hề hay biết, như vậy RPC có thể được thực hiện nhiều lần cũng có thể không thực hiện lần nào.

Máy khách có thể thực hiện theo 4 cách sau:

Cách 1: Máy khách không thực hiện gửi lại các yêu cầu vì thế không biết bao giờ yêu cầu đó mới thực hiện được hoặc có thể không bao giờ được thực hiện.

Cách 2: Máy khách liên tục gửi lại yêu cầu, như vậy có thể dẫn tới trường hợp một yêu cầu được thực hiện nhiều lần.

Cách 3: Máy khách chỉ gửi lại yêu cầu nào đó khi không nhận được thông điệp xác nhận phản hồi từ máy chủ thông báo đã nhận thành công. Trường hợp này, máy khách dùng bộ đếm thời gian, sau một khoảng thời gian xác định trước mà không nhận được xác nhận thì sẽ gửi lại yêu cầu đó.

Cách 4: Máy khách gửi lại yêu cầu nếu nhận được thông báo lỗi từ máy chủ.

Với 3 chiến lược trên máy chủ và 4 chiến lược của máy khách sẽ cho 12 khả năng xảy ra, trong đó không có một chiến lược riêng rẽ máy khách và máy chủ cho kết quả tốt, như vậy cần thiết phải phối hợp chiến lược giữa máy khách và máy chủ.

Máy khách	Máy chủ		
	Thực hiện lại	Không thực hiện, gửi thông báo lỗi	Không thực hiện
Không gửi lại	Tốt	Không thực hiện	Không thực hiện
Liên tục gửi lại	Lặp	Tốt	Tốt
Gửi lại nếu không nhận được xác nhận	Lặp	Tốt	Tốt
Gửi lại nếu nhận được thông báo lỗi	Không xảy ra	Tốt	Không xảy ra

Giả sử ký hiệu M là sự kiện gửi thông điệp trả về cho máy khách, P là sự kiện thực thi công việc trong gọi thủ tục từ xa và C là sự kiện lỗi xảy ra trên máy chủ. Khi nhận được yêu cầu từ máy khách, máy chủ có thể thực hiện trước sau đó mới thông báo cho máy khách và ký hiệu là  $P \rightarrow M$  hoặc thông báo trước kết quả sau đó mới thực hiện và ký hiệu là  $M \rightarrow P$ , với ba sự kiện trên sẽ có sáu trường hợp xảy ra như sau:

MPC, PMC: Lỗi xảy ra sau khi đã hoàn thành gửi thông điệp cho máy khách và đã hoàn thành công việc xử lý.

MC(P): Lỗi xảy ra sau khi đã hoàn thành gửi thông điệp cho máy khách nhưng chưa hoàn thành công việc xử lý.

PC(M): Lỗi xảy ra khi đã hoàn thành công việc xử lý nhưng chưa kịp gửi thông điệp cho máy khách.

C(MP), C(PM): Lỗi xảy ra khi chưa thực hiện xử lý và cũng chưa gửi thông điệp cho máy khách

Máy khách	Máy chủ					
	M → P			P → M		
	MPC	MC(P)	C(MP)	PMC	PC(M)	C(PM)
Không gửi lại	Tốt	Không thực hiện	Không thực hiện	Tốt	Tốt	Không thực hiện
Liên tục gửi lại	Lặp	Tốt	Tốt	Lặp	Lặp	Tốt
Gửi lại nếu không nhận được xác nhận	Lặp	Tốt	Không thực hiện	Lặp	Tốt	Không thực hiện
Gửi lại nếu nhận được thông báo lỗi	Tốt	Không thực hiện	Tốt	Tốt	Lặp	Tốt

- Mất thông điệp phản hồi từ máy chủ gửi kết quả cho máy khách: Máy khách đánh số tuần tự cho các yêu cầu, máy chủ sẽ nhận ra được đâu là yêu cầu đã được gửi lại nhờ các số tuần tự này, do đó máy chủ sẽ không thực hiện lặp lại các yêu cầu. Tuy nhiên máy chủ vẫn phải gửi trả về thông điệp thông báo yêu cầu nào bị thất lạc hoặc có thể sử dụng một bit ở phần thông tin điều khiển của yêu cầu để phân biệt yêu cầu nào là yêu cầu đã được gửi lại.
- Máy khách bị lỗi ngay sau khi gửi yêu cầu tới máy chủ: Máy khách gửi yêu cầu tới máy chủ rồi bị lỗi trước khi nhận được trả lời từ máy chủ gửi về, công việc mà máy chủ thực hiện nhưng không có đích nào đợi để nhận kết quả. Như vậy sẽ gây lãng phí thời gian xử lý của CPU, trường hợp này có thể giải quyết bằng bốn cách sau:
  - Cách 1: trước khi gửi đi yêu cầu nào đó, stub của máy khách sẽ tạo ra một bản ghi xác định công việc cần thực hiện này và lưu nhật ký, sau khi phục hồi máy khách sẽ lấy lại bản ghi đó và và việc thực hiện công việc đã bị tạm dừng. Phương pháp này có nhược điểm về chi phí trong việc lưu lại mỗi bản ghi cho mỗi lời gọi thủ tục từ xa.
  - Cách 2: chia thời gian hoạt động liên tục của máy trạm thành các số liên tục gọi là các thời kì. Mỗi khi các máy khách được phục hồi thì tham số đó được tăng thêm một đơn vị, máy trạm sẽ gửi thông báo đến tất cả các máy khác thông báo số thời kì mới của mình.
  - Cách 3: khi nhận được thông điệp thông báo thời kì mới, mỗi máy chủ sẽ kiểm tra xem mình có đang thực hiện một tính toán từ xa nào hay không, nếu có sẽ cố xác định xem máy khách nào đã gửi yêu cầu này, nếu không xác định được thì quá trình tính toán này sẽ bị hủy bỏ.
  - Cách 4: quy định mỗi RPC chỉ có một khoảng thời gian xác định T để thực hiện, sau khi gặp lỗi, máy khách sẽ đợi thêm một khoảng thời gian

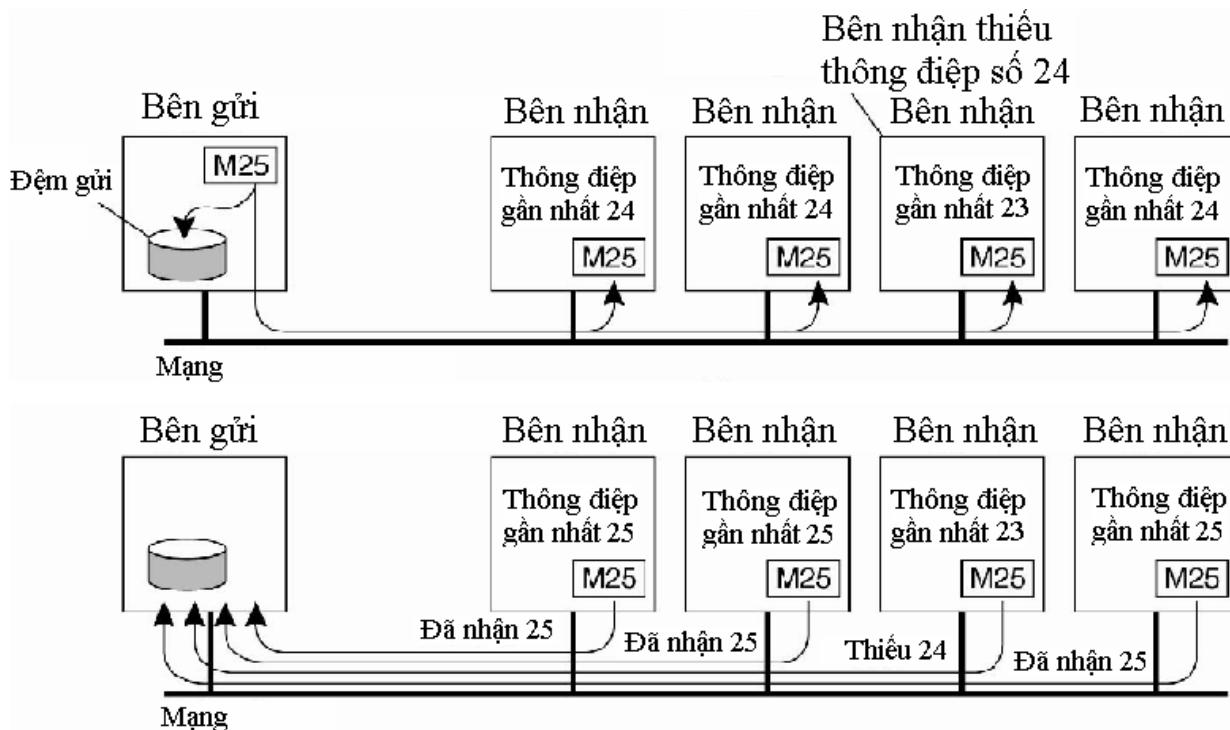
T trước khi thực hiện lại, vẫn để đặt ra là phải lựa chọn giá trị khoảng thời gian T như thế nào cho hợp lý.

## Truyền thông nhóm tin cậy

Truyền thông nhóm tin cậy là phải có cơ chế để đảm bảo thông điệp đó đến được tất cả các thành viên trong nhóm. Nếu xảy ra lỗi thì sử dụng số tuần tự các thông điệp để khắc phục, các thông điệp được lưu tại một vùng đệm của bên gửi cho đến khi nhận được bản tin xác nhận từ tất cả các thành viên trong nhóm. Nếu bên nhận xác định là bị mất một thông điệp nào đó thì nó sẽ yêu cầu gửi lại. Thông thường, bên gửi sẽ tự động gửi lại thông điệp sau trong khoảng thời gian xác định trước nếu không nhận được thông điệp xác nhận.

### 7.2.4.1. Truyền thông nhóm tin cậy cơ bản

Trong truyền thông nhóm tin cậy cơ bản, quá trình truyền tin được thực hiện theo cơ chế truyền thông, nghĩa là bên gửi sẽ chuyển thông điệp đến tất cả các thành viên trong nhóm và từng thành viên phải có trách nhiệm phản hồi kết quả đã nhận được thành công hay không, nếu có lỗi xảy ra thì áp dụng cơ chế sửa lỗi như truyền thông điểm – điểm.



Hình 7.5 Truyền thông nhóm tin cậy cơ bản

Như vậy, nếu một tiến trình nào đó trong nhóm yêu cầu gửi lại thì tất cả các thành viên khác đều nhận lại thông điệp mặc dù trước đó đã nhận thành công, hơn nữa bên gửi sẽ phải tiếp nhận phản hồi của tất cả các tiến trình trong nhóm.

### 7.2.4.2. Truyền tin nhóm tin cậy trong các hệ thống lớn

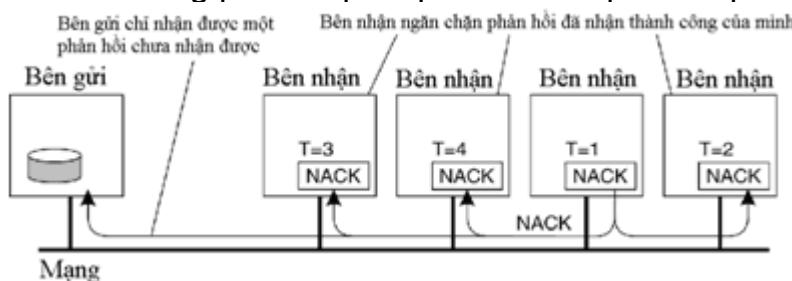
Để tăng hiệu năng khi làm việc với nhóm có số lượng lớn các tiến trình, mô hình truyền thông nhóm tin cậy mở rộng tương tự như nguyên lý làm việc của giao thức SRM (Scalable Reliable Multicasting): không trả về thông điệp xác nhận thành công

mà chỉ trả về thông điệp NACK thông báo khi có lỗi truyền. Để có thể thực hiện truyền thông nhóm tin cậy cho một số lượng lớn các tiến trình thì tổ chức các nhóm theo cấu trúc dạng cây, gốc là nhóm chứa tiến trình gửi và các nút là các nhóm có chứa tiến trình nhận. Việc chia thành các nhóm nhỏ hơn cho phép sử dụng các kịch bản truyền thông nhóm tin cậy cho từng nhóm, mỗi nhóm nhỏ sẽ đề cử một tiến trình đóng vai trò điều phối. Tiến trình điều phối có khả năng điều khiển việc truyền lại khi nhận được thông báo truyền lỗi. Tiến trình điều phối của mỗi nhóm sẽ có bộ đệm riêng, nếu không nhận được thông điệp thì nó sẽ gửi yêu cầu truyền lại tới tiến trình điều phối cấp cao hơn.

Trong qui định của truyền số liệu tin cậy sử dụng thông điệp xác nhận, khi tiến trình điều phối nhận thành công một thông điệp nó sẽ gửi thông điệp xác nhận tới tiến trình điều phối cấp cao hơn. Nếu tiến trình điều phối của một nhóm nhận được thông điệp xác nhận thành công việc chuyển thông điệp của tất cả các tiến trình trong nhóm gửi về thì nó sẽ xóa thông điệp khỏi bộ đệm của nó. Phương pháp phân cấp này sinh vấn đề xây dựng cấu trúc cây, rất nhiều trường hợp yêu cầu cây phải có cấu trúc động nên phải có một cơ chế tìm đường cho cây.

Khi một tiến trình muốn gửi thông điệp cho một tập các tiến trình khác theo dạng nhóm, nó sẽ không gửi thông điệp tới tất cả các tiến trình của nhóm mà chỉ gửi đến một nhóm nhỏ các tiến trình cần nhận thông điệp đó. Vấn đề đặt ra là phải đảm bảo gửi được thông điệp tới tất cả các tiến trình trong nhóm hoặc không được gửi tới bất kì tiến trình nào nếu một tiến trình trong nhóm bị lỗi.

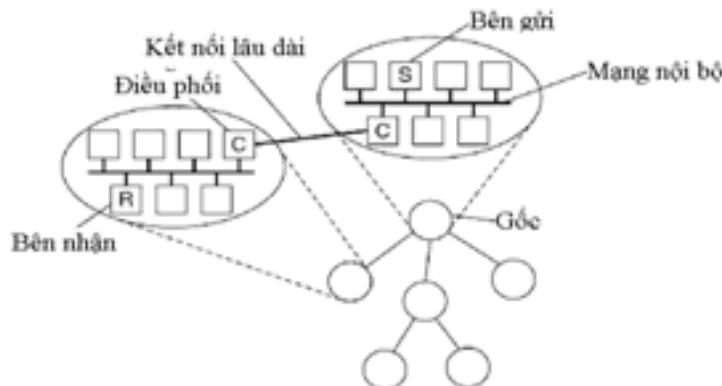
Truyền tin nhóm tin cậy cơ bản sẽ có hiệu năng thấp đối với các hệ thống lớn, giả sử phải gửi thông điệp đến  $N$  thành viên, khi đó bên gửi sẽ nhận được ít nhất  $N$  phản hồi từ các thành viên trong nhóm. Một giải pháp cải tiến có thể thực hiện bằng cách bên nhận không cần phải gửi phản hồi cho tất cả các thông điệp mà chỉ thông báo những thông điệp còn thiếu, do đó bên gửi phải lưu toàn bộ các thông điệp đã gửi. Để giảm thiểu số lượng phản hồi từ các thành viên nhận người ta đã áp dụng hai giải pháp: phản hồi không phân cấp và phản hồi có phân cấp.



Hình 7.6 Phản hồi không phân cấp

Giải pháp phản hồi không phân cấp giảm số lượng phản hồi đến bên gửi bằng cách sử dụng giao thức SRM (Scalable Reliable Multicasting) do Floyd đề xuất năm 1997. Bên nhận không phản hồi thông điệp ACK để xác nhận đã nhận thành công, nếu phát hiện thiếu thông điệp phản hồi cho bên gửi và toàn bộ thành viên trong nhóm thông điệp NACK, nghĩa là chưa nhận được thông điệp hoặc thông điệp bị lỗi. Để giảm hơn nữa số lượng phản hồi NACK, giao thức vận dụng phương pháp tự triệt tiêu thông điệp NAC. Ví dụ, nếu một thành viên dự định phản hồi NACK nó sẽ chờ một thời gian nhất định, nếu không nhận được NACK tương ứng với thông điệp đó thì mới gửi các thành viên khác trong nhóm và bên gửi. Giao thức này đảm bảo chỉ phải gửi lại 01 thông điệp bị mất phụ thuộc vào việc lập lịch thông điệp phản hồi

tại mỗi trạm nhận, nếu không cùng một thời điểm sẽ có nhiều trạm nhận gửi thông điệp NACK. Việc thiết lập thời gian trên toàn nhóm là điều không dễ dàng, bắt buộc mọi thành viên trong nhóm đều phải nhận thông điệp NACK ngay cả khi không cần thiết. Để khắc phục vấn đề này, có thể tạo thêm một nhóm mới phục vụ cho thông điệp NACK, điều này rất khó quản lý trong mạng qui mô lớn. Giải pháp cải tiến giao thức SRM cũng mang lại hiệu quả lớn, các thành viên trong nhóm hỗ trợ nhau phục hồi những thông điệp bị mất trước khi chuyển thông điệp NACK cho bên gửi.



Hình 7.7 Điều khiển phản hồi phân cấp

Thay vì truyền thông cậy cho một nhóm lớn các tiến trình, điều khiển phản hồi phân cấp được tổ chức lại các nhóm nhỏ hơn theo cấu trúc hình cây, trong đó gốc là nhóm chứa tiến trình gửi và các nút là các nhóm có chứa tiến trình nhận. Mỗi nhóm bao gồm tiến trình điều phối có nhiệm vụ xử lý các yêu cầu truyền lại. Tiến trình điều phối của mỗi nhóm sẽ có bộ đệm riêng, nếu không nhận được thông điệp thì sẽ gửi yêu cầu truyền lại tới tiến trình điều phối của nút cha. Đối với các thành viên trong nhóm có thể sử dụng kịch bản truyền nhôm cơ bản hoặc phản hồi không phân cấp. Việc xây dựng cấu trúc cây khá phức tạp, nhiều trường hợp yêu cầu cây phải có cấu trúc động nên phải có một cơ chế tìm đường.

### 7.3. Cam kết phân tán

Một yêu cầu quan trọng của giao tác phân tán là phải đảm bảo tính nguyên tử, nghĩa là các lệnh trong giao tác phải được thực thi thành công trên tất cả các thành viên trong nhóm, nếu có bất kỳ thành viên nào thực hiện không thành công thì các thành viên khác cũng phải hủy bỏ giao tác, đặc tính này có thể đạt được bằng cách cài đặt giao thức khẳng định nguyên tử (ACP – Atomic Commit Protocol). Việc cài đặt giao thức ACP có thể thực hiện một pha, hai pha ...., số pha thực hiện càng nhiều thì tính chính xác càng cao nhưng đổi lại hiệu năng của hệ thống có thể bị suy giảm.

#### 7.3.1. Giao thức khẳng định một pha

Giao thức khẳng định một pha giảm độ phức tạp của cả thông điệp lẫn nhật ký sự kiện dựa trên giả thiết tất cả các thành viên đều thực hiện thành công giao tác. Giao thức khẳng định một pha thực hiện bao gồm không tường minh và nhật ký điều phối coi mỗi hoạt động của giao tác đều được xác nhận trong mỗi thành viên. Thao tác xác nhận trong các giao thức này không chỉ có ý nghĩa thông báo giao tác đã bảo toàn sự cách ly và các thuộc tính không phân tầng mà còn cho biết giao tác không vi phạm bất kỳ ràng buộc nhất quán nào của mỗi bên tham gia.

Mô hình thiết lập cam kết phải là mô hình phân cấp và tiến trình điều phối đảm nhận nhiệm vụ cam kết. Trong cam kết một pha, tiến trình điều phối thông báo với tất cả các thành viên còn lại hoặc là thực hiện hoặc là hủy một thao tác nào đó. Nếu thành viên nào đó không thực hiện được cũng không thể báo lại cho tiến trình điều phối biết, do đó không nên áp dụng cho các hệ thống đòi hỏi tính nhất quán cao.

### 7.3.2. Giao thức khẳng định hai pha

Xét một giao tác phân tán với các thành viên là một tập các tiến trình chạy ở một máy khác và không có lỗi xảy ra, giao thức khẳng định hai pha gồm pha biểu quyết và pha quyết định. Pha biểu quyết thực hiện hai bước, tiến trình điều phối gửi một thông điệp câu biểu quyết VOTE\_REQUEST tới tất cả các thành viên trong nhóm. Sau khi nhận được thông điệp VOTE\_REQUEST, nếu có thể thực hiện được thì thành viên đó sẽ gửi lại cho tiến trình điều phối thông báo chấp nhận biểu quyết VOTE\_COMMIT, nếu không sẽ gửi lại thông báo từ chối VOTE\_ABORT. Pha quyết định cũng thực hiện hai bước, tiến trình điều phối tập hợp tất cả các biểu quyết của các thành viên, nếu tất cả đều chấp nhận giao dịch thì nó sẽ gửi một thông điệp GLOBAL\_COMMIT tới tất cả các thành viên, chỉ cần một thành viên gửi thông báo từ chối thì tiến trình điều phối quyết định hủy giao dịch trên và sẽ gửi một thông điệp GLOBAL\_ABORT cho tất cả các thành viên trong nhóm.

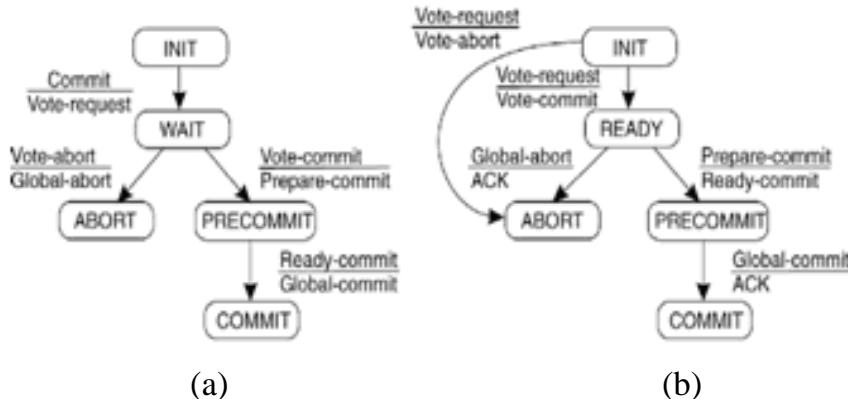


Hình 7.8 Giao thức khẳng định hai pha

Các thành viên sau khi đã gửi thông báo chấp nhận cam kết sẽ đợi phản hồi từ tiến trình điều phối, nếu nhận được thông báo GLOBAL\_COMMIT thì giao tác sẽ được chấp thuận, còn nếu nhận được GLOBAL\_ABORT thì giao tác sẽ bị hủy bỏ. Giao thức khẳng định hai pha đảm bảo tính nguyên tử và phục hồi độc lập nhưng chi phí đáng kể khi thực hiện giao tác bình thường, điều đó ảnh hưởng bất lợi tới hiệu năng của hệ thống. Nguyên nhân là do chi phí phải trả vì sự phức tạp trong trao đổi thông điệp, một lượng lớn thông điệp dùng cho việc điều phối thao tác trên các nút khác nhau và sự phức tạp của nhật ký ghi lại các thao tác trên từng nút.

### 7.3.3. Giao thức khẳng định ba pha

Nhược điểm chính của giao thức khẳng định hai pha là tốn nhiều thời gian chờ đợi, cả tiến trình điều phối lẫn các thành viên còn lại đều phải chờ một thông điệp nào đó được gửi đến, nếu tiến trình điều phối bị lỗi thì hoạt động của cả hệ thống sẽ bị ảnh hưởng. Giao thức khẳng định hai pha vận hành theo nguyên lý phong tỏa, nếu tiến trình điều phối bị lỗi thì các thành viên không thể đạt được quyết định cuối cùng. Để khắc phục nhược điểm của cam kết hai pha trong trường hợp tiến trình điều phối bị lỗi, Skeen đã đề xuất giao thức cam kết ba pha, các trạng thái trong cam kết ba pha khá giống cam kết hai pha nhưng thêm một trạng thái tiền khẳng định PRECOMMIT. Mặc dù giao thức khẳng định ba pha được đề cập đến rất nhiều trong các tài liệu nghiên cứu nhưng nó lại ít được sử dụng trong thực tế bởi vì hiện tượng phong tỏa trong giao thức cam kết hai pha rất ít khi xảy ra.



Hình 7.11 Giao thức khẳng định ba pha

- (a) Máy trạng thái hữu hạn của thành phần điều phối
- (b) Máy trạng thái hữu hạn của thành viên

Giống như giao thức khẳng định hai pha, giao thức khẳng định ba pha cũng gồm nhiều tiến trình trong đó có một tiến trình đóng vai trò điều phối, máy trạng thái hữu hạn của các tiến trình thể hiện trên hình 7.11. Bản chất của giao thức khẳng định ba pha thể hiện ở chỗ các trạng thái của thành phần điều phối và các thành viên tham gia đáp ứng hai điều kiện sau:

1. Không có một trạng thái đơn lẻ nào mà từ đó có thể chuyển trực tiếp sang trạng thái khẳng định hoặc hủy bỏ.
2. Không có trạng thái mà trong đó không thể đưa ra quyết định cuối cùng và từ đó có thể chuyển sang trạng thái khẳng định.

Hai điều kiện trên là cần thiết và đủ để đảm bảo giao thức khẳng định ba pha có thể hoạt động trong chế độ không phong tỏa. Thành phần điều phối trong giao thức khẳng định ba pha bắt đầu bằng việc gửi thông điệp yêu cầu bỏ phiếu (Vote-Request) đến tất cả các thành viên khác, sau đó sẽ chờ phản hồi từ các thành viên này. Chỉ cần 1 tiến trình bỏ phiếu hủy bỏ (ABORT) hoặc một tiến trình thành viên nào đó không phản hồi thì quyết định cuối cùng cũng sẽ là hủy bỏ, như vậy tiến trình điều phối sẽ gửi thông điệp hủy bỏ toàn cục (Global Abort) đến tất cả các tiến trình thành viên. Nếu tất cả các tiến trình thành viên đều phản hồi phiếu bầu đồng ý khẳng định, chắc chắn tất cả các tiến trình thành viên đã ở trạng thái sẵn sàng khẳng định (READY), nghĩa là giao tác có cơ hội được khẳng định, tiến trình điều phối sẽ gửi thông điệp chuẩn bị khẳng định (Prepare commit) đến tất cả các thành viên. Nếu tại thời điểm này tiến trình điều phối bị lỗi thì sẽ không có một thông điệp nào được gửi đến các tiến trình thành viên, sau một thời gian nhất định từng tiến trình thành viên sẽ không được phép hủy bỏ hay khẳng định giao tác, mỗi thành viên áp dụng điều kiện thứ nhất nên chúng phải tham khảo trạng thái của các thành viên khác, tất cả đều ở trạng thái quá hạn chờ thông điệp chuẩn bị khẳng định, do đó giao tác sẽ tự động bị hủy. Nhận được thông điệp chuẩn bị khẳng định từ tiến trình điều phối, các tiến trình thành viên sẽ phản hồi. Sau khi đã nhận được phản hồi từ tất cả các tiến trình thành viên, tiến trình điều phối sẽ gửi thông điệp khẳng định toàn cục (Global commit), nhận được thông điệp này các tiến trình thành viên sẽ thực hiện khẳng định giao tác, như vậy giao tác phân tán đã hoàn thành trên tất cả các thành viên của hệ thống. Xảy ra tình huống một thành viên nào đó bị lỗi nên không có phản hồi, như vậy tiến trình điều phối bị phong tỏa ở trạng thái chuẩn bị khẳng định, nó biết chắc chắn tiến trình thành viên lỗi đã biết giao tác đang chờ để được khẳng định, tiến trình điều phối có thể ra lệnh một cách an

toàn cho những thành viên đang hoạt động bằng cách gửi thông điệp khẳng định toàn cục cho nhóm các tiến trình tham gia. Ngoài ra, tiến trình điều phối dựa vào giao thức phục hồi để các tiến trình lỗi có thể thực hiện khẳng định cho các giao tác khi chúng phục hồi trở lại. Nếu tiến trình thành viên ở trạng thái READY, quá thời gian qui định mà vẫn không nhận được thông điệp chuẩn bị khẳng định, nó sẽ liên lạc với tiến trình thành viên khác, nếu trạng thái của tiến trình thành viên được hỏi là ABORT thì nó sẽ hủy bỏ giao tác, nếu là PRECOMMIT thì sẽ chuyển trạng thái của nó giống như thành viên đã hỏi, nếu là COMMIT thì sẽ thực hiện khẳng định giao tác. Nếu thành viên ở trạng thái PRECOMMIT, quá thời gian qui định mà vẫn không nhận được thông điệp khẳng định toàn cục, nó sẽ liên lạc với tiến trình thành viên khác, nếu trạng thái của tiến trình thành viên được hỏi là COMMIT thì nó thực hiện khẳng định giao tác, nếu tất cả các tiến trình thành viên đều ở trạng thái PRECOMMIT thì nó cũng thực hiện khẳng định giao tác.

## 7.4. Phục hồi

Lỗi có thể xảy ra ở bất kỳ thời điểm nào, một hệ thống đáng tin cậy phải không những có khả năng phát hiện lỗi mà còn biết phục hồi sau khi gặp lỗi, nói cách khác phục hồi là các biện pháp đưa hệ thống từ trạng thái bị lỗi trở về trạng thái không lỗi.

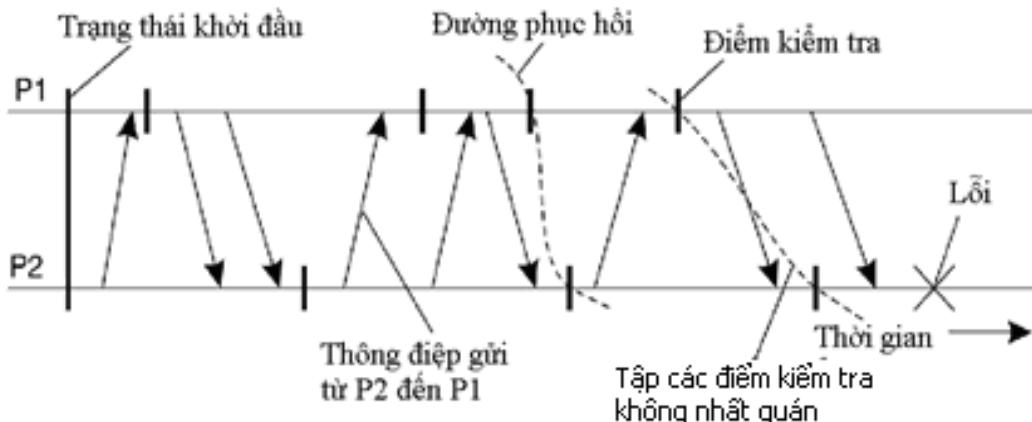
### 7.4.1. Các biện pháp phục hồi

Có hai cách tiếp cận cho phục hồi lỗi, phục hồi lùi (backward) và phục hồi tiến (forward). Tư tưởng của phương pháp phục hồi lùi là đưa hệ thống từ trạng thái lỗi ở thời điểm hiện tại về trạng thái không lỗi ở thời điểm trước khi xảy ra lỗi. Để làm được điều đó cần phải liên tục ghi lại trạng thái của hệ thống và khi hệ thống gặp lỗi sẽ chuyển về trạng thái đã sao lưu, thời điểm hệ thống thực hiện sao lưu dữ liệu và trạng thái gọi là điểm kiểm tra (checkpoint). Khác với phục hồi lùi, phục hồi tiến không đưa hệ thống trở lại trạng thái không lỗi trước khi xảy ra lỗi mà chuyển hệ thống sang trạng thái mới không lỗi để có thể tiếp tục hoạt động. Khó khăn lớn nhất của giải pháp này là phải biết trước những lỗi nào có thể xảy ra, chỉ khi đó mới có thể sửa lỗi và chuyển sang trạng thái mới. Có thể dễ dàng giải thích sự khác biệt giữa phục hồi tiến và phục hồi lùi khi xem xét cài đặt truyền thông tin cậy. Khi phát hiện thấy một đoạn tin bị lỗi, bên nhận sẽ yêu cầu bên gửi phải gửi lại đoạn tin đó, nghĩa là hệ thống đã chuyển về trạng thái trước khi gửi đoạn tin lỗi, đó là cách thể hiện của phục hồi lùi. Một giải pháp khác là sử dụng mã sửa chữa (Erasure Code), giả sử cần phải gửi k đoạn tin, bên gửi sẽ mã hóa thành n đoạn tin gọi là khối mã sửa chữa ( $n, k$ ), chỉ cần nhận đủ k đoạn tin mã hóa sẽ có thể phục hồi lại được k đoạn tin nguyên gốc, nếu chưa đủ số lượng đoạn tin thì vẫn tiếp tục phải gửi cho đến khi sử dụng lại được các đoạn tin đã thất lạc. Nói chung, kỹ thuật phục hồi lỗi lùi được sử dụng rộng rãi như một cơ chế tổng quát về phục hồi lỗi trong các hệ thống phân tán, ưu điểm của nó là tính độc lập với các tiến trình và như vậy có thể tích hợp vào tầng trung gian của hệ thống phân tán như một dịch vụ đa năng. Tuy nhiên phục hồi lùi cũng bộc lộ một số vấn đề, việc phục hồi hệ thống về trạng thái trước sẽ đòi hỏi chi phí tương đối lớn về hiệu năng, cách đơn giản để thoát khỏi vấn đề này là tạo ra cơ chế ít tốn kém bằng cách khởi động lại các thành phần. Cơ chế phục hồi lùi độc lập với các ứng dụng phân tán, không có gì đảm bảo sau khi phục hồi lỗi tương tự lại không lặp lại và như vậy có thể xảy ra vòng lặp phục hồi. Cuối cùng, dựa vào cơ chế điểm kiểm tra nhưng không phải lúc nào cũng có thể phục hồi về trạng thái trước khi xảy ra lỗi, ví dụ trường hợp các máy trả tiền tự động, sau khi máy đã nhả tiền thì khó có thể rút lại được. Thực hiện điểm kiểm tra là thao tác làm suy giảm đáng kể hiệu năng của hệ thống, do đó nhiều hệ thống phân tán có khả năng chịu lỗi kết hợp điểm kiểm tra với việc ghi nhật ký thông điệp, tiến trình gửi sẽ ghi nhật ký thông điệp trước khi gửi và tiến trình nhận sẽ ghi nhật ký thông điệp trước khi chuyển lên tầng ứng dụng để thực hiện. Nếu tiến trình nhận gặp lỗi, tiến trình

gửi chỉ việc phục hồi về trạng thái của điểm kiểm tra sau và lấy các thông điệp trong nhật ký kể từ thời điểm thực hiện điểm kiểm tra để gửi lại, như vậy có thể giãn tần suất thực hiện các điểm kiểm tra và vì vậy sẽ giảm thiểu ảnh hưởng tới hiệu năng của hệ thống. Để phục hồi về trạng thái không lỗi thì dữ liệu phải được lưu trữ an toàn, dữ liệu không bị hỏng hóc ngay cả khi tiến trình bị sụp đổ hoặc ngay cả khi phương tiện lưu trữ bị lỗi. Dữ liệu thường được lưu trữ ở bộ nhớ RAM hoặc ổ đĩa cứng, nếu lưu trữ trong RAM thì sẽ bị mất khi mất điện hoặc khi máy tính gặp lỗi. Dữ liệu trên ổ đĩa cũng có thể bị lỗi, vì vậy cần thiết phải sử dụng kỹ thuật RAID để nhân bản trên nhiều ổ đĩa, thậm chí cần phải xây dựng chính sách sao lưu dữ liệu sang các máy tính khác.

#### 7.4.2. Điểm kiểm tra

Trong hệ thống phân tán có khả năng chịu lỗi, phục hồi lùi đòi hỏi hệ thống phải đều đặn ghi lại trạng thái của nó vào vùng nhớ vĩnh cửu, đặc biệt phải ghi được trạng thái toàn cục nhất quán gọi là bản sao phân tán (snapshot). Trong bản sao phân tán, nếu một tiến trình nào đó ghi nhật ký nhận được bản tin thì chắc chắn tiến trình gửi cũng sẽ phải ghi nhật ký gửi bản tin đó, nếu có lỗi xảy ra thì các tiến trình này biết được sẽ phải bắt đầu từ đâu để phục hồi hệ thống.

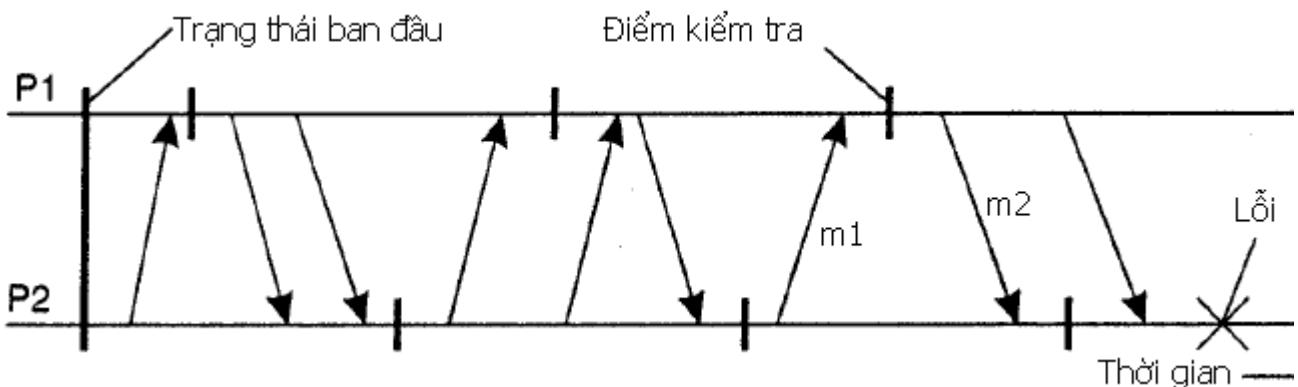


Hình 7.12 Phục hồi lùi sử dụng trang thái toàn cục nhất quán

Trong các lược đồ phục hồi lùi, mỗi tiến trình đều đặn ghi trạng thái của nó vào bộ nhớ cục bộ vĩnh cửu. Để phục hồi sau khi xảy ra lỗi, cần thiết phải dò tìm trạng thái nhất quán toàn cục từ những trạng thái cục bộ này, tốt nhất là phục hồi về bản sao phân tán gần nhất và gọi là đường phục hồi, nói cách khác đường phục hồi tương ứng với tập các điểm kiểm tra nhất quán gần nhất.

#### 7.4.2.1. Điểm kiểm tra độc lập

Việc thực hiện điểm kiểm tra trong các hệ thống phân tán thông thường mang tính chất cục bộ, mỗi tiến trình đều đặn ghi trạng thái cục bộ của theo cách riêng biệt, điều đó sẽ gây khó khăn cho việc xác định đường phục hồi. Để tìm ra đường phục hồi mỗi tiến trình sẽ phải quay lui về trạng thái gần nhất đã được ghi lại, nếu các trạng thái cục bộ này không tạo nên được đường phục hồi thì tiếp tục phải quay lui về trạng thái trước nữa, quá trình phục hồi phân tầng này có thể dẫn đến hiện tượng dây chuyền.



Hình 7.13 Hiệu ứng dây chuyền trong phục hồi lùi

Hình 7.13 cho thấy, khi tiến trình P2 gặp lỗi thì cần phải phục hồi trạng thái của nó về điểm kiểm tra gần nhất, tiến trình P1 cũng sẽ phải quay lui về trạng thái của điểm kiểm tra gần nhất. Rất tiếc hai trạng thái gần nhất được lưu cục bộ này không hình thành trạng thái toàn cục nhất quán, tiến trình P2 ghi nhận ký đã nhận được thông điệp m2 nhưng nhận ký của tiến trình P1 lại không ghi nhận đã gửi thông điệp m2. Như vậy tiến trình P2 lại phải quay lui về trạng thái trước nữa, nhưng ở trạng thái này lại không nhận đã gửi thông điệp m1 trong khi nhận ký của tiến trình P1 cho thấy đã nhận được thông điệp m1, tiến trình P1 lại quay lui về trạng thái trước nữa, cứ như vậy cả hai tiến trình phải quay lui về trạng thái ban đầu mới tìm thấy đường phục hồi.

Các tiến trình thực hiện điểm kiểm tra cục bộ độc lập với nhau gọi là phương pháp điểm kiểm tra độc lập, một giải pháp khác là phối hợp điểm kiểm tra toàn cục, nó đòi hỏi phải có sự đồng bộ trên tất cả các tiến trình và như vậy sẽ làm suy giảm hiệu năng hệ thống. Một nhược điểm khác nữa của điểm kiểm tra độc lập còn thể hiện khi dọn dẹp không gian lưu trữ cục bộ, tuy nhiên nhược điểm chính vẫn là vấn đề xác định đường phục hồi. Cài đặt điểm kiểm tra độc lập đòi hỏi phải ghi lại những phụ thuộc để các tiến trình có thể cùng nhau quay lui về trạng thái toàn cục nhất quán. Giả sử  $CP_i(m)$  là điểm kiểm tra thứ m của tiến trình  $P_i$ ,  $INT_i(m)$  là khoảng thời gian giữa hai điểm kiểm tra  $CP_i(m-1)$  và  $CP_i(m)$ . Khi tiến trình  $P_i$  gửi thông điệp trong khoảng thời gian  $INT_i(m)$ , nó gắn kèm thông điệp cặp  $(i,m)$ . Tiến trình  $P_j$  nhận được thông điệp với cặp đính kèm  $(i,m)$  trong khoảng thời gian  $INT_j(n)$ , nó ghi nhận phụ thuộc  $INT_i(m) \rightarrow INT_j(n)$ , khi thực hiện điểm kiểm tra  $CP_j(n)$  nó cũng ghi nhận sự phụ thuộc này. Khi tiến trình  $P_i$  quay lui về điểm kiểm tra  $CP_i(m-1)$  thì cần phải đảm bảo rằng tất cả các tiến trình đã nhận được thông điệp từ  $P_i$  trong khoảng thời gian  $INT_i(m)$  cũng phải quay lui về điểm kiểm tra trước khi nhận được các thông điệp đó. Như vậy tiến trình  $P_j$  sẽ phải quay lui về điểm kiểm tra  $CP_j(n-1)$ , nếu vẫn chưa đạt được trạng thái nhất quán toàn cục thì tiếp tục phải quay lui về điểm kiểm tra trước nữa. Việc tính toán đường phục hồi đòi hỏi phải phân tích những phụ thuộc đã ghi nhận trong mỗi tiến trình khi thực hiện điểm kiểm tra, không cần phải đi sâu thêm về vấn đề này cũng thấy công việc này phức tạp thế nào và điều đó cho thấy sự cần thiết phải thực hiện điểm kiểm tra phối hợp. Vì vậy kỹ thuật thực hiện điểm kiểm tra phối hợp đã được ứng dụng rộng rãi, đặc biệt đối với các hệ thống có quy mô lớn.

#### 7.4.2.2. Điểm kiểm tra phối hợp

Trong kỹ thuật điểm kiểm tra phối hợp, tất cả các tiến trình đồng bộ với nhau để cùng thực hiện việc ghi trạng thái vào bộ nhớ lưu trữ ổn định cục bộ, ưu điểm chính của nó là các trạng

thái cục bộ đã tự hình thành tính nhất quán toàn cục, như vậy sẽ tránh được hiệu ứng quay lui dây chuyền. Giải thuật ảnh phân tán được sử dụng để phối hợp thực hiện điểm kiểm tra, đó là ví dụ về việc phối hợp điểm kiểm tra không phong tỏa. giải pháp đơn giản hơn là sử dụng giao thức phong tỏa hai pha, tiến trình điều phối gửi thông điệp yêu cầu thực hiện điểm kiểm tra CHECKPOINT REQUEST đến tất cả các tiến trình, nhận được thông điệp này các tiến trình sẽ thực hiện điểm kiểm tra cục bộ và gửi thông điệp xác nhận đã hoàn thành cho tiến trình điều phối, tất cả các thông điệp nhận được sau đó hoặc những thông điệp cần gửi đều chuyển vào hàng đợi cho đến khi nhận được thông điệp hoàn thành điểm kiểm tra CHECKPOINT\_DONE từ tiến trình điều phối. Sau khi nhận được thông điệp xác nhận từ tất cả các tiến trình đã hoàn thành thực hiện điểm kiểm tra, tiến trình điều phối sẽ gửi thông điệp CHECKPOINT\_DONE đến tất cả các tiến trình thành viên để các tiến trình đó có thể tiếp tục thực hiện công việc của mình.

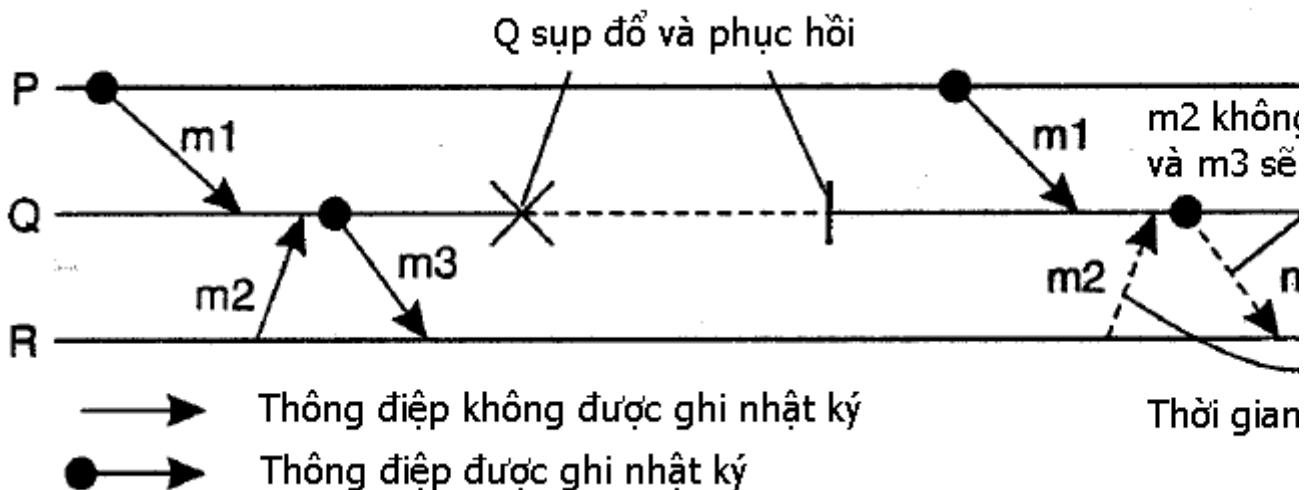
Có thể cải tiến giải thuật này bằng cách chỉ gửi thông điệp yêu cầu điểm kiểm tra đến các tiến trình phụ thuộc vào phục hồi của thành phần điều phối, bỏ qua các tiến trình khác. Một tiến trình được gọi là phụ thuộc khi nó nhận được thông điệp trực tiếp hoặc gián tiếp có quan hệ nhân quả với thông điệp đã nhận được từ tiến trình điều phối kể từ khi thực hiện điểm kiểm tra cuối cùng, điều này dẫn tới khái niệm ảnh lũy tiến. Để thực hiện ảnh lũy tiến, tiến trình điều phối chỉ gửi thông điệp CHECKPOINT\_REQUEST đến các tiến trình đã nhận được thông điệp của nó từ thời điểm thực hiện điểm kiểm tra cuối cùng, tiến trình  $P_i$  nhận được thông điệp này sẽ làm tương tự bằng cách gửi thông điệp CHECKPOINT\_REQUEST đến các tiến trình khác có liên quan đến nó, quá trình này chỉ được phép thực hiện một lần. Sau khi các tiến trình đã hoàn thành thực hiện điểm kiểm tra cục bộ, tiến trình điều phối sẽ gửi thông điệp thứ hai CHECKPOINT\_DONE để các tiến trình có liên quan tiếp tục thực hiện công việc của mình.

#### 7.4.3. Ghi nhật ký thông điệp

Thực hiện điểm kiểm tra đòi hỏi ghi trạng thái vào bộ nhớ ổn định, vì vậy cần giảm số lần thực hiện điểm kiểm tra nhưng vẫn phải đảm bảo khả năng phục hồi, một trong những kỹ thuật quan trọng được áp dụng trong hệ thống phân tán là ghi nhật ký thông điệp. Ý tưởng cơ bản của kỹ thuật này dựa trên giả thiết nếu thông điệp được gửi lại thì có thể đạt được trạng thái nhất quán toàn cục mà không cần phục hồi về trạng thái lấy từ bộ nhớ ổn định, thay vào đó trạng thái của điểm kiểm tra được lấy từ điểm khởi đầu và tất cả các thông điệp từ điểm khởi đầu đó sẽ được gửi lại và xử lý phù hợp. Cách tiếp cận này hoạt động tốt với giả thiết dựa trên mô hình xác định từng phần, việc thực thi của mỗi tiến trình là một chuỗi các khoảng sự kiện, khoảng sự kiện bắt đầu từ thời điểm xảy ra sự kiện không xác định và kết thúc tại thời điểm của sự kiện cuối cùng trước khi xảy ra sự kiện không xác định kế tiếp. Nếu thực hiện lại các khoảng sự kiện này sẽ được kết quả tương ứng với kết quả của những sự kiện không xác định, như vậy việc ghi lại tất cả các sự kiện không xác định chuyển thành có thể lặp lại lại toàn bộ những thực thi của tiến trình theo cách xác định.

Việc ghi nhật ký thông điệp là cần thiết để phục hồi tiến trình sụp đổ và như vậy sẽ phục hồi được trạng thái toàn cục nhất quán, điều quan trọng là phải biết chính xác khi nào các thông điệp cần được ghi lại. Alvisi và Marzullo đã mô tả nhiều lược đồ ghi nhật ký thông điệp, mỗi thông điệp chứa thông tin điều khiển để có thể được gửi lại và xử lý chính xác. Thông tin điều khiển xác định bên gửi và bên nhận, số tuần tự để tránh trùng lặp, số thứ tự phân phát để quyết định chính xác khi nào sẽ được xử lý ở bên nhận. Một thông điệp được coi là ổn định nếu nó không bị thất lạc trên kênh truyền vì nó sẽ được ghi vào bộ

nhớ ổn định, những thông điệp đó sẽ được sử dụng để gửi lại khi cần thực hiện phục hồi. Khi một tiến trình nào bị sụp đổ, tiến trình đó sẽ phải được phục hồi, ngay cả khi phục hồi thành công cũng có thể dẫn tới hiện tượng tiến trình mồ côi, đó là những tiến trình sống sót sau sự sụp đổ của tiến trình khác nhưng trạng thái của nó không nhất quán với tiến trình sụp đổ và đã được phục hồi, để làm rõ vấn đề này cần phải phân tích chi tiết về quá trình ghi nhật ký thông điệp.



Hình 7.14 Phát lại thông điệp không chính xác sau khi phục hồi

Ví dụ trên hình 7.14, tiến trình Q nhận thông điệp m1 từ tiến trình P và m2 từ tiến trình R, thông điệp m1 có ghi nhật ký trong khi thông điệp m2 không được ghi nhật ký, sau đó Q lại gửi thông điệp m3 cho tiến trình R và thông điệp này được ghi vào nhật ký. Sau khi gửi thông điệp m3, tiến trình Q sụp đổ do đó nó phải thực hiện quá trình phục hồi, tiến trình Q lấy thông điệp từ nhật ký và phục hồi thành công thông điệp m1 nhưng không thể phục hồi thông điệp m2 vì thông điệp này không được ghi trong nhật ký. Vì thông điệp m3 phụ thuộc nhân quả vào thông điệp m2, m2 không được phát lại thì Q cũng sẽ không thực hiện gửi thông điệp m3. Tiến trình R có bản sao thông điệp m3 nhưng thông điệp này lại không có trong tiến trình Q sau khi phục hồi và như vậy trạng thái của tiến trình Q không nhất quán với tiến trình R, khi đó R được gọi là **tiến trình mờ coi**.

Một cách tổng quát, mỗi thông điệp m dẫn tới tập  $DEP(m)$  các tiến trình phụ thuộc vào sự phân phát thông điệp m, nếu một thông điệp m' phụ thuộc nhân quả với thông điệp m và thông điệp m' được phân phát đến bất kỳ tiến trình nào thì tiến trình đó cũng sẽ là thành viên của tập  $DEP(m)$ . Thông điệp m' phụ thuộc nhân quả vào m nếu nó được gửi bởi một tiến trình trước đó đã phân phát thông điệp m, hoặc m' thuộc tiến trình đã phân phát thông điệp khác nhưng thông điệp đó phụ thuộc nhân quả vào thông điệp m. Tập  $COPY(m)$  gồm các tiến trình nhận được bản sao thông điệp m nhưng chưa kịp ghi vào vùng nhớ ổn định. Khi một tiến trình chuyển tiếp thông điệp m thì nó cũng là thành viên của  $COPY(m)$ , nếu tất cả các tiến trình này sụp đổ thì rõ ràng việc phát lại thông điệp m sẽ không thể thực hiện được. Tiến trình mò côi xuất hiện khi nó phụ thuộc vào thông điệp nhưng không có cách nào phát lại thông điệp đó. Để tránh hiện tượng tiến trình mò côi thì cần phải chắc chắn rằng mỗi tiến trình trong tập  $COPY(m)$  sụp đổ thì không ở trong tập  $DEP(m)$  của tiến trình sống sót, nghĩa là tất cả các tiến trình trong  $DEP(m)$  cũng phải bị sụp đổ. Điều kiện này có thể được thực hiện nếu đảm bảo rằng mọi tiến trình là thành viên của  $DEP(m)$  thì cũng là thành viên của  $COPY(m)$ , nói cách khác nếu một tiến trình phụ thuộc thông điệp m thì nó phải giữ bản sao của thông điệp m. Về cơ bản, có thể giải quyết vấn đề tiến trình mò côi bằng cách sử dụng giao thức ghi nhật ký bi quan hoặc lac quan. Giao thức bi quan đảm bảo rằng mỗi thông điệp chưa ổn định thì chỉ có

nhiều nhất một tiến trình phụ thuộc vào nó, như vậy tiến trình sẽ chỉ được phép gửi thông điệp khi tất cả các thông điệp đã được ghi vào bộ nhớ ổn định, nghĩa là tiến trình đã thuộc tập COPY(m). Ngược lại, giao thức ghi nhật ký lục quan, công việc thực tế được thực hiện sau khi xảy ra sự cố, tiến trình mô coi trong tập DEP(m) sẽ quay lui cho đến khi không thuộc tập này nữa, như vậy nó phải lưu vết các phụ thuộc và vì vậy sẽ làm phức tạp hóa vấn đề cài đặt. So sánh hai cách tiếp cận trên có thể thấy cách tiếp cận bi quan đơn giản hơn cách tiếp cận lục quan rất nhiều, vì vậy trong thực tế người ta thường sử dụng cách tiếp cận bi quan.

## Chương 8 : Tính nhất quán và nhân bản

### NỘI DUNG

- Truyền thông
- Định danh
- Đồng bộ
- Tiến trình trong các hệ thống phân tán
- Quản trị giao dịch và điều khiển tương tranh
- Phục hồi và chịu lỗi
- Bảo mật
- **Tính nhất quán và vấn đề nhân bản**

### Tính nhất quán và nhân bản

Giới thiệu nhân bản

Mô hình lấy dữ liệu làm trung tâm

Mô hình lấy máy khách làm trung tâm

Quản lý bản sao

Các giao thức đảm bảo tính toàn vẹn

### NHÂN BẢN DỮ LIỆU

- Nhân bản dữ liệu để tăng độ tin cậy của hệ thống
- Nhân bản để nâng cao hiệu năng
  - Qui mô số lượng
  - Qui mô phạm vi địa lý
- Vấn đề này sinh:
  - Giảm tính nhất quán của dữ liệu
  - Chi phí tăng băng thông để duy trì nhân bản
  - **Thao tác thực hiện nhân bản có thể làm giảm hiệu năng xử lý của hệ thống**

Phải tốn nhiều công sức

để xây dựng các mô hình đảm bảo tính nhất quán của dữ liệu.

## Hai cách tiếp cận

- Lấy máy khách làm chung tâm
- Lấy dl làm chung tâm

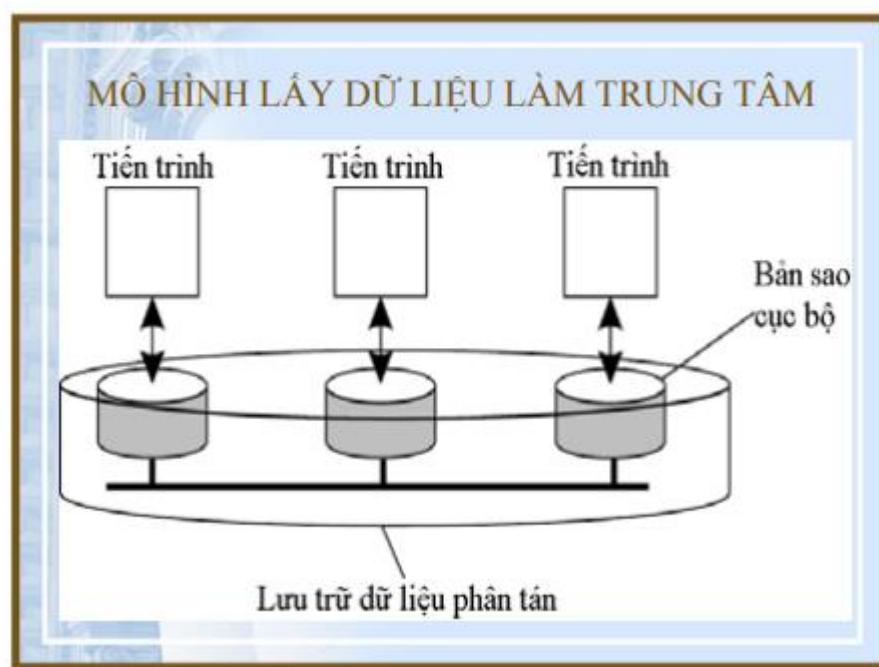
## CÁC MÔ HÌNH NHÂN BẢN DỮ LIỆU

- . Không có mô hình nhân bản dữ liệu tốt nhất. Phải thỏa hiệp giữa các tiêu chí nhân bản để đạt được mục tiêu yêu cầu.
- . Mô hình lấy dữ liệu làm trung tâm
  - Mô hình nhất quán chặt
  - Mô hình nhất quán tuần tự
  - Mô hình nhất quán tuyến tính
  - Mô hình nhất quán nhân quả
  - Mô hình nhất quán FIFO
  - Mô hình nhất quán yếu
  - Mô hình nhất quán **Mục dữ liệu**
  - Mô hình nhất quán đi vào (entry)
- . Mô hình nhất quán lấy máy khách làm trung tâm
  - Mô hình nhất quán cuối cùng
  - Mô hình nhất quán đọc đều
  - Mô hình nhất quán ghi đều
  - Mô hình nhất quán đọc kết quả ghi

Mô hình nhất quán ghi theo sau đọc

## Mô hình lấy DL làm trung tâm

- Tính nhất quán thường được hiểu trong ngữ cảnh các thao tác đọc/ghi bộ nhớ dùng chung (CSDL, tập tin...).
- Trong hệ thống phân tán, tính nhất quán được mở rộng cho trường hợp dữ liệu được lưu trữ tại các máy tính khác nhau.
- Tiến trình đọc dữ liệu từ bản sao (cục bộ) của dữ liệu hệ thống, thao tác ghi sẽ được lan truyền đến tất cả các bản sao dữ liệu trong thống.
- Khó khăn cơ bản trong việc nhân bản dữ liệu là thiếu cơ chế đồng hồ chung cho tất cả các máy tính trong hệ thống
- Hai vấn đề :
  - Tính nhất quán liên tục
  - Tính nhất quán thứ tự giao tác



### Tính nhất quán liên tục

- Nhận bản dữ liệu đưa ra vấn đề nhất quán, nhưng không có cách chung nào giải quyết hiệu quả vấn đề này. Chỉ có nói lỏng tính nhất quán mới hy vọng đạt được giải pháp hiệu quả.
- Không có qui tắc chung nào cho việc nói lỏng tính nhất quán, phụ thuộc lớn vào từng ứng dụng.
- Các cách tiếp cận nói lỏng tính nhất quán:
  - Sự chênh lệch giá trị số giữa các bản sao: Giá trị có thể là của một trường dữ liệu nhưng cũng có thể là số lượng các thao tác thay đổi.
  - Sự chênh lệch trạng thái: Thời gian cập nhật cuối cùng
  - Sự chênh lệch thứ tự các thao tác cập nhật: Khá phức tạp, nhất là đối với các thao tác có ROLLBACK
- Sự chênh lệch trong ba cách tiếp cận trên hình thành phạm vi nhất quán liên tục

## VÍ DỤ NHẤT QUÁN LIÊN TỤC

Giả thiết ban đầu  $x=0, y=0$ . Độ lệch thứ tự thao tác của A là 3, độ lệch thứ tự thao tác của B là 2

Bản sao A

### Đơn vị nhất quán

$$x = 6; y = 3$$

#### Thao tác

$< 5, B>$

#### Kết quả

$[x = 2]$

$< 8, A>$

$[y = 2]$

$< 12, A>$

$[y = 3]$

$< 14, A>$

$[x = 6]$

Bản sao B

### Đơn vị nhất quán

$$x = 2; y = 5$$

#### Thao tác

$< 5, B>$

#### Kết quả

$[x = 2]$

$< 10, B>$

$[y = 5]$

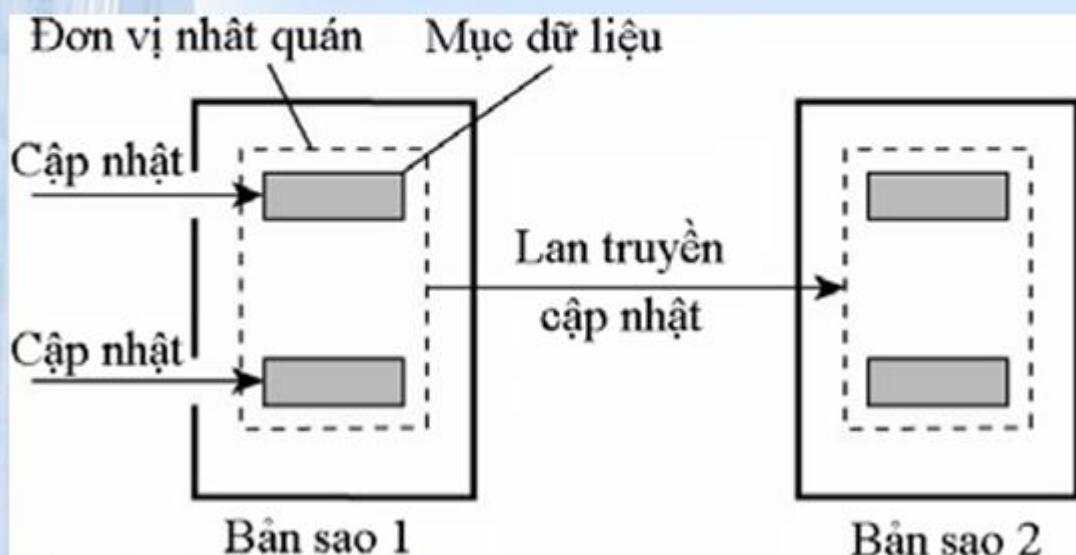
## CHỌN ĐƠN VỊ NHẤT QUÁN THÍCH HỢP

Thao tác cập nhật đã hoàn thành trên bản sao 1 phải được chuyển ngay sang bản sao 2

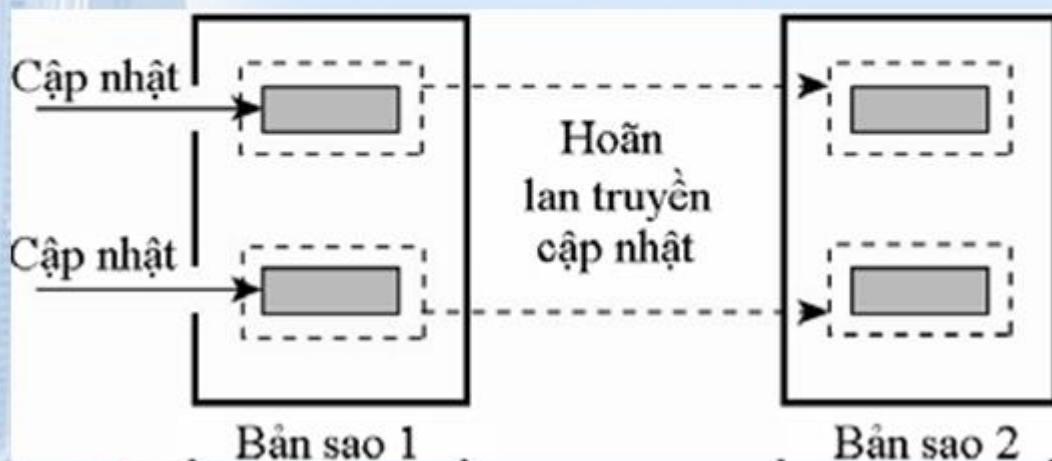
- Thao tác cập nhật đã hoàn thành trên bản sao 1 chưa cần chuyển ngay sang bản sao 2
- Chia nhỏ đơn vị nhất quán quá nhỏ không phải là giải pháp tốt, dễ làm giảm hiệu năng hệ thống

## CHỌN ĐƠN VỊ NHẤT QUÁN THÍCH HỢP

Thao tác cập nhật đã hoàn thành trên bản sao 1 phải được chuyển sang bản sao 2



- Thao tác cập nhật đã hoàn thành trên bản sao 1 chưa cần chuyển sang bản sao 2
- Chia nhỏ đơn vị nhất quán quá nhỏ không phải là giải pháp tốt, dễ làm giảm hiệu năng hệ thống



## TÍNH NHẤT QUÁN THEO THỨ TỰ CÁC THAO TÁC

- Hệ thống phân tán phải đương đầu với các công việc xử lý song song, tương tranh, chia sẻ tài nguyên...

- Nhiều mô hình đã được đưa ra, phần này sẽ đề cập tới mô hình đảm bảo tính nhất quán theo thứ tự các thao tác:

- Mô hình nhất quán chặt
- Mô hình nhất quán tuần tự
- Mô hình nhất quán tuyến tính
- Mô hình nhất quán nhân quả
- Mô hình nhất quán FIFO
- Mô hình nhất quán yếu
- Mô hình nhất quán **bảng dữ liệu**
- Mô hình nhất quán **mục dữ liệu**

### TÍNH NHẤT QUÁN THEO THỨ TỰ CÁC THAO TÁC

- Hệ thống phân tán phải đương đầu với các công việc xử lý song song, tương tranh, chia sẻ tài nguyên...
- Nhiều mô hình đã được đưa ra, phần này sẽ đề cập tới mô hình đảm bảo tính nhất quán theo thứ tự các thao tác:
  - Mô hình nhất quán chặt
  - Mô hình nhất quán tuần tự
  - Mô hình nhất quán tuyến tính
  - Mô hình nhất quán nhân quả
  - Mô hình nhất quán FIFO
  - Mô hình nhất quán yếu
  - Mô hình nhất quán đi ra (release)
  - Mô hình nhất quán đi vào (entry)

## Mô hình nhất quán chặt

### MÔ HÌNH NHẤT QUÁN CHẶT

- Định nghĩa: Bất kỳ thao tác đọc đều trả về một giá trị tương ứng với giá trị ghi gần nhất trên mục dữ liệu của x vẫn là null
- Các kí hiệu:
  - $W_i(x)a$ : thao tác ghi được thực hiện bởi tiến trình  $P_i$  lên mục dữ liệu  $x$  với giá trị  $a$ .
  - $R_i(x)b$ : thao tác đọc được thực hiện bởi tiến trình  $P_i$  lên mục dữ liệu  $x$  cho kết quả  $b$ .
  - Giá thiết  $x$  có giá trị ban đầu là null.

P1:	$W(x)a$
P2:	$R(x)a$

P1:	$W(x)a$	
P2:	$R(x)NIL$	$R(x)a$

- Mô hình sử dụng khái niệm thời gian tuyệt đối (thời gian chung cho cả hệ thống để xác định đúng khái niệm gần nhất), điều này là khó khả thi với hệ phân tán.

## Mô hình nhất quán tuần tự

### MÔ HÌNH NHẤT QUÁN TUẦN TỰ

- Kết quả thực hiện như nhau nếu các thao tác của các tiến trình được thực hiện theo thứ tự,
- Thao tác của mỗi tiến trình xuất hiện theo thứ tự đã được chương trình xác định.
- Khi các tiến trình chạy đồng thời trên các máy khác nhau thì cho phép sự đan xen của các thao tác nhưng tất cả các tiến trình đều phải nhận biết được sự đan xen của các thao tác của nhau.

Trên bản sao dữ liệu giữa các tiến trình  $P_3, P_4$ : Thao tác  $W(x)b$  thực hiện trước  $W(x)a$

Trên bản sao dữ liệu của tiến trình  $P_4$ : Thao tác  $W(x)b$  thực hiện sau  $W(x)a$

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

Nhất quán tuần tự

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

Không thỏa mãn nhất quán tuần tự

## MÔ HÌNH NHẤT QUÁN TUẦN TỤ

- Giá trị ban đầu của các dữ liệu x,y,z đều bằng 0
- Giả sử ba tiến trình P1, P2, P3 thực hiện các thao tác:

Tiến trình P1	Tiến trình P2	Tiến trình P3
$x \leftarrow 1;$ <code>print(y, z);</code>	$y \leftarrow 1;$ <code>print(x, z);</code>	$z \leftarrow 1;$ <code>print(x, y);</code>

- Về lý thuyết sẽ có 720 tổ hợp xảy ra, nếu xét các thao tác gán trước thao tác in sẽ có 30 trường hợp.

## MÔ HÌNH NHẤT QUÁN TUẦN TỤ

- Kết quả: Giá trị được in ra
- Chữ ký: Kết quả in ra theo thứ tự tiến trình P1, P2, P3  
yzxzxy

$x \leftarrow 1;$ <code>print(y, z);</code>	$x \leftarrow 1;$ <code>y \leftarrow 1;</code> <code>print(x, z);</code>	$y \leftarrow 1;$ <code>z \leftarrow 1;</code> <code>print(x, y);</code>	$y \leftarrow 1;$ <code>x \leftarrow 1;</code> <code>z \leftarrow 1;</code> <code>print(x, z);</code>
$z \leftarrow 1;$ <code>print(x, y);</code>	$z \leftarrow 1;$ <code>print(y, z);</code> <code>print(x, y);</code>	$x \leftarrow 1;$ <code>print(z, y);</code> <code>print(y, z);</code>	$print(y, z);$ <code>print(x, z);</code> <code>print(x, y);</code>

Kết quả: 001011    Kết quả: 101011    Kết quả: 010111    Kết quả: 111111  
Chữ ký: 001011    Chữ ký: 101011    Chữ ký: 110101    Chữ ký: 111111

## Mô hình nhất quán nhân quả

### MÔ HÌNH NHẤT QUÁN NHÂN QUẢ

- Mô hình này phân biệt các sự kiện có quan hệ nhân quả và các sự kiện không có quan hệ nhân quả.
- Nếu sự kiện b được gây ra hoặc bị tác động bởi một sự kiện a xảy ra sớm hơn thì mọi thực thể khác phải nhìn thấy a trước khi nhìn thấy b.
- Các thao tác ghi có quan hệ nhân quả tiềm năng phải được nhận biết bởi tất cả các tiến trình khác trong cùng một thứ tự. Các thao tác ghi đồng thời có thể nhận biết được theo thứ tự khác nhau trên các máy khác nhau.

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

### MÔ HÌNH NHẤT QUÁN NHÂN QUẢ

P1: W(x)a

P2: R(x)a W(x)b

P3: R(x)b R(x)a

P4: R(x)a R(x)b

Ví phạm nhất quán nhân quả

P1: W(x)a

P2: W(x)b

P3: R(x)b R(x)a

P4: R(x)a R(x)b

Thứ tự chính xác của các sự kiện trong nhất quán nhân quả

Mô hình hắt quá FiFo

## MÔ HÌNH NHẤT QUÁN FIFO

- Bỏ qua giới hạn về trật tự của bất kì thao tác tương tranh nào.
  - Các thao tác ghi bởi một tiến trình đơn phải được tất cả các tiến trình khác nhìn thấy theo cùng một trật tự mà chúng đề ra.
  - Thao tác ghi bởi nhiều tiến trình khác nhau có thể được các tiến trình khác nhìn thấy theo những trật tự khác nhau.

$$P_1: W(x)a$$

P2:  $R(x)a$      $W(x)b$      $W(x)c$

P3:  $R(x)b$   $R(x)a$   $R(x)c$

P4:  $R(x)a$   $R(x)b$   $R(x)c$

## Mô hình nhất quá yếu

## MÔ HÌNH NHẤT QUÁN YẾU

- Mô hình nhất quán yếu không tập trung vào các thao tác trên dữ liệu mà quan tâm đến trạng thái của các nhóm lệnh bằng việc sử dụng các biến đồng bộ hóa.
  - Mô hình nhất quán yếu có ba đặc tính sau:
    - Việc truy cập đến một biến đồng bộ hóa được kết hợp với kho dữ liệu là một nhất quán tuần tự.
    - Không có thao tác nào lên các biến đồng bộ hóa được phép thực hiện cho đến khi tất cả các thao tác ghi trước đó được hoàn thành ở mọi nơi.
    - Không có thao tác đọc hay ghi dữ liệu lên các mục dữ liệu nào được phép thực hiện cho đến khi tất cả các thao tác trước đó lên các biến đồng bộ hóa được thực hiện.

## Mô hình nhất quá bảng dữ liệu

## MÔ HÌNH NHẤT QUÁN ĐI RA (RELEASE)

- Sử dụng thêm hai lệnh: lệnh acquired để báo muồn vào vùng tới hạn (critical region) và lệnh release để báo giải phóng vùng tới hạn.
- Hai lệnh này cũng có hai cách thực thi khác nhau như: bằng một biến hoặc bằng một lệnh đặc biệt.
- Hai thao tác này chỉ thực hiện với các dữ liệu dùng chung chứ không áp dụng cho tất cả các dữ liệu.

### Điều kiện mô hình

#### ĐIỀU KIỆN MÔ HÌNH NHẤT QUÁN ĐI RA

- Trước khi thực hiện một thao tác đọc hay ghi lên dữ liệu chia sẻ thì tất cả các thao tác acquire do tiến trình này thực hiện trước đó phải hoàn tất.
- Trước khi một thao tác release được phép thực hiện thì tất cả các thao tác đọc và ghi do tiến trình này thực hiện trước đó phải được hoàn tất.

P1:	Acq(L)	W(x)a	W(x)b	Rel(L)
P2:			Acq(L)	R(x)b
P3:				Rel(L) R(x)a

### Mô hình nhất quán **mục** dữ liệu

## MÔ HÌNH NHẤT QUÁN ĐI VÀO (Entry)

- Giống như mô hình nhất quán Release, mô hình nhất quán Entry sử dụng hai lệnh acquired và release khi muốn vào vùng thời hạn.
- Các lệnh này thao tác trên từng mục dữ liệu của vùng chia sẻ.
- Tiền trình nào muốn sử dụng mục dữ liệu thì phải đợi cho tất cả các tiền trình khác giải phóng mục dữ liệu đó.
- Để ghi lên một mục dữ liệu, máy khách phải có được biến đồng bộ hóa của mục đó trong chế độ dành riêng. Điều đó có nghĩa là không máy khách nào khác có thể sử dụng biến đó. Khi máy khách cập nhật xong mục dữ liệu, thi nó giải phóng biến đó.
- Khi máy khách muốn đọc một mục dữ liệu nào đó, nó phải có được biến đồng bộ hóa kết hợp ở chế độ không dành riêng. Nhiều máy khách có thể giữ một biến đồng bộ hóa ở chế độ không dành riêng.
- Khi thực hiện một thao tác acquire, máy khách lấy về phiên bản mới nhất của mục dữ liệu từ tiền trình cuối cùng thực hiện thao tác acquire trên biến đó.

### Điều kiện mô hình

#### ĐIỀU KIỆN MÔ HÌNH NHẤT QUÁN ĐI VÀO

- Một thao tác acquire để truy cập vào một biến đồng bộ hóa không được phép thực hiện trong một tiền trình cho đến khi tất cả các cập nhật lên mục dữ liệu trong tiền trình đó được thực hiện.
- Trước khi một truy cập trong chế độ dành riêng của một tiền trình tới một biến đồng bộ hóa được phép thực hiện thì không tiền trình nào khác còn được giữ các biến đồng bộ hóa, trong chế độ không dành riêng thì không cần yêu cầu như vậy.
- Sau khi một truy cập trong chế độ dành riêng lên một biến đồng bộ hóa được thực hiện thi bất kì sự truy cập của tiền trình nào khác trong chế độ không dành riêng lên biến đó cũng không được thực hiện cho đến khi chủ nhân của biến đồng bộ thực hiện xong việc truy cập của mình.

P1: Acq(Lx) W(x)a Acq(Ly) W(y)b Rel(Lx) Rel(Ly)

P2: Acq(Lx) R(x)a R(y) NIL

P3: Acq(Ly) R(y)b

## Mô hình lấy MK làm trung tâm

**MÔ HÌNH NHẤT QUÁN  
LẤY MÁY KHÁCH LÀM TRUNG TÂM**

- Các mô hình lấy dữ liệu làm trung tâm nhằm tới cách nhìn nhận tính toàn vẹn dữ liệu toàn bộ hệ thống trong việc lưu trữ dữ liệu.
- Mô hình lấy dữ liệu làm trung tâm chủ yếu giải quyết vấn đề tương tranh và tính tuân tự thực hiện các thao thác.
- Mô hình nhất quán lấy máy khách làm trung tâm bỏ qua các yêu cầu về tính tương tranh:
  - Mô hình nhất quán sau cùng
  - Mô hình nhất quán đọc đều
  - Mô hình nhất quán ghi đều
  - Mô hình nhất quán đọc kết quả ghi
  - Mô hình nhất quán ghi theo sau đọc

Mô hình nhất quán sau cùng

## Mô hình nhất quán cuối cùng

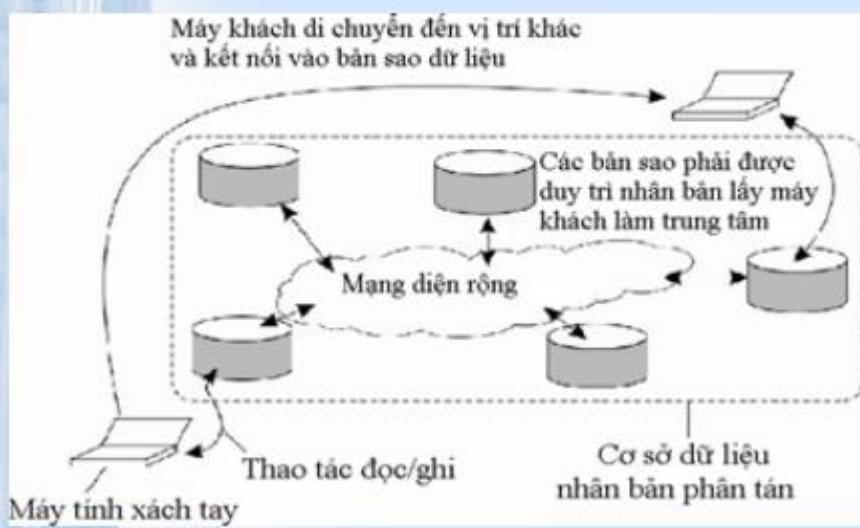
Đây là mô hình nhất quán mà khi 1 dữ liệu có nhiều bản sao thì khi thực hiện cập nhật thì tất cả các bản sao phải giống nhau (cùng được cập nhật). Tức là yêu cầu duy nhất của mô hình này là tất cả các bản sao cuối cùng rồi sẽ phải giống nhau. Tất cả các cập nhật phải đảm bảo là cuối cùng cũng sẽ được truyền tới tất cả các bản sao khác. Yêu cầu này sẽ được thực hiện tốt nếu mỗi client luôn chịu khó cập nhật cho các bản sao. Nhưng vấn đề sẽ xảy ra nếu quá trình đó diễn ra liên tục và sẽ làm ảnh hưởng đến hiệu năng của hệ thống. Hệ thống đảm bảo rằng nếu không có thêm cập nhật nào nữa, thì cuối cùng tất cả các truy nhập sẽ trả về giá trị cập nhật cuối cùng. Nếu không có trục trặc nào xảy ra, quãng thời gian lớn nhất của inconsistency window có thể được xác định dựa trên các yếu tố như độ trễ đường truyền, tải của hệ thống, và số bản sao dữ liệu được nhân bản. Khi một dữ liệu có nhiều bản sao thì yêu cầu đưa ra là sau khi các thao tác cập nhật thì tất cả các bản sao phải bằng nhau. Yêu cầu này được thực hiện tốt nếu mỗi client luôn chịu khó cập nhật cho các bản sao. Nếu các client là di động thì việc thực hiện yêu cầu trên khó khăn hơn. Phải luôn đảm bảo rằng ngay cả khi client thay đổi vị trí vật lý thì việc xử lý các bản sao cũng phải chính xác. Tức là các bản sao luôn luôn nhất quán. Ví dụ như khi người dùng cập nhật tại 1 địa điểm ( ghi dữ liệu vào database chặng hạn )( Trong 1 khoảng thời gian đủ ngắn ) Rồi đến 1 địa điểm khác , truy cập vào dữ liệu đó . Người đó sẽ tương tác với 1 cơ sở dữ liệu khác , và sẽ thấy rằng công việc mình làm hôm nay biến mất . -Ưu điểm : + Đảm bảo khả năng

trong suốt cao + Thực hiện tốt + Rẻ, dễ thực hiện. -Nhược điểm + Yêu cầu các client thực hiện cập nhật thường xuyên + Khi các client di chuyển khó thực hiện + Thường chỉ áp dụng cho những hệ thống có ít người cập nhật Hệ thống phổ biến nhất dùng mô hình này là DNS (Domain Name System). Các cập nhật đối với tên miền được phân tán dựa vào mẫu đã được cấu hình trước và kết hợp với việc cache; cuối cùng mọi khách hàng đều nhìn thấy cập nhật.

### MÔ HÌNH NHẤT QUÁN SAU CÙNG (Eventual)

- Khi có nhiều bản sao dữ liệu, một yêu cầu đặt ra là sau các thao tác cập nhật thì tất cả các bản sao sau cùng phải giống nhau. Yêu cầu này sẽ được thực hiện tốt nếu mỗi máy khách luôn cập nhật các bản sao.
- Việc cập nhật các bản sao ngay sau khi cập nhật bản chính có thể kéo dài thời gian thực hiện, do đó lập trình viên cần dự đoán thời gian thực hiện mỗi yêu cầu và lựa chọn phương án thích hợp.
- Nếu các máy khách di động thì việc thực hiện yêu cầu trên gặp khó khăn hơn. Phải luôn đảm bảo rằng ngay cả khi máy khách thay đổi về vị trí vật lý thì việc sử dụng các bản sao cũng phải chính xác.

### MÔ HÌNH NHẤT QUÁN SAU CÙNG



## MÔ HÌNH NHẤT QUÁN SAU CÙNG



Mô hình nhất quán đọc đều

## MÔ HÌNH NHẤT QUÁN ĐỌC ĐỀU

- Một tiến trình thực hiện thao tác đọc trên một mục dữ liệu thì phải đảm bảo bất kì thao tác đọc nào cũng đều cho cùng một kết quả hay kết quả gần nhất.
- Mô hình nhất quán đọc đều đảm bảo rằng một máy khách sẽ luôn nhìn thấy những dữ liệu mới hơn và không bao giờ phải nhìn thấy những dữ liệu cũ hơn những gì đã đọc trước đó.
- Khi một máy khách thực hiện một thao tác đọc trên một bản sao rồi tiếp theo lại đọc trên một bản sao khác thì bản sao thứ hai ít nhất cũng phải được ghi giống với bản sao đầu tiên.

## MÔ HÌNH NHẤT QUÁN ĐỌC ĐỀU

- Thao tác cập nhật trên L1 sẽ được lan truyền trên L2

L1: WS( $x_1$ )

R( $x_1$ )

L2:

WS( $x_1; x_2$ )

R( $x_2$ )

- Không có gì đảm bảo thao tác cập nhật trên L1 sẽ được lan truyền sang L2 và ngược lại.

L1: WS( $x_1$ )

R( $x_1$ )

L2:

WS( $x_2$ )

R( $x_2$ )

## Cách làm

Dữ liệu đánh dấu phiên bản

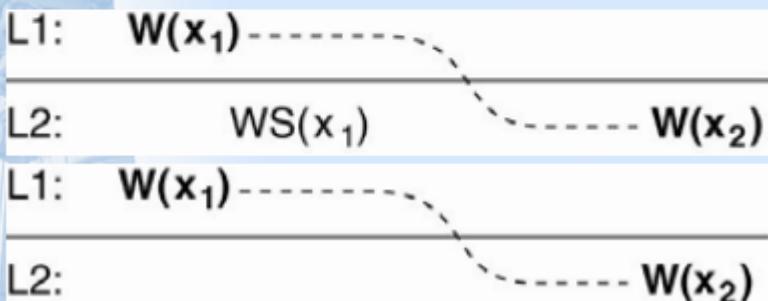
Lưu trong cache của máy khách

So sánh phiên bản

## Mô hình nhất quán ghi đè

### MÔ HÌNH NHẤT QUÁN GHI ĐỀU

- Thao tác ghi trên mục dữ liệu  $x$  của một tiến trình phải được hoàn thành trước bất kỳ một thao tác ghi nào khác trên  $x$  bởi cùng một tiến trình.
- Các thao tác ghi lên một mục dữ liệu sẽ được sắp xếp một cách có trật tự.



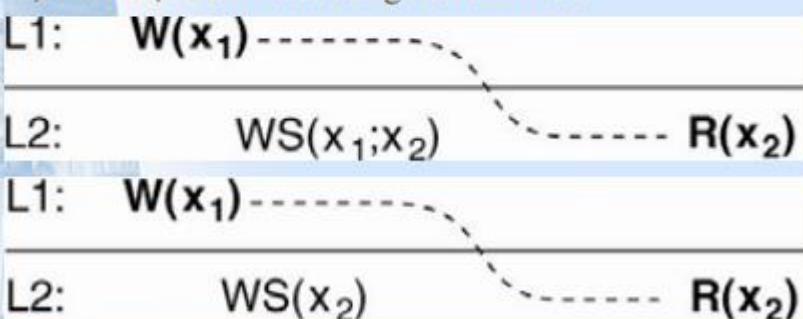
#### Cách làm

- Đưa yêu cầu thay đổi dữ liệu cho mục  $X$  vào hàng đợi xử lý

## Mô hình nhất quán đọc kết quả ghi

### MÔ HÌNH NHẤT QUÁN ĐỌC KẾT QUẢ GHI

- Người dùng được đảm bảo sẽ luôn được nhìn thấy những kết quả mới nhất.
- Tác động của một thao tác ghi của một tiến trình lên mục dữ liệu  $x$  sẽ luôn được nhìn thấy bởi một thao tác đọc lần lượt trên  $x$  của cùng tiến trình đó.



- Ghi cục bộ → Lan tỏa đến các bản sao

Mô hình nhất quán ghi theo sau đọc

### MÔ HÌNH NHẤT QUÁN GHI THEO SAU ĐỌC

- Mô hình nhất quán này ngược với nhất quán đọc kết quả ghi, nó đảm bảo người dùng sẽ luôn thực hiện thao tác ghi lên một phiên bản dữ liệu mà ít nhất cũng phải mới bằng phiên bản cuối cùng của nó.
- Tác động bởi một thao tác ghi của một tiến trình lên mục dữ liệu x sẽ luôn được nhìn thấy bởi một thao tác đọc liên tiếp lên x của cùng tiến trình đó.

L1: WS( $x_1$ )	$R(x_1)$	-
L2: WS( $x_1; x_2$ )	-	- $W(x_2)$
L1: WS( $x_1$ )	$R(x_1)$	-
L2: WS( $x_2$ )	-	- $W(x_2)$

- Dữ liệu được đánh dấu phiên bản
- Đọc phân tán
- So sánh phiên bản với dữ liệu cần ghi

## Quản lý bản sao

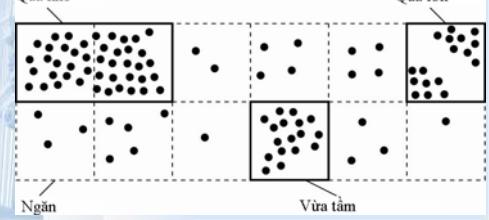
# Sắp xếp bản sao máy chủ

## QUẢN LÝ BẢN SAO

- Việc nhân bản trong hệ thống phân tán cần phải trả lời 5W (When, Where, Who, What, Why) nhằm đảm bảo tính nhất quán của hệ thống.
- Đối với câu hỏi ở đâu (Where) cần phân biệt chi tiết:
  - Máy chủ: Tìm vị trí thích hợp nhất để đặt máy chủ
  - Nội dung: Tìm máy chủ thích hợp nhất để nhân bản nội dung

## SẮP XẾP BẢN SAO-MÁY CHỦ

- Xét hình trạng của mạng
- Xét nhu cầu truy nhập nội dung
- Quá nhỏ



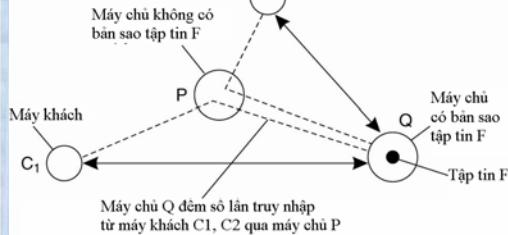
### SẮP XẾP BẢN SAO NỘI DUNG

- Các bản sao thường trực: trong tiến trình hay trên máy luôn có một bản sao. Số lượng các bản sao thường xuyên này rất ít, thường được tập hợp lại thành nhóm các máy, thường là các máy chủ Web hay các máy chủ chứa cơ sở dữ liệu dự phòng.
- Bản sao khởi đầu từ máy chủ: Các bản sao này được sử dụng để làm tăng hiệu năng. Các bản sao này được xếp đặt động dựa vào yêu cầu của máy chủ khác. Một ví dụ điển hình là dịch vụ đặt vị trí trang web sử dụng để xác định vị trí địa lý của các bản sao gần nhất khi cần.
- Các bản sao khởi đầu từ máy khách: Các bản sao này được tạo ra từ yêu cầu của máy khách, chẳng hạn như việc cache dữ liệu của một trình duyệt. Chúng được xếp đặt động dựa vào yêu cầu của máy khách.

### TỔ CHỨC BẢN SAO NỘI DUNG



### BẢN SAO KHỞI NGUỒN TỪ MÁY CHỦ



### PHÂN TÁC NỘI DUNG

- Quản lý bản sao cần phải giải quyết vấn đề lan truyền cập nhật nội dung nhân bản.
- Cần phải cân bằng các yếu tố khác nhau để đạt hiệu năng tốt nhất có thể được:
  - Trạng thái hay thao tác
  - Giao thức kéo hay đẩy
  - Truyền theo điểm-to-point (Unicast) hay theo nhóm (Multicast)

### CÁC GIAO THỨC KÉO VÀ ĐẨY

- Giao thức đẩy (Giao thức dựa trên máy chủ):** Mỗi khi có thay đổi, máy chủ sẽ gửi yêu cầu cập nhật đến tất cả các bản sao. Tính nhất quán rất cao.
- Giao thức kéo:** Máy khách hoặc máy chủ gửi yêu cầu đến máy chủ khác để nhận dữ liệu mới nhất (nếu có).

### TRẠNG THÁI HAY THAO TÁC

- **Chi thông báo là có cập nhật:** Thường dùng trong việc cache dữ liệu. Thông báo về việc mất hiệu lực của một giao thức. Phương pháp này tốt khi tỉ lệ các thao tác đọc so với thao tác ghi nhỏ.
- **Truyền dữ liệu cập nhật từ bản sao này tới một bản sao khác:** Thực hiện tốt khi có nhiều thao tác đọc. Ghi lại các thay đổi và tập hợp các cập nhật lại để truyền đi (chỉ truyền đi các thay đổi chứ không truyền cả dữ liệu đã bị thay đổi, vì thế tiết kiệm được băng thông).
- **Lan truyền các thao tác cập nhật tới các bản sao khác:** Tốn ít băng thông nhưng đòi hỏi năng lực xử lý cao vì trong nhiều trường hợp các thao tác rất phức tạp.

## CÁC GIAO THỨC KÉO VÀ ĐÂY

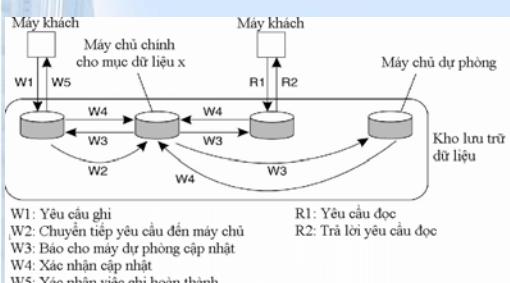
- Giao thức dây (Giao thức dựa trên máy chủ):** Mỗi khi có thay đổi, máy chủ sẽ gửi yêu cầu cập nhật đến tất cả các bản sao. Tính nhất quán rất cao.
- Giao thức kéo:** Máy khách hoặc máy chủ gửi yêu cầu đến máy chủ khác để nhận dữ liệu mới nhất (nếu có).

Vận đề	Đây	Kéo
Trạng thái máy chủ	Cần duy trì danh sách các bản sao và cache của máy khách	Không
Thông điệp gửi	Máy chủ gửi yêu cầu cập nhật cho máy khách	Định kỳ quét để lấy yêu cầu cập nhật mới nhất
Thời gian đáp ứng tại máy khách	Ngay lập tức	Thời gian quét

## CÁC GIAO THỨC NHẤT QUÁN

- Các giao thức nhất quán mô tả cách cài đặt các mô hình nhất quán.
- Các giao thức nhất quán dựa trên các giải pháp trong mô hình nhất quán liên tục:
  - Sự chênh lệch giá trị số giữa các bản sao: Giá trị có thể là của một trường dữ liệu nhưng cũng có thể là số lượng các thao tác thay đổi.
  - Sự chênh lệch trạng thái: Thời gian cập nhật cuối cùng
  - Sự chênh lệch thứ tự các thao tác cập nhật: Khá phức tạp, nhất là đối với các thao tác có ROLLBACK

## CÁC GIAO THỨC GHI TỪ XA



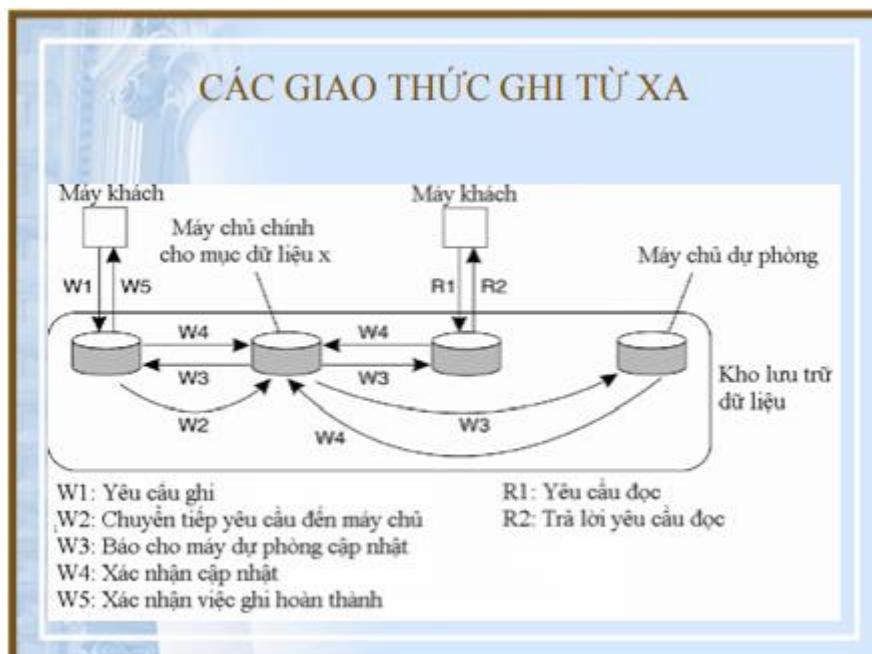
**Giao thức ghi từ xa** : có 4 bản sao

- Yêu cầu ghi dl
- Chuyển lên bản sao chính
- Bản sao chính commit phân tán
- Trả về kq cho máy khách

Ưu : Tính nhất quán cao , chỉ cần đọc cục bộ

Nhược : Thời gian đáp ứng về máy khách lâu do mỗi giao tác cần commit phân tán

Vd : Các HT tài chính(pp ghi từ xa)



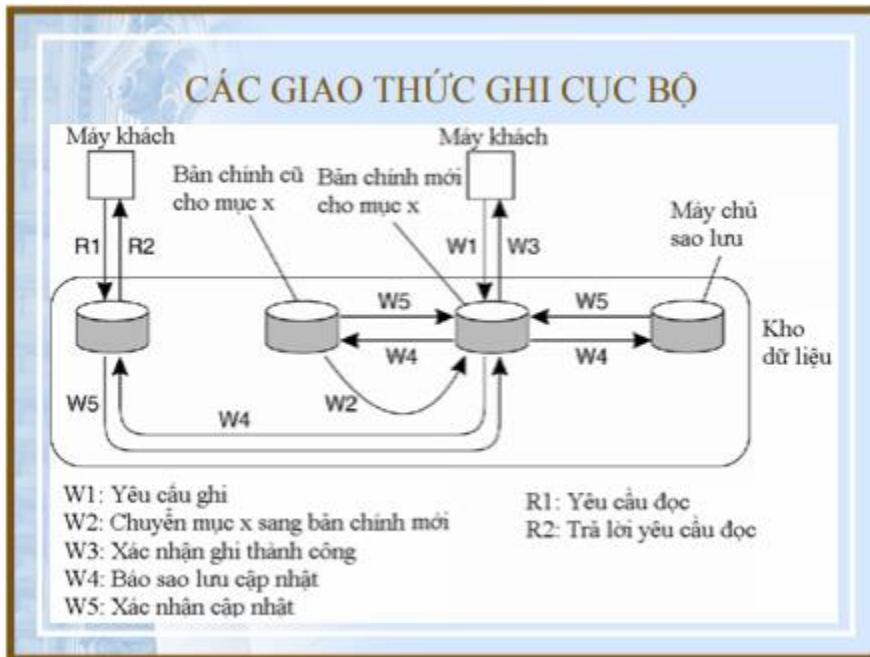
**Giao thức ghi cục bộ** : tương tự như ghi từ xa tuy nhiên

**Tính nhất quán lỏng lẻo hơn**

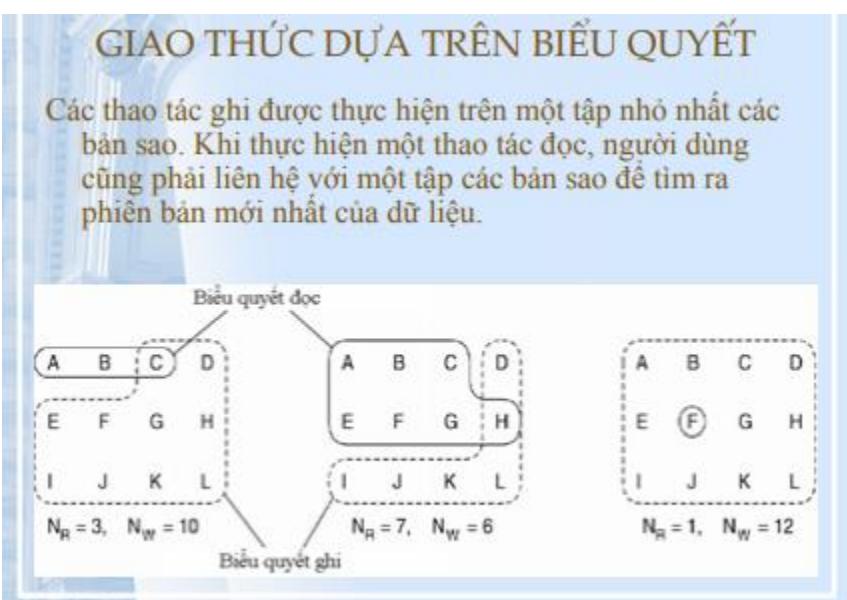
**Thời gian đáp ứng nhanh hơn**

Khác cách ghi : Lấy phiên bản từ bản chính thay đổi cục bộ , chuyển tới máy khách rồi mới cập nhật về các bản sao

Vd : Thuê bao cập nhật tK



### Giao thức dựa trên biểu quyết :



Tổng số bản sao: N

Quy tắc :

- $N_R$ : Số lượng đọc
- $N_W$ : Số lượng ghi
- $N$ : Số lượng bản sao

$$N_R + N_W > N$$

$$N_W > N/2$$



## Các Gt

### Bài tập youtube

Dạng 1 gt bekerley

Câu 1 (3 điểm) : Sử dụng giải thuật Berkeley thực hiện đồng bộ thời gian cho các tiến trình sau:

Điền giá trị cần điều chỉnh cho mỗi tiến trình (tính bằng ms)

Tiến trình	Loại	Thời gian hiện hành	Điều chỉnh (ms)	Chênh lệch (ms)	Đáp án (ms)	Điểm
P1	Điều phối	2018-09-27 09:32:29.960	0	0	107	0
P2	Thành viên	2018-09-27 09:32:28.659	0.107	-1301	1408	0
P3	Thành viên	2018-09-27 09:32:31.581	-0.107	1621	-1514	0

Thời gian sau khi đồng bộ

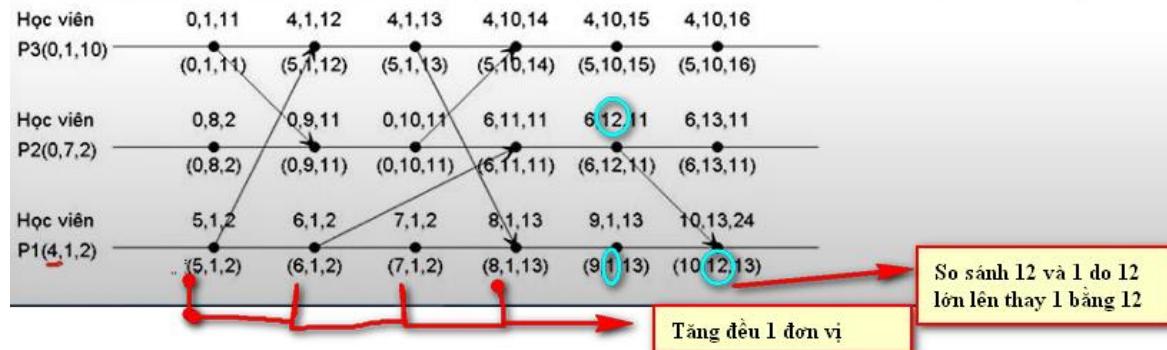
2018-09-27 09:32:28.766 2018-09-27 09:32:30.067 0 điểm

**Điểm số: 0.0**

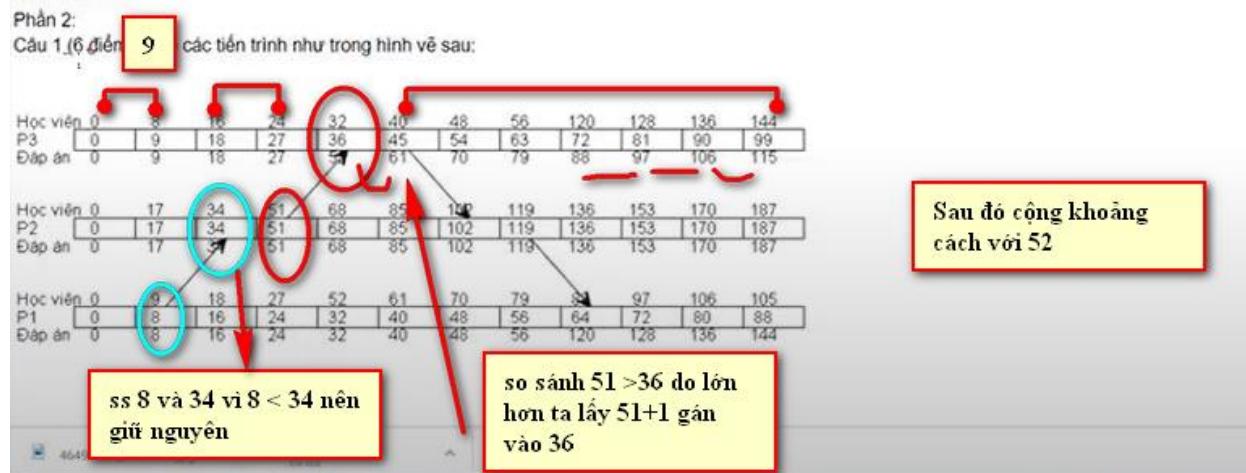
B1 : Lấy giờ Tvien -  
Gio điêu phôi  
B2 : cộng các giá trị /  
số tvien  
B3 : Cập nhật giờ

## Dạng bài Vector thời gian

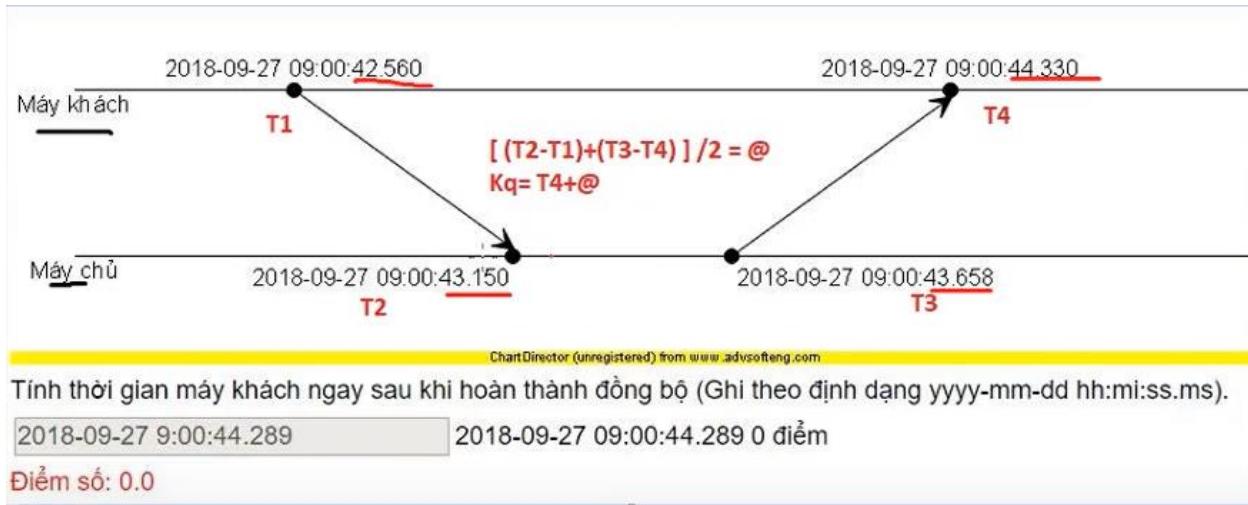
Câu 1 (7 điểm): Nhãn thời gian của mỗi tiến trình được đặt theo cấu trúc (P1,P2,P3), các sự kiện trên mỗi tiến trình thể hiện như trong hình vẽ sau:



## GT nhãn tgian lamport



## GT Cristianas



## Bài tập Web

**Máy khách gửi yêu cầu R1 và R2 đến máy chủ nhưng nhận được kết quả của yêu cầu R2 sau đó mới nhận được kết quả yêu cầu của R1. Giải thích nguyên nhân!**

Khi máy khách gửi yêu cầu đến máy chủ bằng giao thức TCP/IP theo thứ tự là R1 trước R2 thì khi yêu cầu được đưa đến tầng Internet trong mô hình TCP/IP, cụ thể ở giao thức IP các yêu cầu R1, R2 đã được chia nhỏ ra thành các gói packet ở tầng vận tải sẽ được giao thức IP chuyển các gói tin đến máy chủ.

Ví dụ r1 bị chia làm 5 gói chuyển được 4 gói qua cho máy chủ rồi còn 1 gói thì đến sau.

trong khi đó r2 chia làm 3 gói tin và được chuyển hết sang máy chủ. máy chủ tổng hợp lại để thành yêu cầu r2 trước r1

Giả dụ máy chủ xử lý yêu cầu nào đến trước thì xử lý trước thì suy ra yêu cầu r2 sẽ được xử lý trước và gửi trả kết quả về máy khách trước yêu cầu r1

## **Cam kết ba pha giải quyết vấn đề gì? Giải thích tại sao?**

Cam kết ba pha giải quyết được nhược điểm của cam kết hai pha trong trường hợp tiến trình điều phối bị lỗi. Nhược điểm của cam kết hai pha là tốn nhiều thời gian chờ đợi. Cả tiến trình điều phối lẫn các thành viên còn lại đều phải chờ đợi một thông điệp nào đó gửi đến. Nếu tiến trình điều phối bị lỗi thì hoạt động của hệ thống sẽ bị ảnh hưởng. Cam kết ba pha đã khắc phục nhược điểm này. Cam kết ba pha khá giống cam kết hai pha nhưng thêm một trạng thái PRECOMIT

## **Trình bày những vấn đề trong giao tác phân tán (distributed) và nêu biện pháp khắc phục.**

\*Những vấn đề trong giao tác phân tán(distributed): -Tính nguyên tử: mọi giao tác không thể phân chia nhỏ thành các giao tác nhỏ hơn. Nếu giao tác không thành công thì hệ thống sẽ trở lại trạng thái trước khi giao tác bắt đầu. Tính nhất quán: giao tác không xâm phạm các bất biến của hệ thống. Có nghĩa là trước khi giao tác thực thi các bất biến hệ thống như thế nào thì sau giao tác các bất biến đó vẫn giữ nguyên. ví dụ trong ngân hàng ,khi có một giao dịch nội bộ thì sau giao dịch đó tổng số tiền trong ngân hàng vẫn giữ nguyên. Tuy nhiên trong quá trình giao dịch thì số tiền đó có thể bị thay đổi , nhưng nó lại ẩn nếu đứng ở ngoài giao dịch. Tính cô lập: các giao tác đồng thời không gây trở ngại cho nhau . Nghĩa là các giao tác cùng chạy mỗi thời gian thì ko gây ra ảnh hưởng cho nhau , ko thay đổi giao tác khác Tính lâu bền: khi giao tác đã cam kết thì các thay đổi đối với nó không phải là tạm thời mà là kéo dài. \*Các biện pháp khắc phục vấn đề là: -Việc phân tán dữ liệu trở nên hiệu quả thì mỗi tài nguyên cần

được quản lý bằng một chương trình, cung cấp một giao diện truyền thông cho phép tài nguyên được điều khiển truy cập và cấp nhật một cách tin cậy cũng như đồng nhất. -Cần phải tích hợp các ứng dụng một cách thích hợp từ cơ sở dữ liệu của nó và các thành phần ứng dụng của nó có thể truyền thông trực tiếp với nhau. Vì nhu cầu truyền thông giữa các ứng dụng nên dẫn đến nhiều mô hình truyền thông cần phải tồn tại.

## **Có thể sử dụng nhãn thời gian Lamport để truyền tin nhóm theo thứ tự được không? Giải thích vì sao?**

<http://hocvienmang.com/website/PrintArticle.aspx?ArticleId=157720>

Có thể sử dụng nhãn thời gian Lamport để truyền tin nhóm theo thứ tự vì nhãn thời gian được thực hiện bằng đồng hồ logic. Khi site của một tiến trình có yêu cầu cho CS, nó cập nhật đồng hồ địa phương của mình và phân công các yêu cầu một nhãn thời gian. Thuật toán thực hiện CS theo thứ tự ngày càng tăng của thời gian. Mỗi site S<sub>j</sub> giữ một hàng đợi, request\_queuei, trong đó có yêu cầu loại trừ lẫn nhau theo lệnh của nhãn thời gian của chúng. Hàng đợi này alf khác nhau từ các hàng đợi có chứa các yêu cầu địa phương để thực hiện CS đang chờ đến lượt mình). Thuật toán này yêu cầu các kênh truyền thông để cung cấp các gói thông tin theo trật tự FIFO.

## **Trình bày những điểm khác biệt giữa mô hình nhất quán lấy dữ liệu làm trung tâm và mô hình nhất quán lấy máy khách làm trung tâm.**

Điểm khác biệt giữa mô hình nhất quán lấy dữ liệu làm trung tâm và mô hình nhất quán lấy máy khách làm trung tâm là :

- Mô hình lấy dữ liệu làm trung tâm nhằm tối cách nhìn nhận tính toàn vẹn dữ liệu toàn bộ hệ thống trong việc lưu trữ và nó chủ yếu giải quyết vấn đề tương tranh và tính tuân tự thực hiện các thao tác. Còn Mô hình lấy máy khách làm trung tâm bỏ qua các yêu cầu về tính tương tranh.
- Việc cài đặt của mô hình nhất quán lấy máy khách làm trung tâm dễ dàng hơn việc cài đặt mô hình nhất quán lấy dữ liệu làm trung tâm.
- Mô hình nhất quán lấy dữ liệu làm trung tâm gồm có 2 kiểu chính :
  - + Nhất quán liên tục.
  - + Nhất quán thứ tự các thao tác.
- Mô hình nhất quán lấy máy khách làm trung tâm gồm 5 kiểu:
  - + Nhất quán sau cùng.
  - + Nhất quán đọc đều.
  - + Nhất quán ghi đều.
  - + Nhất quán đọc kết quả ghi.
  - + Nhất quán ghi sau khi đọc.
- Nói笼 về điều kiện:
  - + Nói笼 giá trị.
  - + Nói笼 trạng thái.
  - + Nói笼 các thao tác thực hiện.

## Có thể sử dụng phương pháp RPC bằng cách đưa tên và nội dung của tập tin vào tham số của thủ tục hay không? Vì sao?

Có thể sử dụng phương pháp RPC bằng cách đưa tên và nội dung của tập tin vào tham số của thủ tục. Bởi vì: Remote Procedure Call (RPC) – Thủ tục gọi hàm từ xa là một kỹ thuật tiên bộ cho quá trình kết nối từ Client đến Server để sử dụng các ứng dụng và dịch vụ. RPC cho phép client có thể kết nối tới 1 dịch vụ sử dụng dynamic port nằm ở một máy tính khác. RPC được thiết kế để cung cấp cho việc truyền tải thông tin giữa client và server dễ dàng hơn, bảo mật hơn, và thuận tiện hơn cho việc đồng bộ hóa các luồng dữ liệu. Các

hàm chứa trong RPC hỗ trợ cho việc truy cập bất kỳ chương trình nào đòi hỏi phương pháp giao tiếp từ client đến server. Khi bạn viết chương trình, át hẳn bạn cũng đã quen với các khái niệm thủ tục và hàm. Các đoạn chương trình lặp đi lặp lại bạn viết lại thành một hàm (hay thủ tục) và sau đó khi dùng chỉ cần gọi thủ tục hoặc hàm đó với các tham số thích hợp. Các thủ tục hoặc hàm đó bạn có thể: -Để chung trong một file nguồn cùng với chương trình chính, -Để trong một file nguồn khác và được include vào file nguồn chính khi cần, - Được biên dịch sẵn và để trong một thư viện hoặc unit để các chương trình khác của bạn (cùng được viết bằng một ngôn ngữ) sử dụng, -Được biên dịch sẵn để trong file DLL để các chương trình (được viết bằng các ngôn ngữ khác nhau) sử dụng. Điểm chung của tất cả các phương thức trên là các hàm và thủ tục cần gọi đều nằm trên cùng một máy với nơi gọi chúng. Tuy nhiên bạn có thể thấy là nơi gọi và hàm cần gọi có thể được tách rời nhau ra: từ chung file, đến khác file, rồi đến khác ngôn ngữ.

## Sử dụng giao thức NTP có đảm bảo đồng bộ thời gian chính xác hay không? Vì sao?

Giao thức NTP dùng để đồng bộ đồng hồ của các hệ thống máy tính thông qua mạng dữ liệu chuyên mạch gói với độ trễ thay đổi. Tuy nhiên giao thức này cũng chưa chắc đã đảm bảo đồng bộ thời gian chính xác bởi vì giao thức này giải quyết vấn đề trễ thông tin lưu chuyển trên mạng bằng cách tính toán tương đối độ lệch thời gian theo giải thuật Cristian và hoạt động theo phương thức:

+NTP client gửi một gói tin trong đó chưa một thẻ thời gian tới cho NTP server.

+NTP server nhận được gói tin, gửi trả lại NTP client một gói tin khác, có thẻ thời gian là thời điểm nó gửi gói tin đó đi.

NTP client nhận được gói tin đó, tính toán độ trễ dựa vào thẻ thời gian mà nó nhận được cùng với độ trễ đường truyền, NTP client sẽ set lại thời gian của nó.

Như vậy độ chính xác là <50ms nên có thể đồng bộ vẫn có thể không chính xác.

## **Trình bày về tính trong suốt và phân loại.**

Tính chất mấu chốt nhất phân biệt hệ phân tán với các hệ thống khác là tính trong suốt. Mục tiêu trong suốt đối với người sử dụng nhằm che giấu vị trí thực của thông tin đối với người sử dụng, người sử dụng không biết được thông tin được lưu trữ ở đâu và xử lý trên máy tính nào trong mọi hoàn cảnh và tạo ra một môi trường thuận nhất cho người dùng. Tính trong suốt thể hiện trong nhiều khía cạnh, những khía cạnh điển hình nhất:

- Trong suốt truy nhập: Truy nhập đối tượng địa phương/toàn cục theo cùng một cách thức. Sự tách rời vật lý của các đối tượng hệ thống được che khuất tới người dùng.
- Trong suốt định vị (còn được gọi là trong suốt tên): Người dùng không nhận biết được vị trí của đối tượng. Đối tượng được định vị và chỉ dẫn theo tên lôgic trong một hệ thống thống nhất.
- Trong suốt di trú (còn được gọi là độc lập định vị): là tính chất bổ sung vào trong suốt định vị theo nghĩa không những đối tượng được chỉ dẫn bằng tên lôgic mà đối tượng còn được di chuyển tới định vị vật lý khác mà không cần đổi tên.
- Trong suốt đồng thời: cho phép chia sẻ đối tượng dùng chung không gặp tranh chấp. Nó tương tự như khái niệm phân chia thời gian theo nghĩa khái quát.
- Trong suốt nhân bản: đưa ra tính nhất quán của đa thể hiện (hoặc vùng) của file và dữ liệu. Tính chất này quan hệ mật thiết với trong suốt đồng thời song được cụ thể hơn vì file và dữ liệu là loại đối tượng đặc biệt,
- Trong suốt song song: cho phép các hoạt động song song mà người dùng không cần biết hoạt động song song đó xảy ra như thế nào, ở đâu và khi nào. Tính song song có thể không được người dùng đặc tả.
- Trong suốt lỗi: cung cấp khả năng thứ lỗi của hệ thống được hiểu là lỗi trong hệ thống có thể được biến đổi thành sự giảm hiệu năng hệ thống một cách mềm dẻo hơn chứ không phải chỉ là làm cực tiểu sự đỗ vỡ và nguy hiểm đối với người dùng,
- Trong suốt hiệu năng: cố gắng giành được tính nhất quán và khẳng định (không cần thiết ngang bằng) mức độ hiệu năng thậm chí khi thay đổi cấu trúc hệ thống hoặc phân bố tải. Hơn nữa, người dùng không phải chịu sự chậm trễ hoặc thay đổi quá mức khi thao tác từ xa.
- Trong suốt hiệu năng còn được thể hiện là hiệu năng hệ thống không bị giảm theo thời gian.
- Trong suốt kích thước: liên quan đến tính mềm dẻo và tiềm tàng. Nó cho phép sự tăng trưởng của hệ thống được che khuất

đối với người sử dụng. Kích thước hệ thống không tạo ra tác động đối với nhận thức của người dùng. - Trong suốt duyệt lại chỉ dẫn rằng sự tăng trưởng hệ thống theo chiều dọc là tỷ lệ nghịch với sự tăng trưởng hệ thống theo chiều ngang. Sự duyệt lại phần mềm bị che khuất đối với người dùng. Trong suốt duyệt lại cũng được hiểu như trong suốt phân đoạn.

## **Phân tích qui trình hoạt động của kiến trúc tập trung khi xây dựng hệ thống phân tán.**

kiến trúc tập trung khi xây dựng hệ thống phân tán mang tính chất tập trung mọi thứ về một trung tâm để ra các quyết định vận hành. Cái lợi của mô hình này là: - Không có sự chòng chéo, mỗi quyết định / thông tin / dữ liệu đều là duy nhất, tránh rắc rối khi có nhiều dublicate. - Hoạt động xuyên suốt, ít trở ngại. - Truy cập thông tin nhanh và chính xác, do chỉ cần kết nối với trung tâm là được. Khiêm khuyết của mô hình : \* Tuy vậy, mô hình tập trung gặp khuyết điểm là tôn kém rất nhiều để có thể xây dựng được cả hệ thống lớn, lại có những hệ thống lớn không thể đạt được mô hình tập trung vì các bộ phận của nó không tuân theo một trung tâm đầu não nào. Ngoài ra, khi trung tâm đầu não gặp vấn đề thì toàn bộ hệ thống bị tê liệt

## **Xây dựng hệ thống thông tin chứng khoán nên chọn mô hình nhân bản nào? Vì sao?**

Xây dựng hệ thống thông tin chứng khoán nên chọn mô hình nhân bản server.

Vì :

- Nhân bản server sẽ giúp tăng tốc độ tin cậy của hệ thống. Khi 1 server bị tấn công (DOS,DDOS) hoặc hư hại gì đó, ta có thể chuyển sang server để dùng, ngoài ra có thể sử dụng các bản sao của server để sửa chữa các server bị hư hại. Hệ thống thông tin chứng khoán sẽ có rất nhiều người đăng nhập vì thế chúng ta phải nhân bản server để

khi server bị hư hại gì ta sẽ có biện pháp để khắc phục nhanh nhất cho khách hàng.

- Nhân bản server để tăng hiệu suất. Nguyên tắc cơ bản để tăng hiệu suất làm việc là tăng số tiến trình xử lý song song. Dựa vào đó, trong hệ thống phân tán ta có thể cải thiện hiệu suất bằng cách tạo ra bản sao của máy và sau đó phân chia công việc cho từng bản sao. Hệ thống thông tin chứng khoán có thể đặt server ở nhiều nơi khác nhau giúp việc xử lý nhanh hơn rất nhiều. Đảm bảo được tính trong suốt về vị trí của hệ thống phân tán. Tuy nhiên việc nhân bản dữ liệu đòi hỏi phải chắc chắn rằng khi một bản sao được cập nhật thì tất cả những bản còn lại cũng phải được cập nhật.
- Vì phải đảm bảo không bị ngừng hoạt động, nên phải có nhiều server, nếu 1 cái dừng hoạt động thì hệ thống vẫn hoạt động bình thường.

## **Phân tích ưu điểm và nhược điểm của các phương pháp điều khiển tương tranh.**

### **1. Sử dụng khóa**

- **Ưu điểm** Đảm bảo tính tuần tự không xảy ra tương tranh
- **Nhược điểm** Tăng tải xử lý cho hệ thống. Sử dụng khóa có thể dẫn đến trạng thái khóa chết(deadlock) dẫn tới bộ điều khiển tương tranh sẽ không cấp khóa cho đến khi khóa này được giải phóng

### **2. Cấp quyền**

- **Ưu điểm** Khắc phục được hiệu năng của hệ thống so với phương pháp sử dụng khóa
- **Nhược điểm** Nếu phát hiện xung đột hủy bỏ giao tác nào đó vì vậy nếu hủy giao tác đang thực hiện mà giao tác đó chưa thực hiện xong dẫn đến bản dữ liệu tạm thời không được ghi

### **3. Nhãn thời gian**

- **Ưu điểm** Sẽ không còn hiện tượng tương tranh. Ưu tiên những giao tác thực hiện nhanh và thường xuyên hơn các giao tác khác
- **Nhược điểm** Cần phải có một thành phần điều phối để đảm bảo những giao tác có thời gian thực hiện nhanh và thường xuyên sẽ được ưu tiên

## **Phân tích ưu điểm và nhược điểm của Mô hình nhất quán tuyến tính**

Mô hình nhất quán tuyến tính  
Ưu điểm: + Mô hình nhất quán tuyến tính yếu hơn mô hình nhất quán chặt nhưng mạnh hơn mô hình nhất quán tuần tự.  
+ Kết quả của bất kì sự thực hiện nào là như nhau nếu thao tác của tất cả các tiến trình lên dữ liệu được thực hiện tuần tự và các thao tác của mỗi tiến trình xuất hiện trong chuỗi phải theo đúng thứ tự đã được chỉ ra trong chương trình đó.  
+ Trong nhất quán tuyến tính thì việc lưu dữ liệu cũng phải đảm bảo tính tuyến tính tuần tự.  
+ Việc sử dụng nhãn thời gian làm cho thứ tự sắp xếp chặt chẽ hơn.

Nhược điểm:  
+ Mô hình có tính nhất quán cao nhưng hiệu năng hệ thống lại không cao.

## **Phân tích ưu điểm và nhược điểm của Mô hình nhất quán FIFO**

Mô hình nhất quán FIFO là mô hình nhất quán yếu thuộc các mô hình nhất quán lấy dữ liệu làm trung tâm. Toàn vẹn FIFO thỏa mãn:  
- Các thao tác ghi bởi một tiến trình đơn phải được tất cả các tiến trình khác nhìn thấy theo cùng một trật tự mà chúng đề ra.  
- Thao tác ghi dữ liệu của các tiến trình khác nhau có thể được nhìn thấy theo những trật tự khác nhau bởi các tiến trình khác nhau.

ƯU ĐIỂM:  
- Đơn giản, dễ cài đặt vì nhất quán FIFO bỏ qua giới hạn về trật tự của bất kì thao tác đồng thời nào.

NHƯỢC ĐIỂM:  
- Việc sử dụng các bản sao gây ra các hạn chế đó là tính nhất quán dữ liệu của hệ thống bị suy giảm.  
Do sử dụng bản sao nên có thể xảy ra trường hợp có sự thay đổi trên một dữ liệu mà không cập nhật trên các bản sao của nó  
- Sai lệch giữa các bản sao có thể lớn.

Phân tích vì sao không nên xây dựng hệ thống phân tán với độ trong suốt tuyệt đối.

Tính chất mấu chốt nhất phân biệt hệ thống phân tán với các hệ thống khác là tính trong suốt. Mục tiêu trong suốt đối với người sử dụng nhằm che giấu vị trí thực của thông tin đối với người sử dụng, người sử dụng không biết được thông tin được lưu trữ ở đâu và xử lý trên máy tính nào, tạo ra một môi trường thuận nhất cho người dùng.

Sự che khuất thông tin phụ thuộc hệ thống khỏi người dùng dựa trên việc cân bằng giữa tính đơn giản và tính hiệu quả, nhưng, hai tính chất này là xung đột nhau. Bởi vậy, mục tiêu trong suốt tuyệt đối là không thích hợp. Hơn thế nữa, ta xét ví dụ:

Ví dụ:

- Trong suốt định vị: ẩn nói lưu trữ thông tin, nghĩa là người dùng không nhận biết được vị trí của đối tượng.
- Nhưng, khi bạn sử dụng điện thoại cố định thì liên lạc điện thoại nội tỉnh sẽ khác với liên lạc điện thoại ngoại tỉnh (phải bổ sung thêm mã vùng tỉnh).

Điều đó vi phạm trong suốt định vị.

## **Phân tích ưu điểm và nhược điểm của Mô hình nhất quán tự**

- Nhất quán tự là một trong những mô hình thông nhất sử dụng trong lĩnh vực máy tính đồng thời (ví dụ như trong phân phối bộ nhớ chia sẻ, giao dịch phân tán, vv). - Mô hình tự có điều kiện cần phải thỏa mãn như sau : “Kết quả của bất kỳ sự thực thi nào cũng phải giống nhau, như thể tất cả các hành động đọc , ghi dữ liệu là của mọi tiến trình và được thực thi theo một thứ tự nào đó. Hành động đọc ghi của mỗi tiến trình riêng biệt xuất hiện trong thứ tự đó là do chương trình của chúng nó quyết định.” Có nghĩa là khi mà các tiến trình được diễn ra đồng thời trên các máy khác nhau mà có sự xen ngang của hành động đọc ghi dữ liệu thì tất cả các tiến trình đều nhìn thấy hành động đó. - Nhất quán tự đòi hỏi rằng: + Tất cả các lệnh được thực hiện theo thứ tự. + Mọi hoạt động đọc ghi trở nên ngay lập tức nhìn thấy được trên toàn hệ thống. - Ưu nhược điểm: + Ưu điểm : Trong điều kiện của mô hình không nhắc đến khái niệm trước sau về thời gian tuyệt đối khắc phục được nhược điểm không thể thực thi của mô hình nhất quán chặt chẽ. + Nhược điểm : Chính vì sự định nghĩa thứ tự của các hành động đọc ghi của mỗi tiến trình

chưa có khái niệm tem thời gian nên nó không thật sự chặt chẽ, nên mô hình nhất quán này hơi yếu

## **Phân tích những lỗi có thể xảy ra khi gọi thủ tục từ xa, trình bày các giải pháp khắc phục.**

các vấn đề gọi thủ tục từ xa: thực thi mã lệnh được thực hiện trên các vùng nhớ khác nhau hoặc nếu một trong hai máy tính bị lỗi trong quá trình thực thi mã lệnh cũng nảy sinh nhiều vấn đề phức tạp. -Vấn đề truyền tham số: +Gọi thủ tục truyền thông có ba cách truyền tham số: truyền giá trị, truyền tham chiếu và truyền con trỏ. Đối với việc gọi thủ tục từ xa, không thể áp dụng phương pháp truyền con trỏ do đó chỉ có thể thực hiện bằng cách truyền giá trị hoặc truyền tham chiếu. Truyền giá trị trỏ nên phức tạp đối với hệ thống phân tán không đồng nhất(có hai cách định dạng dữ liệu, các bộ vi xử lý của Intel theo định dạng little endian,trong khi đó các bộ vi xử lý của Sun theo định dạng big endian). =>Khắc phục: phải bổ sung thêm thông tin về kiểu dữ liệu của tham số. +Con trỏ là kiểu dữ liệu đặc biệt, nó lưu trữ địa chỉ của một biến số mà địa chỉ đó chỉ có ý nghĩa bên trong một máy tính, như vậy không thể áp dụng phương pháp này để truyền tham số trong gọi thủ tục từ xa. =>khắc phục: Giải pháp truyền tham chiếu có thể thực hiện bằng cách tạo một biến tham chiếu tương ứng trên Skeleton và thủ tục trên máy chủ sẽ sử dụng địa chỉ của biến này như phương pháp truyền tham chiếu của một thủ tục thông thường.Phương pháp gọi thủ tục từ xa che giấu quá trình trao đổi thông tin trên mạng,quá trình này được thực hiện bằng việc chuyển đổi tham số sang thông điệp trong các stub. Để thực hiện công việc này, cả phía máy khách và máy chủ đều phải tuân thủ quy định về định dạng tham số, đó là giao thức gọi thủ tục từ xa. Để đơn giản hóa quá trình tạo Stub cho máy trạm và máy chủ, một ngôn ngữ mới được sử dụng gọi là ngôn ngữ định nghĩa giao diện (IDL). Lập trình viên chỉ việc viết các hàm và các thủ tục theo qui định của ngôn ngữ, sau đó dùng chương trình dịch IDL để chuyển thành ngôn ngữ lập trình tương ứng. - Gọi thủ tục từ xa tiến trình trên máy khách sẽ bị phong tỏa cho đến khi nhận được kết quả trả về, việc chờ đợi này là không cần thiết. => khắc phục: với giải pháp gọi không đồng bộ, nó cho phép tiến trình gọi trên máy khách gửi yêu cầu đến máy chủ, sau khi máy chủ xác nhận đã nhận được yêu cầu, tiến trình trên máy khách có thể tiếp tục xử lý các tác vụ

khác mà không cần chờ đợi kết quả xử lý của máy chủ, như vậy sẽ rút ngắn thời gian phong tỏa hệ thống.

## **Vì sao sử dụng giao thức cam kết hai pha không thể giải quyết trọn vẹn vấn đề phong tỏa hệ thống?**

Chế độ bị phong tỏa hệ thống (blocked): Trong chế độ bị phong tỏa, khi tiến trình client hoặc server phát ra lệnh gửi dữ liệu (send), việc thực thi của tiến trình sẽ bị tạm ngừng cho tới khi tiến trình nhận phát ra lệnh nhận dữ liệu (receive). và nhược điểm chính của cam kết hai pha là tốn nhiều thời gian chờ đợi. Cả quan lí và các thành viên còn lại đều phải chờ một bản tin nào đó được gửi đến cho mình. Nhược điểm thứ hai là nếu HT quản lí bị lỗi thì hoạt động của cả hệ thống sẽ bị ảnh hưởng.

## **Phân tích ưu điểm và nhược điểm của Mô hình nhất quán yếu**

### **Ưu điểm của mô hình nhất quán yếu là:**

- Làm tăng tính đồng nhất dữ liệu.
- Trật tự được các thao tác vì thực hiện thành nhóm nhiều lệnh một lúc.

### **Nhược điểm của mô hình nhất quán yếu là:**

- Phải thực hiện từng nhóm các thao tác một có nghĩa là phải đợi chờ đến khi có nhiều hơn hai thao tác mới thực hiện lan tỏa yêu cầu bằng biến đồng bộ, việc này làm giảm đi thời gian thực hiện từng công việc vì phải đợi thao tác khác.
- Phải sử dụng đến hai biến đồng bộ là bắt đầu đồng bộ và kết thúc đồng bộ.

### **Nêu các nguyên nhân không che dấu được hoàn toàn các lỗi xảy ra trong hệ thống phân tán**

Che dấu lỗi là đảm bảo người dùng hoàn toàn không biết về các lỗi xảy ra trong hệ thống và sự khắc phục các lỗi này. Che dấu lỗi là một trong những yêu cầu khó thực hiện nhất và có thể không thực hiện được trong một số tình huống cụ thể. Một số nguyên nhân dẫn tới việc không che dấu được lỗi xảy ra trong hệ thống là:

- + Ta không thể phân biệt được là tài nguyên truy cập chậm hay là không truy cập được. Đây là điểm khó nhất. Ví dụ: Khi truy cập 1 web server bận, trình duyệt thông báo “time-out”, thì ta không thể biết được server đó có bị lỗi hay không.
- + Các nhà mạng không đáng tin cậy.
- + Khi một hệ thống cố gắng để che dấu lỗi thì có thể hiệu suất của hệ thống có thể sẽ bị ảnh hưởng xấu. Do đó, để đảm bảo hiệu suất của hệ thống, thì việc che dấu lỗi sẽ bị cản trở

### **Máy chủ cung cấp dịch vụ tập tin thường được cài đặt theo dạng nào, vì sao?**

Máy chủ cung cấp dịch vụ tập tin, tên Tiếng Anh là File-Server, thuộc dạng Client-Server. Máy chủ là một máy tính trong mạng có

mục đích chính là cung cấp một địa điểm để lưu trữ các tập tin máy tính được chia sẻ (như tài liệu, các file âm thanh, hình chụp, phim ảnh, hình ảnh, cơ sở dữ liệu, vv...) mà có thể được truy cập bởi các máy trạm làm việc trong mạng máy tính. Thuật ngữ máy chủ nêu bật vai trò của máy trong sơ đồ Client-server, nơi mà các khách hàng là các máy trạm sử dụng kho lưu trữ. Máy chủ tập tin thường không thực hiện bất kỳ tính toán, và không chạy bất kỳ chương trình nào thay mặt cho khách hàng (client). Nó được thiết kế chủ yếu để cho phép lưu trữ nhanh chóng và lấy dữ liệu, các tính toán được thực hiện bởi các máy trạm. Trong mô hình cơ sở dữ liệu theo kiểu file - server các thành phần ứng dụng và phần mềm cơ sở dữ liệu ở trên một hệ thống máy tính và các file vật lý tạo nên cơ sở dữ liệu nằm trên hệ thống máy tính khác. Một cấu hình như vậy thường được dùng trong môi trường cục bộ(ví dụ trường học, văn phòng), trong đó một hoặc nhiều hệ thống máy tính đóng vai trò của server, lưu trữ các file dữ liệu cho hệ thống máy tính khác thâm nhập tới. Trong môi trường file - server, phần mềm mạng được thi hành và làm cho các phần mềm ứng dụng cũng như phần mềm cơ sở dữ liệu chạy trên hệ thống của người dùng cuối coi các file hoặc cơ sở dữ liệu trên file server thực sự như là trên máy tính của người chính họ. Mô hình file server rất giống với mô hình CSDL tập trung nhưng cũng có điểm khác. Các file cơ sở dữ liệu nằm trên máy khác với các thành phần ứng dụng và phần mềm cơ sở dữ liệu; tuy nhiên các thành phần ứng dụng và phần mềm cơ sở dữ liệu có thể có cùng thiết kế để vận hành một môi trường tập trung. Thực chất phần mềm mạng đã làm cho phần mềm ứng dụng và phần mềm cơ sở dữ liệu tưởng rằng chúng đang truy nhập cơ sở dữ liệu trong môi trường cục bộ. Một môi trường như vậy có thể phức tạp hơn mô hình tập trung bởi vì phần mềm mạng có thể phải thực hiện cơ chế đồng thời cho phép nhiều người dùng cuối có thể truy nhập vào cùng cơ sở dữ liệu

## **Phân tích vì sao nên hạn chế số lượng luồng (thread) trong tiến trình máy chủ.**

Máy chủ là máy chạy các tiến trình để cung cấp dịch vụ theo yêu cầu của các máy trạm. Máy chủ thường là những máy tính có cấu hình đủ lớn để luôn sẵn sàng đáp ứng các yêu cầu dịch vụ của máy khách. Máy chủ đa luồng tiếp nhận yêu cầu nhưng không xử lý mà chuyển

yêu cầu đó cho luồng khác xử lý. Khi một tiến trình máy chủ thực hiện nếu như ta không giới hạn số lượng luồng thì sẽ tốn nhiều thời gian hơn Ví dụ: Cùng một chương trình xử lý gồm 8 luồng thì hết n thời gian. Nhưng nếu là 50 luồng thì sẽ hết  $n^*10$  thời gian. Nay giờ ta xét các vấn đề ảnh hưởng đến hiệu suất của máy tính là: - Hạn chế CPU (cần nhiều tài nguyên CPU) - Hạn chế Memory (cần nhiều tài nguyên bộ nhớ RAM) - Hạn chế I / O (Mạng và / hoặc tài nguyên ổ cứng) Như vậy nhu cầu sử dụng 50 luồng dẫn đến việc cần nhiều tài nguyên CPU gây lãng phí Vì vậy chúng ta nên hạn chế số lượng luồng trong tiến trình máy chủ để không gây lãng phí tài nguyên, tài nguyên bộ nhớ và các thiết bị vào ra.

## **Phân tích những kỹ thuật thường được áp dụng đối với việc xử lý trong các hệ thống phân tán qui mô lớn**

Các kỹ thuật thường được áp dụng xử lý trong hệ thống phân tán lớn theo em thường để giải quyết các vấn đề: Kiểm soát sự đồng thời và đồng bộ hóa dữ liệu Kiểm soát sự đồng thời để giải quyết tranh chấp của các giao tác( đảm bảo kết quả chuẩn xác từ các hoạt động truy cập song song, mà vẫn cho kết quả nhanh nhất có thể ). Giao tác là một chuỗi các yêu cầu của máy khách được thực hiện như một đơn vị riêng, trong cơ sở dữ liệu nó gồm 1 hoặc nhiều các hoạt động trong cơ sở dữ liệu, mà chia ra thành 2 loại: đọc - lấy dữ liệu từ database ,và viết - thêm,sửa, xóa dữ liệu trong database. Khi các giao tác xảy ra đồng thời cùng truy cập một dữ liệu mà có ít nhất một giao tác chứa hoạt động viết thì sẽ dẫn đến xung đột. Vì vậy người ta có kỹ thuật sử dụng khóa, nó giúp tuần tự hóa các hoạt động trong các giao tác. Các giao tác sẽ được khóa chung 1 ô khóa, và chỉ 1 chìa khóa, trong 1 thời điểm chỉ có 1 giao tác được lấy chìa khóa,các giao tác khác sẽ phải đợi, sau khi thực hiện xong thì giao tác này trả chìa khóa và các giao tác khác tiếp tục lần lượt lấy chìa khóa.

## **Phân tích những điểm hạn chế của kiến trúc nhiều bên (Multitiered)**

Kiến trúc ba tầng được ứng dụng tương đối phổ biến, tuy nhiên trong một số tình huống chúng ta có thể mở rộng thành kiến trúc n-tầng. Về cơ bản việc phân tầng phải đảm bảo các nguyên tắc sau: - Tầng ứng dụng: quản lý tương tác của người dùng với ứng dụng. - Tầng trình diễn: Xác định cách thức hiển thị giao diện người dùng và cách quản lý các yêu cầu của người dùng. - Tầng nghiệp vụ: Mô hình hóa các quy tắc nghiệp vụ. - Tầng dịch vụ hạ tầng: Cung cấp các chức năng cần thiết phục vụ cho tầng nghiệp vụ kiến trúc n-tầng bao gồm tất cả các ưu điểm của mô hình 3-tầng từ sự mở rộng của các phương pháp 3-tầng. Chủ yếu là hiệu suất được tăng lên do off-load từ lớp cơ sở dữ liệu và tầng client, cho phép nó phù hợp với môi trường của các ngành khối lượng cao Hạn chế: Việc thêm mỗi tầng sẽ phức tạp hơn trong vấn đề xử lý, thực hiện, duy trì và làm tăng giá thành sản phẩm phần mềm.

## **Kiến trúc ba bên (Three-Tired) là gì? Cho ví dụ và phân tích.**

-Kiến trúc ba bên gồm 3 tầng:Presentation Tier,Business Tier,Data Tier là sự tương tác 3 chiều trong môi trường client/server.Giao diện người dùng được lưu trong client,logic ứng dụng doanh nghiệp được lưu trữ trong 1 hoặc nhiều server,các dữ liệu được lưu trữ trong cơ sở dữ liệu máy chủ. -Ví dụ:Để thêm 1 sản phẩm vào giỏ hàng thì người dùng sẽ bấm nút Add To Cart. Người dùng bấm nút Add To Cart để thêm sản phẩm vào giỏ hàng. Tầng Presentation chuyển yêu cầu đến tầng Business là muốn thêm sản phẩm này vào giỏ hàng. Tầng Business nhận yêu cầu và nói tầng Data cập nhật giỏ hàng của người dùng bằng cách thêm vào sản phẩm đã chọn. Tầng Data cập nhật vào CSDL và trả kết quả cho tầng Business. Tầng Business sẽ xử lý kết quả và tất cả lỗi xảy ra trong quá trình cập nhật giỏ hàng. Sau đó, tầng Business sẽ trả kết quả mà nó đã xử lý cho tầng Presentation. Tầng Presentation sẽ tạo giao diện giỏ hàng ở trạng thái đã cập nhật. Phần giao diện sẽ được chuyển thành HTML và trả về trình duyệt của người dùng.

## **Trình bày nguyên lý cài đặt máy chủ phân tán.**

- Máy chủ là máy chạy các tiến trình để cung cấp dịch vụ theo yêu cầu của các máy trạm. Máy chủ thường là những máy tính có cấu hình đủ lớn để luôn sẵn sàng đáp ứng các yêu cầu dịch vụ của máy khách. - Có nhiều hình thức thiết kế hệ thống máy chủ, có thể một máy chủ được tổ chức đơn luồng, đa luồng, đa tiến trình hoặc cụm máy chủ. Với một máy chủ đơn luồng nó phải tiếp nhận yêu cầu và xử lý sau đó trả về kết quả cho máy khách. Máy chủ đa luồng tiếp nhận yêu cầu nhưng không xử lý mà chuyển yêu cầu đó cho luồng khác xử lý. - Đối với các hệ thống tương tranh bắt buộc phải thiết kế dưới dạng đa luồng hoặc đa tiến trình. Một vấn đề quan trọng cần phải giải quyết đó là làm thế nào máy khách biết được điểm truy nhập dịch vụ. Để giải quyết vấn đề này người ta đã qui định trên mỗi máy tính có 65636 cổng, các cổng từ 0-1024 là dành cho những dịch vụ công cộng. Như vậy, mỗi tiến trình cung cấp dịch vụ trên máy chủ sẽ được gán cho một cổng và máy chủ thường xuyên nghe cổng đó. Máy khách muốn sử dụng dịch vụ thì phải cung cấp cặp thông tin địa chỉ máy chủ và cổng dịch vụ. - Tuy nhiên, một số dịch vụ không sử dụng đến cổng, trong trường hợp này máy chủ phải gán điểm truy nhập dịch vụ động và gán cho mỗi hệ điều hành và đồng thời phải có một tiến trình đặc biệt (gọi là daemon) luôn theo dõi điểm truy nhập dịch vụ này. Thông thường mỗi điểm cuối sẽ được gán cho một dịch vụ riêng biệt, nếu cài đặt mỗi dịch vụ lại sử dụng các phương pháp này của máy chủ riêng biệt sẽ lãng phí tài nguyên. Điều này có thể giải quyết bằng cách cung cấp một tiến trình chuyên tiếp nhận yêu cầu của máy khách sau đó chuyển cho tiến trình khác tiếp tục xử lý dịch vụ. - Một vấn đề khác cần phải tính đến khi thiết kế phần mềm trên máy chủ là vấn đề làm thế nào máy chủ có thể ngừng khi chưa hoàn thành yêu cầu xử lý dịch vụ. - Điểm cuối cùng trong thiết kế máy chủ là vấn đề có lưu vết trạng thái hay không? Nếu không lưu giữ thông tin trạng thái của máy khách thì máy chủ có thể tùy ý thay đổi trạng thái của mình mà không cần báo lại cho máy khách. Nếu lưu trạng thái của máy khách thì máy chủ không những không được phép tùy ý thay đổi trạng thái mà còn tăng dung lượng lưu trữ, do đó có thể dung hòa hai cách tiếp cận trên bằng cách lưu trạng thái của máy khách trong một khoảng thời gian nhất định.

## **Các dịch vụ tầng vận tải có phù hợp với việc xây dựng các ứng dụng phân tán hay không? Phân tích vì sao?**

Có, vì tầng vận tải có thực hiện việc ghép kênh, phân kênh cắt hợp dữ liệu (nếu cần). Đóng gói thông điệp, chia thông điệp dài thành nhiều gói tin và gộp các gói nhỏ thành một bộ. Thực tế, việc xây dựng nền tảng cho các hệ thống phân tán vẫn dựa trên mô hình 7 lớp OSI, trong đó 4 lớp thấp (vật lý, liên kết dữ liệu, mạng và vận tải) giải quyết các vấn đề như phát hiện và sửa lỗi, định tuyến , ...

## **Trao đổi thông tin qua mạng luôn có một độ trễ nhất định làm suy giảm hiệu năng của hệ thống, nên các biện pháp khắc phục vấn đề này.**

Các biện pháp khắc phục vấn đề trao đổi thông tin qua mạng luôn có một độ trễ nhất định làm suy giảm hiệu năng của hệ thống : 1.Tạo ra một hệ thống mạng liên kết network link gồm nhiều server có data giống nhau và đồng bộ , những server này được đặt ở nhiều nơi khác nhau , chúng được gọi là cache server hoặc replica server .Điều này nhằm mục đích tiếp cận người dùng nhanh hơn khi người dùng chỉ phải trao đổi dữ liệu tại server gần nhất. 2.Thiết kế mô hình trao đổi dữ liệu hợp lí cho hệ thống. 3.Sử dụng dây mạng tốt hơn. 4.Sử dụng modem/ router tốt hơn. 5.Nâng cấp đường truyền.

## **Nêu những tình huống nên áp dụng máy chủ đơn luồng.**

Các tình huống nên áp dụng máy chủ đơn luồng là:

1. Các dịch vụ đơn giản, yêu cầu xử lý ít, yêu cầu thời gian nhanh.  
Cụ thể:

Máy chủ đơn luồng chạy trên 1 máy cụ thể có thể đáp ứng ngay lập tức yêu cầu. Tuy 1 lúc chỉ có thể đáp ứng 1 yêu cầu nhưng điều đó là đủ cho các hệ thống đơn giản, ít yêu cầu cần xử lý.

#### 2. Các hệ thống nhỏ không cần mở rộng:

Các máy chủ đa luồng đáp ứng được yêu cầu mở rộng của các hệ thống lớn. Nhưng với các hệ thống không yêu cầu mở rộng thì nên áp dụng máy chủ đơn luồng giúp đơn giản và tiết kiệm chi phí.

#### 3. Các hệ thống yêu cầu chi phí nhỏ

Các hệ thống máy chủ đơn luồng sử dụng 1 máy chủ nên chi phí sẽ nhỏ hơn. Với các hệ thống có chi phí đầu tư nhỏ thì cũng có thể sử dụng máy chủ đơn luồng.

#### 4. Hệ thống không yêu cầu tính bảo mật, an toàn cao

Các hệ thống dùng máy chủ đơn luồng khi xảy ra lỗi rất dễ mất mát thông tin nên không dùng cho các ứng dụng đòi hỏi tính an toàn, bảo mật cao như y tế, ngân hàng..

#### 5. Dùng cho các hệ thống tiêu tốn ít tài nguyên, không cần tính toán song song

## **Hệ thống phân tán mở là gì, cho ví dụ.**

Hệ thống phân tán mở là hệ thống có chuẩn giao tiếp với hệ thống, như vậy sẽ dễ dàng cho việc trao đổi tài nguyên. Một hệ thống mở phải tuân thủ một chuẩn giao tiếp nào đó đã được công bố, nghĩa là sản phẩm của các nhà sản xuất khác nhau có thể tương tác với nhau theo các luật và các quy tắc hoặc các tiêu chuẩn đã được công bố. Ví dụ: Hệ thống sử dụng ngôn ngữ IDL, XML, Giao thức dịch vụ WEB...

## **Proxies là gì, nêu ý nghĩa ứng dụng của chúng.**

Proxy Server là gì? Proxy Server là một server đóng vai trò cài đặt proxy làm trung gian giữa người dùng trạm( workstation user) và

Internet. Với Proxy Server, các máy khách( clients) tạo ra các kết nối đến các địa chỉ mạng một cách gián tiếp. Những chương trình client của người sử dụng sẽ qua trung gian proxy server thay thế cho server thật sự mà người sử dụng cần giao tiếp. Proxy server xác định những yêu cầu từ client và quyết định đáp ứng hay không đáp ứng, nếu yêu cầu được đáp ứng, proxy server sẽ kết nối với server thật thay cho client và tiếp tục chuyển tiếp đến những yêu cầu từ client đến server, cũng như đáp ứng những yêu cầu của server đến client. Vì vậy proxy server giống như trung gian giữa server và client. Hiểu một cách đơn giản là : Proxy server là một trung tâm cài đặt các proxy .Mà các proxy này nằm giữa máy tính của bạn và tài nguyên internet (bộ đệm) mà bạn đang truy nhập . Dữ liệu mà bạn yêu cầu đến proxy trước , rồi sau đó nó mới truyền dữ liệu cho bạn và ngược lại. Ý nghĩa ứng dụng của proxy. Proxy không chỉ có giá trị bởi nó làm được nhiệm vụ của một bộ lọc thông tin, nó còn tạo ra được sự an toàn cho các khách hàng của nó, firewall Proxy ngăn chặn hiệu quả sự xâm nhập của các đối tượng không mong muốn vào máy của khách hàng. Proxy lưu trữ được các thông tin mà khách hàng cần trong bộ nhớ, do đó làm giảm thời gian truy tìm làm cho việc sử dụng băng thông hiệu quả. Proxy server giống như một vệ sĩ bảo vệ khỏi những rác rối trên Internet. Một Proxy server thường nằm bên trong tường lửa, giữa trình duyệt web và server thật, làm chức năng tạm giữ những yêu cầu Internet của các máy khách để chúng không giao tiếp trực tiếp Internet. Người dùng sẽ không truy cập được những trang web không cho phép (bị cấm). Mọi yêu cầu của máy khách phải qua Proxy server, nếu địa chỉ IP có trên proxy, nghĩa là website này được lưu trữ cục bộ, trang này sẽ được truy cập mà không cần phải kết nối Internet, nếu không có trên Proxy server và trang này không bị cấm, yêu cầu sẽ được chuyển đến server thật, DNS server... và ra Internet. Proxy server lưu trữ cục bộ các trang web thường truy cập nhất trong bộ đệm để giảm chi phí kết nối,gia tăng tốc độ duyệt web nhanh hơn. Proxy server bảo vệ mạng nội bộ khỏi bị xác định bởi bên ngoài bằng cách mang lại cho mạng hai định danh: một cho nội bộ, một cho bên ngoài. Điều này tạo ra một "bí danh" đối với thế giới bên ngoài và gây khó khăn đối với nếu người dùng "tự tung tự tác" hay các hacker muốn xâm nhập trực tiếp máy tính nào đó.

## **Nêu những điểm khác biệt giữa phân tán dọc và phân tán ngang**

- Phân tán dọc :

+ Các công việc xử lí được thực hiện bằng cách đặt máy tính lớn theo cấu trúc lớp.

+ Các tiến trình xử lý được phân cho các lớp thấp hơn tương ứng với cấu trúc tổ chức và loại nhiệm vụ.

- Phân tán dọc :

+ Bao gồm nhiều máy tính được kết nối ngang hàng vào mạng để xử lí công việc, có thể thêm máy tính nhằm nâng cao độ linh hoạt và nâng cấp hệ thống.

+ Các công việc trước kia được tập trung trên một máy tính thì có thể chia tính toán với các máy tính khác.

+ Có thể sử dụng các thư viện được cung cấp từ máy tính khác, điều này đảm bảo được sự phân tán chức năng và sử dụng chung các nguồn tài nguyên .

## **Sử dụng ngôn ngữ C# viết hàm trao đổi thông điệp giữa các tiến trình bang cách sử dụng winsock.**

<http://hocvienmang.com/website/PrintArticle.aspx?ArticleId=157556>

## **Sử dụng ngôn ngữ C# viết hàm phát hiện tên các thiết bị di động xuất hiện gần máy tính (có thể là bộ phát wifi)**

```
public void showAccessPoint()
{
    WlanClient client = new WlanClient();
```

```

foreach (WlanClient.WlanInterface wlanIface in
client.Interfaces)
{
    // Lists all available networks
    Wlan.WlanAvailableNetwork[] networks =
wlanIface.GetAvailableNetworkList(0);
    foreach (Wlan.WlanAvailableNetwork network in
networks)
    {
        Console.WriteLine("Access Point: {0}",
GetStringForSSID(network.dot11Ssid));
    }
}
}

```

**Sử dụng ngôn ngữ C# viết hàm xác định số dung lượng ổ đĩa trên máy tính, dung lượng đã sử dụng.**

<http://hocvienmang.com/website/PrintArticle.aspx?ArticleId=157554>

**Sử dụng ngôn ngữ Java Script viết hàm xác định số hiệu Serie CPU của máy tính.**

```

<script type="text/javascript">
var cpu_id = "";
var get_cpuid = GetObject("winmgmts:{impersonationLevel=impersonate}");
e = new Enumerator(get_cpuid.InstancesOf("Win32_Processor"));
for(; !e.atEnd(); e.moveNext()) {
    var s = e.item();
    cpu_id = s.ProcessorID;
}
console.log(cpu_id);
</script>

```

## **Hệ thống mở là gì, tại sao một số hệ thống không xây dựng dựa trên kiến trúc mở. Trình bày điểm khác biệt giữa truyền thông có hướng (có liên kết) và truyền thông vô hướng (không liên kết).**

1.

Một hệ thống mở là một hệ thống thường xuyên trao đổi thông tin phản hồi với môi trường bên ngoài của nó. Hệ thống mở là hệ thống của tiến trình, vì vậy đầu vào, quy trình kết quả đầu ra, mục tiêu, đánh giá đều rất quan trọng. Các khía cạnh đó cực kỳ quan trọng hệ thống mở bao gồm ranh giới và môi trường bên ngoài. Hệ thống mở liên tục trao đổi thông tin phản hồi với môi trường của họ, phân tích thông tin phản hồi, điều chỉnh hệ thống nội bộ khi cần thiết để đạt được mục tiêu của hệ thống, và sau đó truyền thông tin cần thiết lại ra môi trường

2.

Một kiến trúc có đặc điểm kỹ thuật công cộng. Điều này bao gồm các tiêu chuẩn chính thức phê duyệt cũng như kiến trúc được thiết kế riêng có thông số kỹ thuật được công bố bởi các nhà thiết kế. Sự đối lập mở là đóng hoặc độc quyền. Ưu điểm lớn nhất của kiến trúc mở là bất cứ ai có thể thiết kế thêm các sản phẩm cho nó. Tuy nhiên, bằng cách làm cho một kiến trúc công cộng, một nhà sản xuất cho phép những người khác để nhân bản sản phẩm của mình. Linux, ví dụ, được coi là kiến trúc mở, vì mã nguồn của nó là có sẵn cho công chúng miễn phí. Ngược lại, hệ điều hành DOS, Windows, và các kiến trúc và hệ điều hành Macintosh đã bị đóng cửa chủ yếu. Nhiều vụ kiện đã được nộp qua việc sử dụng các kiến trúc trong các máy nhân bản. Ví dụ, IBM đã đưa ra một Cease và Trat tự chấm dứt, theo sau là một pin các vụ kiện, khi COMPAQ xây dựng máy tính đầu tiên của mình

3.

+ Kết nối có hướng:

-Tồn tại kênh giao tiếp ảo giữa 2 bên giao tiếp

-Dữ liệu được gửi đi theo chế độ đảm bảo: có kiểm tra lỗi, truyền lại gói tin lỗi hay mất, đảm bảo thứ tự đến của các gói tin -Dữ liệu chính xác tốc độ truyền chậm

+ Kết nối vô hướng:

-Không tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp

-Dữ liệu được gửi đi theo chế độ không đảm bảo: Không kiểm tra lỗi, không phát hiện không truyền lại gói tin bị lỗi hay mất, không đảm bảo thứ tự đến các gói tin

-Dữ liệu không chính xác, tốc độ truyền nhanh . Thích hợp cho các ứng dụng cần tốc độ, không cần chính xác cao

## **Sử dụng ngôn ngữ C# viết hàm đọc trạng thái của tập tin (tập tin có đang được mở bởi tiến trình nào đó hay không)**

```
public bool FileIsLocked(string strFullName) { bool blnReturn =  
false; System.IO.FileStream fs; try { fs =  
System.IO.File.Open(strFullName, FileMode.OpenOrCreate,  
FileAccess.Read, FileShare.None); fs.Close(); } catch  
(System.IO.IOException ex) { blnReturn = true; } return blnReturn;  
}
```

## **Sử dụng ngôn ngữ Java Script viết hàm đếm số lượng lần sử dụng phím xóa Delete (Xóa ký tự liền sau con trỏ)**

```
//var count = 0;//initial a variable for counting  
//document.addEventListener("keydown", KeyCheck); //add  
function for counting  
function KeyCheck(event) {//this is the function for counting how  
many times that backspace was clicked  
var KeyID = event.keyCode;  
switch (KeyID) {  
case 46://delete key count++;  
break; default: break;
```

```
    }
    console.log(count);//output the count after backspace is clicked
}
```

## Trình bày nguyên lý hoạt động của tiến trình máy ảo.

Nguyên lý hoạt động của tiến trình máy ảo (Process Virtual Machine): Tiến trình máy ảo xây dựng hệ thống thời gian chạy cung cấp tập lệnh trùu tượng cho phần mềm ứng dụng, các lệnh này sẽ được dịch (ví dụ môi trường thời gian chạy Java) hoặc cũng có thể được mô phỏng như các lời gọi hệ thống (ví dụ chạy trên nền tảng hệ điều hành Windows hoặc UNIX). Hình thức tiến trình máy ảo này chủ yếu áp dụng cho một tiến trình.

## Trình bày nguyên lý cài đặt máy chủ phân tán.

- Máy chủ là máy chạy các tiến trình để cung cấp dịch vụ theo yêu cầu của các máy trạm. Máy chủ thường là những máy tính có cấu hình đủ lớn để luôn sẵn sàng đáp ứng các yêu cầu dịch vụ của máy khách. - Có nhiều hình thức thiết kế hệ thống máy chủ, có thể một máy chủ được tổ chức đơn luồng, đa luồng, đa tiến trình hoặc cụm máy chủ. Với một máy chủ đơn luồng nó phải tiếp nhận yêu cầu và xử lý sau đó trả về kết quả cho máy khách. Máy chủ đa luồng tiếp nhận yêu cầu nhưng không xử lý mà chuyển yêu cầu đó cho luồng khác xử lý. - Đối với các hệ thống tương tranh bắt buộc phải thiết kế dưới dạng đa luồng hoặc đa tiến trình. Một vấn đề quan trọng cần phải giải quyết đó là làm thế nào máy khách biết được điểm truy nhập dịch vụ. Để giải quyết vấn đề này người ta đã qui định trên mỗi máy tính có 65636 cổng, các cổng từ 0-1024 là dành cho những dịch vụ công cộng. Như vậy, mỗi tiến trình cung cấp dịch vụ trên máy chủ sẽ được gán cho một cổng và máy chủ thường xuyên nghe cổng

đó. Máy khách muốn sử dụng dịch vụ thì phải cung cấp cặp thông tin địa chỉ máy chủ và cổng dịch vụ. - Tuy nhiên, một số dịch vụ không sử dụng đến cổng, trong trường hợp này máy chủ phải gán điểm truy nhập dịch vụ động và gán cho mỗi hệ điều hành và đồng thời phải có một tiến trình đặc biệt (gọi là daemon) luôn theo dõi điểm truy nhập dịch vụ này. Thông thường mỗi điểm cuối sẽ được gán cho một dịch vụ riêng biệt, nếu cài đặt mỗi dịch vụ lại sử dụng các phương điệu này của máy chủ riêng biệt sẽ lãng phí tài nguyên. Điều này có thể giải quyết bằng cách cung cấp một tiến trình chuyên tiếp nhận yêu cầu của máy khách sau đó chuyển cho tiến trình khác tiếp tục xử lý dịch vụ. - Một vấn đề khác cần phải tính đến khi thiết kế phần mềm trên máy chủ là vấn đề làm thế nào máy chủ có thể ngừng khi chưa hoàn thành yêu cầu xử lý dịch vụ. - Điểm cuối cùng trong thiết kế máy chủ là vấn đề có lưu vết trạng thái hay không? Nếu không lưu giữ thông tin trạng thái của máy khách thì máy chủ có thể tùy ý thay đổi trạng thái của mình mà không cần báo lại cho máy khách. Nếu lưu trạng thái của máy khách thì máy chủ không những không được phép tùy ý thay đổi trạng thái mà còn tăng dung lượng lưu trữ, do đó có thể dung hòa hai cách tiếp cận trên bằng cách lưu trạng thái của máy khách trong một khoảng thời gian nhất định.

## Trình bày các loại bản ghi trong dịch vụ phân giải tên miền (DNS)

Một tên miền được chia ra thành nhiều cấp bậc:

**gTLD (Generic Top Level Domain)** – tên miền kết thúc bằng .com, .net, .org, .gov, .edu và .name.

**ccTLD (Country Code Top Level Domain)** – tên miền riêng của mỗi quốc gia (ví dụ .il, .ru, .uk, .us, .cn, .vn).

**Infrastructure TLD** – Xử lý và điều hướng chuyển đổi từ địa chỉ IP sang tên miền, Ví dụ IPv4 48.199.81.in-addr.arpa. Sử dụng trong bản ghi PTR (được giải thích bên dưới).

**Cấp 2Id, 3Id, 4Id** – Ví dụ www.somedomain.co.uk, uk là ccTLD, co là 2Id, somedomain là 3Id và www là 4Id.

### Các bản ghi DNS

Các bản ghi DNS được phân loại như sau:

- A

Bản ghi A xác định địa chỉ IPv4 hoặc IPv6 của tài nguyên mạng. Ví dụ bản ghi:  
www IN A 12.34.56.78  
trong tên miền somedomain.co.uk định nghĩa www.somedomain.co.uk được truy cập duy nhất tại địa chỉ IPv4 12.34.56.78.

- **CNAME**

Bản ghi CNAME (canonical name) xác định một định danh khác hay còn gọi là "bí danh" (alias) sẽ được phân giải đến khi tìm thấy bản ghi A. Ví dụ bản ghi:  
www IN CNAME www.somedomain.co.uk  
trong tên miền somedomain.com định nghĩa định danh duy nhất  
www.somedomain.com là bí danh cho www.somedomain.co.uk.

- **MX**

## Trình bày nguyên lý hoạt động của máy điều hành ảo (Virtual Machine Monitor)

### **Trình bày nguyên lý hoạt động của máy điều hành ảo (Virtual Machine Monitor)**

Máy điều hành ảo (Virtual Machine Monitor) dựa trên kiến trúc phân tầng, nó che đậy hoàn toàn phần cứng nhưng cung cấp đầy đủ các chỉ thị lệnh của phần cứng. Điểm quan trọng của hình thức này là giao diện của nó cung cấp đồng thời cho nhiều chương trình khác nhau, điều đó dẫn tới việc nhiều hệ điều hành có thể cùng chạy trên một nền tảng phần cứng. Như vậy, hệ điều hành giao tiếp với phần cứng qua lớp máy ảo.

Máy điều hành ảo cho phép cài đặt một ứng dụng và môi trường của nó hoàn toàn. Máy điều hành ảo cũng cung cấp sự phân tách giữa phần cứng và phần mềm, cho phép một môi trường có thể được di chuyển từ máy này sang máy khác.

## Trình bày nguyên lý chia sẻ kênh (bus) trong các máy tính nhiều bộ vi xử lý (multiprocessor).

Khi một hệ thống có nhiều CPU cùng kết nối với bus và chia sẻ chung cùng với một bộ nhớ. Khi bất kì CPU nào muốn đọc từ bộ nhớ ra, nó đưa địa chỉ vào bus và đưa tín hiệu cho bus điều khiển rằng nó muốn đọc. Khi yêu cầu được chấp nhận và địa chỉ được tìm kiếm xong trong bộ nhớ, CPU yêu cầu một lần nữa với bus điều khiển để xác nhận sẵn sàng, khi đó CPU bắt đầu thực hiện việc đọc từ bộ nhớ ra. Cách ghi vào bộ nhớ cũng tương tự. Để ngăn chặn hai hay nhiều CPU cố gắng truy cập vào bộ nhớ cùng một lúc, thì cần đến "trọng tài" phân quyền truy xuất, đó là một loại phân quyền. Ví dụ: Để có được bus, CPU phải yêu cầu được phép sử dụng bus bằng một lệnh đặc biệt, khi yêu cầu được cấp phép, lúc này CPU mới được sử dụng bus. Cách cấp phép có nhiều loại, thứ tự yêu tiên khác nhau tùy thuộc vào bộ xử lý, có thể CPU nào yêu cầu trước thì sẽ được cấp phép trước, hoặc CPU nào có quyền ưu tiên cao sẽ được cấp trước, v.v.. Tuy nhiên, việc chỉ có một bus kết nối với nhiều bộ xử lý dễ dẫn tới tình trạng "quá tải" (overloaded). Khi các bộ xử lý gửi yêu cầu quá nhiều mà không được đáp ứng, một cách để giải quyết là mỗi bộ xử lý sẽ có một bộ nhớ cache, khi đó, các vùng nhớ được sử dụng lại sẽ lấy từ cache thay vì bộ nhớ, từ đó sẽ không cần yêu cầu bus, giảm thiểu số lần sử dụng bus, tránh gây quá tải.

## Trình bày các mô hình cung cấp dịch vụ tập tin (File Service

Các mô hình cung cấp dịch vụ tập tin:

- Truyền tập tin: không có mạng, các khả năng truyền tải tập tin giữa các máy tính bị hạn chế
- Lưu trữ tập tin: phần lớn các dữ liệu quan trọng trên mạng đều được lưu trữ tập trung theo nhiều cách khác nhau

- Lưu trữ trực tuyến: dữ liệu được lưu trữ trên đĩa cứng nên truy suất dễ dàng, nhanh chóng, bất kể thời gian
- Lưu trữ ngoại tuyến: thường áp dụng cho dữ liệu ít khi cần truy suất. Các thiết bị phổ biến dùng là đĩa từ, đĩa quang
- Lưu trữ ngoại tuyến: giúp ta khắc phục được tình trạng truy suất chậm của phương pháp lưu trữ ngoại tuyến
- Di trú dữ liệu: là công nghệ tự động dời các dữ liệu ít dùng từ kho lưu trữ trực tuyến sang kho lưu trữ cận tuyến hay ngoại tuyến
- Đồng bộ hóa việc cập nhật tập tin

## Nêu vai trò của các biện pháp đảm bảo an toàn thông tin trong các hệ thống phân tán.

Bảo mật là một trong những vấn đề quan trọng nhất của hệ thống phân tán để tránh hoặc hạn chế các thiệt hại do các cuộc tấn công, đảm bảo tính bí mật toàn vẹn và khả năng sẵn sàng cao cho hệ thống. Khi dữ liệu được phân tán trên nhiều mạng hay thông tin được chuyển giao thông qua mạng công cộng, nó trở thành các đối tượng tấn công bởi các phần tử có hại. Các tài nguyên máy tính hay các thiết bị lưu trữ cũng có thể bị tấn công bởi tin tặc. Bảo mật trong hệ thống phân tán có thể chia làm 2 nhóm chính: bảo mật trong quá trình trao đổi thông tin và bảo mật việc lưu trữ thông tin tại các kho dữ liệu. Biện pháp để đảm bảo an toàn thông tin trong hệ thống -

Đảm bảo tính bí mật của dữ liệu: chỉ những người dùng hợp pháp mới có thể truy cập thông tin và thông tin đó phải đảm bảo cơ chế đảm bảo an toàn, tránh truy cập của người dùng bất hợp pháp. - Các kênh bảo mật + Xây dựng tường lửa: Tường lửa có 2 dạng Tường lửa bảo vệ toàn bộ mạng và tường lửa bảo vệ riêng từng máy. Tường lửa được xây dựng dựa trên cấu trúc các gói dữ liệu theo mô hình phân lớp mạng, kết quả của tường lửa là kiểm soát truy nhập. + Xây

dụng mạng riêng ảo: Mạng riêng ảo là một kênh truyền bảo mật trên môi trường mạng công cộng. nhằm tiết kiệm về chi phí so với việc sử dụng kênh truyền riêng. VPN sử dụng các giao thức "đường hầm" cho phép đảm bảo tính toàn vẹn, bí mật thông tin, xác thực và mã hóa dữ liệu truyền trên kênh. Nhược điểm của VPN là tốc độ, không vượt quá tốc độ của đường truyền internet. +Hệ thống phát hiện và ngăn chặn đột nhập: là khả năng phát hiện, nhận dạng mã độc tấn công vào mạng hoặc một máy tính nào đó +Xác thực truy nhập: xác thực thông tin của người dùng dựa vào thông tin mà họ có, như mật khẩu hay chứng chỉ số khi đăng nhập vào hệ thống. - Mã hóa dữ liệu: nhằm bảo vệ thông tin khi lưu chuyển trên mạng hay truy nhập vào kho dữ liệu có thể sử dụng một số thuật toán mã hóa thông tin như DES, RSA, MD5...

## **Hệ thống phân tán là gì, hãy liệt kê một số hệ thống phân tán.**

Hệ thống phân tán là nhóm các máy tính(bao gồm các thiết bị khác như điện thoại di động, PDA ...) độc lập nhưng hoạt động gắn kết với nhau như một thể thống nhất. Một số hệ thống phân tán: - Hệ thống quản lý sinh viên - Hệ thống ngân hàng - Hệ thống World Wide Web - Hệ thống bán vé qua mạng

## **Tính linh hoạt của hệ thống thể hiện ở những điểm nào?**

Tính linh hoạt của hệ thống thể hiện ở những điểm là:

+Tính chia sẻ tài nguyên:

Việc chia sẻ tài nguyên trên hệ phân tán - trong đó tài nguyên bị lệ thuộc về mặt vật lý với một máy tính nào đó - được thực hiện thông qua truyền thông. Để chia sẻ tài nguyên một cách hiệu quả thì mỗi tài nguyên cần phải được quản lý bởi một chương trình có giao diện truyền thông, các tài nguyên có thể truy nhập, cập nhật được một cách tin cậy và nhất quán. Quản lý tài nguyên ở đây bao gồm lập kế hoạch và dự phòng, đặt tên các lớp tài nguyên,

cho phép tài nguyên được truy cập từ nơi khác, ánh xạ tên tài nguyên vào địa chỉ truyền thông.

#### +Tính mở

Tính mở của một hệ thống máy tính là tính dễ dàng mở rộng phần cứng (thiết bị ngoại vi, bộ nhớ, các giao diện truyền thông ...) và phần mềm (các mô hình HĐH, các giao thức truyền thông, các dịch vụ chia sẻ tài nguyên ...) của nó. Nói một cách khác, tính mở của hệ thống phân tán mang ý nghĩa bao hàm tính dễ dàng cấu hình cả phần cứng lẫn phần mềm của nó.

#### +Khả năng song song

Khả năng làm việc song song trong hệ phân tán được thi hành do hai tình huống:Nhiều người sử dụng đồng thời đưa ra các lệnh hay tương tác với chương trình ứng dụng (đồng thời xuất hiện nhiều QT khách). Nhiều QT phục vụ chạy đồng thời, mỗi QT đáp ứng yêu cầu của một trong số các QT Khách.

#### +Khả năng mở rộng

Khả năng mở rộng của một hệ phân tán được đặc trưng bởi tính không thay đổi phần mềm hệ thống và phần mềm ứng dụng *khi hệ thống được mở rộng*.

#### +Khả năng thứ lỗi

Hệ phân tán cung cấp khả năng sẵn sàng cao để đối phó với các sai hỏng phần cứng. Khả năng sẵn sàng của hệ thống được đo bằng tỷ lệ thời gian mà hệ thống sẵn sàng làm việc so với thời gian có sự cố. Khi một máy trên mạng sai hỏng thì chỉ có công việc liên quan đến các thành phần sai hỏng bị ảnh hưởng. Người sử dụng có thể chuyển đến một trạm khác nếu máy họ đang sử dụng bị hỏng, một QT phục vụ có thể được khởi động lại trên một máy khác.

## Nêu vai trò và ứng dụng của lớp trung gian (Middleware) trong hệ thống phân tán

Vai trò:

Phần mềm lớp trung gian (middleware) sẽ che giấu sự không đồng nhất của các hệ thống ở

## Nêu những hạn chế của việc đặt thời gian quá hạn (timeout) sử dụng trong việc truyền tin, trình bày biện pháp khắc phục

### Những hạn chế:

- Khi sử dụng time-out, nếu xảy ra mất gói tin gửi, bên gửi sẽ phải gửi lại các gói tin khi hết thời gian timeout. Việc đó có nghĩa bộ đệm bên gửi luôn phải giữ lại mà thông thường cần được giải phóng để tránh tắc nghẽn.
- Nếu thời gian time-out quá ngắn, các gói bị gửi lại một cách liên tục không cần thiết sẽ lại gây tắc nghẽn. Nếu quá dài sẽ vượt quá thời gian trễ. Việc gửi lại quá nhiều sẽ gây lãng phí dải thông.

### Để khắc phục:

- + Hủy bỏ gói dữ liệu nếu cần.
- + Sử dụng phương pháp điều khiển lưu lượng.
- + Giảm đầu vào khi có xảy ra xung đột.
- + Ta có thể hủy bỏ các gói không được chuyển đi xa một cách có hệ thống.
- + Giải pháp sau cùng là nâng cấp bộ đệm.

## Liệt kê và phân tích những nguyên nhân gây lỗi trong các hệ thống phân tán

- Lỗi sụp đổ xảy ra khi một server ngừng hoạt động trước dự kiến, nhưng vẫn làm việc chính xác cho đến khi nó dừng. Một ví dụ điển hình của trường hợp này là khi hệ điều hành gặp phải một lỗi nghiêm trọng, và chỉ có một giải pháp duy nhất là khởi động lại. - Lỗi bỏ sót xảy ra khi server không có khả năng nhận hay đáp ứng đầy đủ các yêu cầu, đa phần là do không thể kết nối được với nhau. Trong trường hợp lỗi nhận các yêu cầu, có khả năng server không bao giờ nhận được yêu cầu ngay trong lần đầu tiên. Kết nối giữa client và server có thể đã được tạo ra nhưng không có luồng lắng nghe các request gửi đến. Lỗi gửi các bản tin xảy ra khi server đã hoàn thành công việc của nó, nhưng vì một lý do nào đó mà không thể gửi được phản hồi. Một loại khác không liên quan đến kết nối có thể gây ra bởi các lỗi phần mềm như vòng lặp vô tận hoặc quản lý bộ nhớ không hợp lý dẫn đến server bị treo. - Lỗi thời gian xảy ra khi bản tin phản hồi được gửi đi trong một khoảng thời gian không thích hợp, như gửi dữ liệu quá sớm có thể dễ dàng gây ra rắc rối cho phía nhận nếu bên nhận không đủ bộ nhớ đệm để lưu giữ tất cả các dữ liệu đến. Tuy nhiên thực tế thường chỉ xảy ra ở mức server phản hồi quá chậm dẫn đến giảm hiệu năng của hệ thống. - Lỗi đáp ứng

xảy ra khi các bản tin phản hồi của server không thích hợp. Có 2 loại lỗi phản hồi có thể xảy ra là value failure và state transition failure. Value failure là khi server cung cấp các phản hồi sai cho một yêu cầu nào đó. Chẳng hạn một search engine đưa ra kết quả tìm kiếm các trang web không liên quan gì tới các từ khóa tìm kiếm. State transition failure xảy ra nếu không có tiêu chuẩn nào được đưa ra để điều khiển các bản tin. Cụ thể là trong trường hợp một server lỗi có thể có những quyết định mặc định không hợp lý. - Lỗi phức hợp có thể xảy ra khi server tạo những đầu ra mà khi bình thường nó không bao giờ tạo ra, sau đó kết hợp với những server khác để tại ra những câu trả lời sai, có quan hệ chặt chẽ với lỗi sụp đổ. Trong thực tế, khi xảy ra server sẽ ngừng tạo ra những đầu ra, nhờ đó mà các tiến trình khác có thể nhận thấy được sự ngừng hoạt động của nó. Trong trường hợp tốt nhất, server có thể thông báo rằng nó sắp ngừng hoạt động.

## Nêu những điểm khác biệt giữa khái niệm liên kết lỏng và liên kết chặt trong các hệ thống phân tán

- 1) Liên kết lỏng (tính toán phân tán): mỗi vi xử lý làm chủ bộ nhớ của riêng mình, chạy hệ điều hành của riêng mình, không giao tiếp thường xuyên. - Những hệ thống đầu tiên sử dụng chương trình mạng đơn giản và minh bạch • FTP (rcp): giao thức truyền file • telnet (rlogin/rsh): giao thức đăng nhập từ xa • mail (SMTP) - Mỗi hệ thống là một hệ thống hoàn toàn độc lập và tự trị, được kết nối với những hệ thống khác trên mạng. - Ngày nay, hầu hết các hệ thống phân tán đều liên kết lỏng. • Mỗi CPU chạy một hệ điều hành độc lập và tự chủ • Các máy tính không thật sự tin tưởng lẫn nhau • Một số tài nguyên được chia sẻ nhưng hầu như là không có • Hệ thống trông sẽ khác nhau từ những hướng khác nhau 2) Liên kết chặt (xử lý song song): các bộ vi xử lý chia sẻ đồng hồ và dữ liệu, chạy một hệ điều hành, giao tiếp thường xuyên. - Một hệ thống liên kết chặt thường dùng để chỉ một bộ đa xử lý • Hệ điều hành chỉ thực hiện 1 công việc trong 1 thời gian theo thứ tự hàng đợi • Có một không gian địa chỉ duy nhất • Thông thường có một đường bus đơn

hoặc bảng mạch mè làm cho vi xử lý và bộ nhớ được kết nối • Có độ trễ truyền thông thấp • Bộ xử lý giao tiếp thông qua bộ nhớ được chia sẻ

## Nêu những ưu điểm và nhược điểm của hệ thống phân tán so với hệ thống tập trung.

**Ưu điểm:** +Trong hệ thống phân tán việc giảm dư thừa dữ liệu phức tạp và hiệu quả hơn hệ thống tập trung. +Tính bảo mật cao hơn. Do: Nếu một vị trí trong hệ thống phân tán bị hỏng, các vị trí khác vẫn tiếp tục làm việc. +Tốc độ tính toán cao hơn: Hệ thống phân tán cho phép phân chia việc tính toán trên nhiều vị trí khác nhau để tính toán song song. + Hiệu năng hơn: khắc phục được khó khăn do sự chậm trễ thông tin liên lạc. +Chia sẻ tài nguyên :chia sẻ tài nguyên trong hệ thống phân tán cung cấp một cơ chế để chia sẻ tập tin ở vị trí xa + Tính độc lập dữ liệu cao hơn do có tính trong suốt phân tán. **Nhược điểm:** + Khả năng điều khiển tập trung trên toàn nguồn tài nguyên hệ thống kém hơn trong hệ thống tập trung

## Phân tích quá trình phân giải tên miền mail.yahoo.com thành địa chỉ IP

Việc phân giải tên miền được thực hiện nhờ hệ thống tên miền DNS(Domain Name System). Client sẽ gửi yêu cầu cần phân giải địa chỉ IP cần phân giải cho sever. sever sẽ phân tích tên miền xem có do mình quản lý hay không. Nếu thuộc nó quản lý nó sẽ trả về địa chỉ IP tương ứng. Trong trường hợp trên tên miền là mail.yahoo.com là subdomain của tên miền yahoo.com. Trong hệ thống IP thì mỗi một quốc gia có một đầu mã IP khác nhau, trong quốc gia đó lại chia thành những vùng khác nhau, phân cấp dần về vị trí. Vì thế khi một tên miền được sử dụng hệ thống tên miền sẽ phân tích vị trí và tìm ra địa chỉ IP tương ứng.

## **Trình bày những điểm khác biệt cơ bản giữa hệ điều hành phân tán và hệ điều hành mạng.**

Những điểm khác biệt cơ bản giữa hệ điều hành phân tán và hệ điều hành mạng:

- Về mặt kiến trúc:

+ Hệ điều hành phân tán là hệ thống quản lý tài nguyên cho một nhóm các máy tính độc lập nhưng hoạt động gắn kết chặt chẽ với nhau như một thể thống nhất.

+ Hệ điều hành mạng là hệ thống quản lý việc chia sẻ tài nguyên và truy cập tài nguyên chia sẻ từ máy tính khác thông qua mạng.

- Về tiêu chí:

+ Hệ điều hành phân tán có tính trong suốt: là sự che giấu tài nguyên vật lý và các tiến trình xử lý được phân tán trên nhiều máy tính khác nhau nhưng người dùng cảm giác chỉ giao tiếp với một máy tính.

+ Đối với hệ điều hành mạng thì cần gọi thủ tục từ xa, không có sự điều khiển phần cứng hoặc phần mềm trực tiếp từ máy tính này đến máy tính khác trong hệ thống.

- Về mặt xử lý:

+ Trong hệ điều hành phân tán, các tiến trình được phân phối trên nhiều máy tính khác nhau nên cần phải điều phối, đồng bộ các tiến trình, xử lý tương tranh...

+ Hệ điều hành mạng cần quản lý việc chia sẻ tài nguyên dùng chung.

## **Trình bày qui trình hoạt động khi gọi thủ tục được cài đặt trên máy tính khác.**

Để gọi thủ tục được cài trên máy tính khác, qui trình hoạt động như sau:

- Ta sẽ xem máy tính cục bộ là Client và máy tính khác sẽ là Server

- Client sẽ khởi tạo lời gọi hàm từ xa, lời gọi hàm này sẽ kích hoạt một thủ tục cục bộ nằm ở phần Stub
- ClientStub cung cấp một các hàm cục bộ mà Client có thể gọi, mỗi hàm này sẽ đại diện cho một hàm nằm ở Server.
- Khi một hàm nào đó ở ClientStub được gọi, ClientStub sẽ đóng gói một thông điệp để mô tả về thủ tục mà Client muốn thực hiện cùng với tham số(nếu có). Sau đó sẽ thông qua hệ thống RPCRuntime cục bộ gửi đến ServerStub của Server.
- RPCRuntime là phần quản lý việc truyền thông điệp qua mạng giữa Client và Server, nó đảm nhận việc truyền lại, báo nhận, chọn đường gói tin và mã hoá thông tin, RPCRuntime có thể được cung cấp bởi hệ thống.
- RPCRuntime của Client sẽ gọi lệnh send() để gửi thông điệp đến RPCRuntime của Server, sau đó gọi lệnh wait() để chờ kết quả trả về từ Server.
- RPCRuntime của Server sau khi nhận được thông điệp sẽ chuyển nó đến ServerStub.
- ServerStub đọc thông điệp và xác định thủ tục mà Client muốn gọi cùng tham số của nó. Sau đó ServerStub sẽ gọi một thủ tục tương ứng nằm ở Server.
- Server thực hiện thủ tục được gọi từ ServerStub và gửi kết quả thực thi được cho ServerStub.
- ServerStub đóng gói kết quả vào một gói tin trả lời và chuyển cho RPCRuntime ở Server để gửi trả về RPCRuntime của Client.
- RPCRuntime của Client sau khi nhận được gói tin trả về, gửi nó cho ClientStub, ClientStub mở thông điệp chứa kết quả thực thi cho Client tại vị trí phát ra lời gọi thủ tục từ xa.

## Trình bày giao thức LDAP.

LDAP là viết tắt của Lightweight Directory Access Protocol, hay dịch ra tiếng Việt có nghĩa là giao thức truy cập nhanh các dịch vụ thư mục. - Là một giao thức tìm, truy nhập các thông tin dạng thư

mục trên server. - Nó là giao thức dạng Client/Server dùng để truy cập dịch vụ thư mục. - LDAP chạy trên TCP/IP hoặc các dịch vụ hướng kết nối khác. - Là một mô hình thông tin cho phép xác định cấu trúc và đặc điểm của thông tin trong thư mục. - Là một không gian tên cho phép xác định cách các thông tin được tham chiếu và tổ chức - Một mô hình các thao tác cho phép xác định các tham chiếu và phân bố dữ liệu. - Là một giao thức mở rộng - Là một mô hình thông tin mở rộng. 2- Phương thức hoạt động của LDAP - Ldap dùng giao thức giao tiếp client/sever + Giao thức giao tiếp client/sever là một mô hình giao thức giữa một chương trình client chạy trên một máy tính gửi một yêu cầu qua mạng đến cho một máy tính khác đang chạy một chương trình sever (phục vụ). Chương trình sever này nhận lấy yêu cầu và thực hiện sau đó trả lại kết quả cho chương trình client + Ý tưởng cơ bản của giao thức client/server là công việc được gán cho những máy tính đã được tối ưu hoá để thực hiện công việc đó. + Một máy server LDAP cần có rất nhiều RAM(bộ nhớ) dùng để lưu trữ nội dung các thư mục cho các thao tác thực thi nhanh và máy này cũng cần đĩa cứng và các bộ vi xử lý ở tốc độ cao. + Client mở một kết nối TCP đến LDAP server và thực hiện một thao tác bind. Thao tác bind bao gồm tên của một directory entry , và uỷ nhiệm thư sẽ được sử dụng trong quá trình xác thực, uỷ nhiệm thư thông thường là password nhưng cũng có thể là chứng chỉ điện tử dùng để xác thực client. + Sau khi thư mục có được sự xác định của thao tác bind, kết quả của thao tác bind được trả về cho client. Client phát ra các yêu cầu tìm kiếm. + Server thực hiện xử lý và trả về kết quả cho client. + Server gửi thông điệp kết thúc việc tìm kiếm. + Client phát ra yêu cầu unbind, với yêu cầu này server biết rằng client muốn huỷ bỏ kết nối. + Server đóng kết nối. - LDAP là một giao thức hướng thông điệp + Do client và sever giao tiếp thông qua các thông điệp, Client tạo một thông điệp(LDAP message) chứa yêu cầu và gửi nó đến cho server. Server nhận được thông điệp và xử lý yêu cầu của client sau đó gửi trả cho client cũng bằng một thông điệp LDAP. Ví dụ: Khi LDAP client muốn tìm kiếm trên thư mục, client tạo LDAP tìm kiếm và gửi thông điệp cho server. Sever tìm trong cơ sở dữ liệu và gửi kết quả cho client trong một thông điệp LDAP + Nếu client tìm kiếm thư mục và nhiều kết quả được tìm thấy, thì các kết quả này được gửi đến client bằng nhiều thông điệp + Do nature LDAP là giao thức hướng thông điệp nên client được phép phát ra nhiều thông điệp yêu cầu đồng thời cùng một lúc. Trong LDAP, message ID dùng để phân biệt các yêu cầu của client và kết quả trả về của server. + Việc cho phép nhiều thông điệp cùng

xử lý đồng thời làm cho LDAP linh động hơn các nghi thức khác. + Ví dụ như HTTP, với mỗi yêu cầu từ client phải được trả lời trước khi một yêu cầu khác được gửi đi, một HTTP client program như là Web browser muốn tải xuống cùng lúc nhiều file thì Web browser phải thực hiện mở từng kết nối cho từng file, LDAP thực hiện theo cách hoàn toàn khác, quản lý tất cả thao tác trên một kết nối. 3- Làm việc với LDAP thông qua PHP Trình tự cơ bản khi có thao tác với LDAP gồm các bước - Connect (kết nối với LDAP) - Bind (kiểu kết nối: nặc danh hoặc đăng nhập xác thực) - Search (tìm kiếm) - Interpret search (xử lý tìm kiếm) - Result (kết quả) - Unblind (hủy kết nối) - Close connection (đóng kết nối)

#### \* Mô hình LDAP

LDAP LDAP còn định nghĩa ra 4 mô hình, các mô hình này cho phép linh động trong việc sắp xếp thư mục

+ Mô hình LDAP information: xác định cấu trúc, đặc điểm của thông tin trong thư mục

+ Mô hình LDAP Naming: xác định các cách thông tin được

## Phân tích qui trình hoạt động của kiến trúc ngang hàng khi xây dựng hệ thống phân tán.

Kiến trúc phân tán ngang bao gồm nhiều máy tính được kết nối ngang hàng vào mạng để xử lý công việc, có thể thêm máy tính nhằm nâng cao độ linh hoạt và nâng cấp hệ thống. Các công việc trước kia được tập trung trên một máy tính thì có thể chia tính toán với các máy tính khác. Có thể sử dụng các thư viện được cung cấp từ các máy tính khác, điều này đảm bảo được sự phân tán chức năng và sử dụng chung các nguồn tài nguyên.

## Viết ứng dụng bằng ngôn ngữ C# minh họa truyền thông điệp sử dụng hàng đợi

```
using System;
using System.Messaging;

namespace MyProject
{
    public class MyNewQueue
    {
        public static void Main()
        {
            MyNewQueue myNewQueue = new MyNewQueue();

            myNewQueue.SendMessage();

            return;
        }

        public void SendMessage()
        {
            MessageQueue myQueue = new MessageQueue(".\\myQueue");

            if(myQueue.Transactional == true)
            {
                MessageQueueTransaction myTransaction =
                    new MessageQueueTransaction();
                myTransaction.Begin();

                myQueue.Send("This is my message.", myTransaction);

                myTransaction.Commit();
            }
            else
            {
                myQueue.Send("This is my message.");
            }
        }

        return;
    }
}
```

```
        }  
    }  
}
```

## Phân tích các chỉ số đảm bảo chất lượng truyền dữ liệu dạng luồng (stream).

Đối với việc truyền dữ liệu dạng luồng cần phải đáp ứng về mặt thời gian cũng như tính liên tục của dữ liệu cần phải chuyển. Đối với việc truyền tin phụ thuộc thời gian, yêu cầu quan trọng là phải đáp ứng tiêu chuẩn về chất lượng truyền tin, đó là các thông số mà hệ thống phân tán cần phải đáp ứng để đảm bảo chất lượng dịch vụ. Các dịch vụ đa phương tiện cần phải được đồng bộ giữa các luồng dữ liệu đó và cần duy trì mối tương quan giữa các luồng dữ liệu.

## Trình bày qui trình viết ứng dụng khách/chủ

Qui trình viết ứng dụng khách/chủ:

- Bước 1: Gọi chương trình uidgen để tạo ra một tập tin mẫu IDL (Interface Definition Language - ngôn ngữ định nghĩa giao diện) chứa một định danh duy nhất (Universal Unique Identifier). uidgen sẽ không bao giờ tạo ra một định danh trùng lặp ở bất cứ đâu. Tính duy nhất của định danh này được đảm bảo bằng cách mã hóa thời gian và địa điểm tạo ra bên trong nó.
- Bước 2: Chỉnh sửa tập tin IDL, điền tên của các thủ tục từ xa và các tham số của nó. Đáng chú ý là RPC (gọi thủ tục từ xa) không hoàn toàn trong suốt, ví dụ: máy khách và máy chủ không thể chia sẻ các biến toàn cục tuy nhiên các quy tắc IDL sẽ ngăn cản việc sử dụng các cấu trúc không được hỗ trợ (không tồn tại). Khi tập tin IDL hoàn thiện, nó sẽ được xử lý bởi trình biên dịch IDL. Kết quả

của quá trình này gồm ba tập tin: tập tin tiêu đề, một thành phần trên máy khách (gọi là stub) và một thành phần tương ứng trên máy chủ (gọi là skeleton). Tập tin tiêu đề chứa định danh, định nghĩa kiểu, định nghĩa hằng và các nguyên mẫu hàm. Nó sẽ được "#include" trong đoạn code ở cả máy khách và máy chủ. Stub gồm các thủ tục mà máy khách sẽ gọi. Nó chịu trách nhiệm thu thập, đóng gói các tham số vào thông điệp và gọi hệ thống thời gian chạy (runtime) gửi đi cũng như nhận kết quả trả về từ máy chủ. Skeleton gồm các thủ tục sẽ được gọi khi thông điệp từ máy khách tới. Chúng sẽ lần lượt gọi tới các thủ tục trên máy chủ để thực hiện yêu cầu nhận được. - Bước 3: Lập trình viên xây dựng code trên máy chủ và máy khách. Sau đó các đoạn code được biên dịch. Code trên máy khách sẽ kết nối với stub và trên máy chủ kết nối với skeleton. Máy khách và máy chủ khởi chạy đồng nghĩa với ứng dụng được thực thi.

## Trình bày cơ chế đảm bảo tính trong suốt khi gọi thủ tục từ xa (RPC)

- Lời gọi thủ tục từ xa là một cơ chế cho phép một chương trình có thể gọi thực thi một thủ tục (hay hàm) trên một máy tính khác. Trong chương trình lúc này, tồn tại hai loại thủ tục:
  - Thủ tục cục bộ.
  - Thủ tục từ xa.

Thủ tục cục bộ là thủ tục được định nghĩa cài đặt và thực thi trên máy tính của chương trình.

Thủ tục từ xa là thủ tục được định nghĩa, cài đặt và thực thi trên máy tính khác.

Trong hệ thống RPC, Server chính là máy tính cung cấp các thủ tục từ xa cho phép chương trình trên các máy tính khác gọi thực hiện. Client chính là các chương trình có thể gọi các thủ tục ở xa trong quá trình tính toán của mình.

Với hệ thống RPC, lỗi có thể xảy ra là:

## **Phân tích những lỗi có thể xảy ra khi gọi thủ tục từ xa.**

\_ Lỗi nhất thời: lỗi này thường xuất hiện 1 lần rồi biến mất. Để khắc phục thì chỉ cần thực hiện lại lần hoạt động gây ra lỗi.

\_ Lỗi chập chờn: lỗi xuất hiện rồi biến mất, sau đó lại xuất hiện và cứ tiếp tục như vậy.

\_ Lỗi thường xuyên: là loại lỗi vẫn tồn tại ngay cả khi đã xác định được nguyên nhân và sửa thành phần gây ra lỗi.

=> các mô hình lỗi:

\_ Lỗi sụp đổ: máy chủ bất ngờ bị treo mặc dù trước đó vẫn hoạt động bình thường, cách khắc phục duy nhất là khởi động lại máy.

\_ Lỗi bỏ sót: máy chủ không thể đáp ứng được yêu cầu gửi tới: gồm 2 loại: lỗi nhận thông điệp và đáp ứng thông điệp.

\_ Lỗi đáp ứng thời gian: máy chủ phản hồi kết quả chập chờn hơn thời gian cho phép.

\_ Lỗi giá trị: giá trị trả về không chính xác.

\_ Lỗi chuyên trạng thái: Tiến trình máy chủ hoạt động không đúng với luồng điều khiển, kết quả trả về ngoài ý muốn.

\_ Lỗi phức tạp: có thể xảy ra ở bất cứ thời điểm nào.

## **Cân bằng tải là gì? Nêu cách ứng dụng khi xây dựng hệ thống phân tán.**

- Cân bằng tải là một phương pháp phân phối khói lượng tải trên nhiều máy tính hoặc một cụm máy tính để có thể sử dụng tối ưu các nguồn lực, tối đa hóa thông lượng, giảm thời gian đáp ứng và tránh tình trạng quá tải trên máy chủ. -> Cân bằng tải dùng để chia sẻ dữ

liệu truyền trên mạng giúp cho việc truyền tải thông suốt, hạn chế nghẽn mạch do quá tải hay do một sự cố nào đó. Hoặc khi có một máy server nào đó bị trục trặc thì sẽ có một máy server khác nhận thay dữ liệu, giúp việc truyền tải không bị ngừng do máy server bị lỗi gây ra. - Ứng dụng cân bằng tải vào xây dựng hệ thống phân tán: Hệ thống phân tán bao gồm mạng vật lý, các nút và các phần mềm điều khiển. Một trong những cách cải tiến sự tối ưu của hệ thống phân tán là thông qua sự cân bằng tải giữa các nút hay các máy chủ. Có nhiều phương pháp để chia tải, trong đó có 3 phương pháp phổ biến nhất là: + Kết hợp nhiều server một cách chặt chẽ tạo thành một server ảo: các hệ điều hành cho máy chủ thế hệ mới của các hãng Microsoft, IBM, ... đều cung cấp khả năng này. -> Ưu điểm : có thể chia sẻ nhiều tài nguyên trong hệ thống, theo dõi được trạng thái của các máy chủ trong nhóm để chia tải hợp lý Nhược điểm: tính phức tạp cao, khả năng mở rộng bị hạn chế, phức tạp trong triển khai và khắc phục khi xảy ra sự cố + Dùng proxy: tận dụng khả năng chia tải sẵn có trên phần mềm proxy như ISA Proxy của Microsoft -> Ưu điểm: chi phí thấp, khả năng mở rộng tốt, dễ quản trị Nhược điểm: tính ổn định kém, hiệu năng thấp, dễ mắc lỗi + Dùng thiết bị chia kết nối: dùng các module cảm biến trên các thiết bị chuyên dụng như Bộ định tuyến hay Bộ chuyển mạch để chia tải theo luồng -> Ưu điểm: hiệu năng cao, tính ổn định cao, khả năng mở rộng tốt do dùng thiết bị chuyên dụng Nhược điểm: giá thành cao, chia tải động kém, không theo dõi được trạng thái của các máy chủ, dễ gây lỗi ứng dụng => Như vậy để xây dựng một hệ thống phân tán hoàn chỉnh có khả năng quản trị lưu lượng, có khả năng kiểm soát, điều khiển và tối ưu hóa lưu lượng chạy qua nó cần phải lựa chọn công nghệ phù hợp với yêu cầu thực tế hoặc kết hợp chúng lại.

## **Trình bày qui trình hoạt động khi gọi thủ tục trên một máy tính.**

Các thủ tục được gọi từ bất cứ nơi nào của chương trình nhờ lệnh gọi thủ tục CALL. Để khi chấm dứt việc thi hành thủ tục thì chương trình gọi được tiếp tục bình thường, ta cần lưu giữ địa chỉ trả về tức địa chỉ của lệnh sau lệnh gọi thủ tục CALL.

Khi chấm dứt thi hành thủ tục, lệnh trả về RETURN nạp địa chỉ trả về vào PC. Khi gặp một lời gọi thủ tục thì nó sẽ bắt đầu được thực hiện Khi một thủ tục được kích hoạt, hệ điều hành nạp mã chương trình vào bộ nhớ trong.Nói cách khác, khi máy gặp lời gọi thủ tục ở một vị trí nào đó trong chương trình, máy sẽ tạm dời chỗ đó và chuyển đến thủ tục tương ứng. Quá trình đó diễn ra theo trình tự sau:

- Cấp phát bộ nhớ cho các biến cục bộ.
- Gán giá trị của các tham số thực cho các tham số hình thức tương ứng.

- Thực hiện các câu lệnh trong thân hàm/thủ tục.

Cụ thể :

1. Địa chỉ của ô nhớ chứa lệnh đầu tiên của chương trình được nạp vào bộ đệm chương trình PC;
  2. Địa chỉ ô nhớ chứa lệnh từ PC được chuyển đến bus địa chỉ thông qua thanh ghi MAR;
  3. Bus địa chỉ chuyển địa chỉ ô nhớ đến đơn vị quản lý bộ nhớ (MMU - Memory Management Unit);
  4. MMU chọn ra ô nhớ và thực hiện lệnh đọc nội dung ô nhớ;
  5. Lệnh (chứa trong ô nhớ) được chuyển ra bus dữ liệu và tiếp theo đó chuyển tiếp đến thanh ghi MBR;
  6. MBR chuyển lệnh đến thanh ghi lệnh IR; IR chuyển lệnh vào bộ điều khiển CU;
  7. CU giải mã lệnh và sinh các tín hiệu điều khiển cần thiết, yêu cầu các bộ phận chức năng của CPU, như ALU thực hiện lệnh;
  8. Giá trị địa chỉ trong bộ đệm PC được tăng lên 1 đơn vị lệnh và nó trả đến địa chỉ của ô nhớ chứa lệnh tiếp theo;
  9. Các bước từ 2-8 được lặp lại với tất cả các lệnh của chương trình.
- Khi gặp câu lệnh return hoặc dấu } cuối cùng của thân hàm thì máy sẽ xoá các tham số hình thức, biến cục bộ và ra khỏi hàm. Nếu trả về từ một câu lệnh return có chứa biểu thức thì giá trị của biểu thức được gán cho hàm. Giá trị của hàm sẽ được sử dụng trong các biểu thức chứa nó.

## Trình bày thuật toán Chord

Chord là một trong những giao thức phổ biến nhất được sử dụng trong bảng băm phân tán. Hoạt động chung của Chord: - Sử dụng bảng băm consistent hashing gán cho mỗi node và mỗi key một định danh m-bit sử dụng SHA-1 (Secure Hash Standard). - Consistent hashing tạo ra phân phối đều trên các tập Key ID và Node ID -Key ID và Node ID đều được ánh xạ vào trong cùng một không gian định danh ID (vòng Ring). Cách xây dựng: - Các định danh được sắp xếp trên vòng tròn định danh theo chiều kim đồng hồ có độ dài  $2m$ (Chord Ring). - Một khóa k sẽ được gán cho một node mà địa chỉ định danh của nó bằng hoặc lớn hơn địa chỉ định danh của khóa. - Node này được gọi là successor(k), đó là node đầu tiên theo chiều kim đồng hồ tính từ k. + Tìm kiếm dữ liệu với thuật toán Chord: - Tìm kiếm đơn giản : Thuật toán này chỉ yêu cầu các nút biết được successor của mình. Truy vấn cho một định danh được chuyển xung quanh vòng Chord qua các nút successor cho đến khi gặp nút có chứa khóa với định danh cần tìm. Để tăng tốc độ quá trình tìm kiếm, Chord sử dụng thêm một số thông tin định tuyến. + Thêm 1 nút mới vào mạng : Để tham gia vào mạng, nút n thực hiện truy vấn tìm kiếm cho chính id của nó thông qua một số nút ban đầu đã tham gia vào mạng và tự đưa nó vào vòng Chord, các con trỏ successor và predecessor cũng được cập nhật. + Khi một nút tự nguyện rời khỏi mạng, tất cả các khóa (các item liên quan đến khóa) được chuyển cho successor, sau đó thông báo cho successor và predecessor cập nhật. + Khi 1 nút đột ngột rời : Chord lưu trên mỗi nút một danh sách các nút nằm sau nó trong vòng Chord. Nếu một nút đột ngột không liên lạc được với successor thì nó sẽ sử dụng các nút liền sau trong danh sách. Tiếp nữa các khóa (các item liên quan tới khóa) sẽ được sao chép trên các nút có trong danh sách đó. Do đó một khóa (item liên quan đến khóa) sẽ chỉ bị mất khi có  $\log_2(N)+1$  các nút trong danh sách phải đồng thời rời khỏi mạng. Nhược điểm của thuật toán Chord (trên mạng có độ ổn định kém) - Không có sự cập nhật bảng định tuyến tức thời khi có sự thay đổi node mạng. - Không có cơ chế sao chép các khóa : các nút gia nhập và rời đi bất thường thì các khóa mà nút đang nắm giữ dễ bị lỗi hoặc không tìm thấy. - Không có

cơ chế lưu trữ kết quả tìm kiếm : kiến trúc Chord không có cơ chế lưu trữ lại các nút trên tuyến đường truy vấn dữ liệu mà nó đã đi qua. Nên khi có yêu cầu sẽ phải thực hiện lại từ đầu.

### **Trình bày mô hình phân tầng, cho ví dụ.**

Mô hình phân tầng là mô hình được tổ chức thành từng lớp có sự ràng buộc chặt chẽ, lớp trên được gọi các thành phần lớp dưới liền kề

Bên yêu cầu gửi thông tin yêu cầu được lưu chuyển từ lớp trên xuống lớp dưới, kết quả trả về được chuyển từ lớp dưới lên lớp trên. Bên thực hiện yêu cầu tiếp nhận thông tin yêu cầu được lưu chuyển từ lớp dưới lên lớp trên, kết quả trả về được chuyển từ lớp trên xuống lớp dưới. Như vậy, số lượng tầng càng lớn thì hệ thống sẽ càng được mô đun hóa cao và hiệu năng phụ thuộc vào số lượng tầng.

VD:

Phân tích thiết kế của máy tính theo quan điểm phân tầng, mỗi hệ thống thành phần của mạng được xem như một cấu trúc đa tầng, trong đó mỗi tầng được xây dựng trên tầng trước đó, số lượng các tầng và chức năng của mỗi tầng phụ thuộc vào nhà thiết kế.

## **Phân biệt dự phòng nguội và dự phòng nóng trong các hệ thống phân tán.**

Sự khác nhau giữa dự phòng nguội và dự phòng nóng trong các hệ thống phân tán: \* Dự phòng nguội - Cho phép thay thế trực tuyến các module vào/ra và các card khác - Cho phép thay thế các trạm điều khiển trong một thời gian nhanh nhất \* Dự phòng nóng - Dự phòng CPU: Mỗi trạm điều khiển cần có CPU dự phòng cạnh tranh, thực hiện song song và đồng bộ với CPU chính và so sánh kết quả - Dự phòng trạm điều khiển: Dự phòng dự trữ 1:1, chuyển mạch kịp thời, tron tru - Dự phòng dự trữ hệ thống mạng: Dự phòng cáp truyền, dự phòng module truyền thông và các thiết bị mạng khác, chuyển mạch kịp thời, tron tru, thời gian chuyển mạch < 1ms - Dự phòng vào/ra - Dự phòng trạm vận hành 1:n - Dự phòng trạm server 1:1

## **Trình bày tính trong suốt về hiệu năng trong các hệ thống phân tán.**

Trong suốt về hiệu năng trong một hệ thống phân tán là cố gắng giành được tính nhất quán và khẳng định (không cần thiết ngang bằng) mức độ hiệu năng thậm chí khi thay đổi cấu trúc hệ thống hoặc phân bố tải. Hơn nữa, người dùng không phải chịu sự chậm trễ hoặc thay đổi quá mức khi thao tác từ xa. Trong suốt hiệu năng còn được thể hiện là hiệu năng hệ thống không bị giảm theo thời gian

## **Trình bày tính trong suốt về vị trí trong các hệ thống phân tán.**

Trong suốt về vị trí (Location transparency) trong các hệ thống phân tán là che giấu về vị trí của tài nguyên. - Đó là người sử dụng không biết cũng như không quan tâm vị trí của các tài nguyên trên hệ thống vì vị trí của chúng không ảnh hưởng đến cách thức truy xuất. Giống như người sử dụng không cần biết về vị trí vật lí của dữ liệu mà có quyền truy cập đến cơ sở dữ liệu tại bất cứ vị trí nào. - Các thao tác để lấy hoặc cập nhật một dữ liệu từ xa được tự động thực hiện bởi hệ thống tại thời điểm đưa ra yêu cầu. - Tính trong suốt về vị trí rất hữu ích, nó cho phép người sử dụng bỏ qua các bản sao dữ liệu đã tồn tại ở mỗi vị trí. Do đó có thể di chuyển một bản sao dữ liệu từ một vị trí này đến vị trí khác và cho phép tạo các bản sao mới mà không ảnh hưởng đến các ứng dụng.

## **Trình bày vai trò của tính chịu lỗi trong các hệ thống phân tán.**

Vai trò của tính chịu lỗi trong hệ thống phân tán là: tối thiểu hóa các ảnh hưởng của lỗi đến hệ thống. Nói cách khác, khi có lỗi xảy ra thì hệ thống vẫn vận hành theo cách có thể chấp nhận được hay tự khởi

tạo lại trạng thái tốt nhất và đồng thời đảm bảo tính mượt mà nhỏ nhất, tự xử lý các trường hợp lỗi. Tính chịu lỗi của một hệ thống liên quan mật thiết tới khái niệm hệ thống tin cậy, một hệ thống được coi là đáng tin cậy nếu đáp ứng được bốn tiêu chí sau: - **Khả năng sẵn sàng phục vụ** của hệ thống - Độ tin cậy của hệ thống - Độ an toàn của hệ thống - **Khả năng bảo trì** hệ thống

## **Trình bày tính trong suốt về tương tranh trong các hệ thống phân tán.**

- Trong suốt: người dùng và lập trình viên không nhìn thấy được sự tồn tại của các thành phần trong hệ thống - tương tranh xảy ra khi nhiều hơn một luồng thực thi nhiệm vụ => tính trong suốt về tương tranh trong các hệ thống phân tán: + Người dùng không biết được các luồng đang truy xuất đến tài nguyên nào (cục bộ hay tài nguyên ở vị trí nào đó) dẫn đến việc truy xuất đến một tài nguyên có tranh chấp. + không biết được vị trí tài nguyên trên hệ thống + Cũng như không thay đổi được vị trí + có thể nhân bản đặt ở vị trí khác nhau nhưng không thể biết được sự tồn tại của nhân bản + Có thể một dữ liệu sẽ được chia sẻ vào cung một vị trí nhưng sẽ không biết được sự chia sẻ này => trùng lặp + không thể tự động khắc phục được sự cố bởi vì không biết chính xác về vị trí

## **Trình bày tính trong suốt về tính nhất quán của các hệ thống phân tán.**

Tính chất mấu chốt nhất phân biệt hệ phân tán với các hệ thống khác là tính trong suốt, thuật ngữ thường xuyên được nhắc trong các hệ thống phân tán. Nó là mục tiêu thúc đẩy việc che khuất đi những chi tiết phụ thuộc hệ thống mà không thích hợp đối với người dùng

trong mọi hoàn cảnh và tạo ra một môi trường thuận nhất cho người dùng. Nguyên lý này đã được thực tế hóa khi thiết kế hệ thống máy tính qua một thời gian dài. Tính trong suốt trở nên quan trọng hơn trong hệ thống phân tán và thực hiện khó khăn hơn chính từ tính hỗn tạp của hệ thống. Tính trong suốt thể hiện trong nhiều khía cạnh, một số khía cạnh điển hình là: trong suốt truyền thông, trong suốt tên, trong suốt di trú, trong suốt tương tranh, trong suốt về tính nhất quán... Trong suốt về tính nhất quán (hay còn gọi là trong suốt nhân bản): đưa ra tính nhất quán của đa thể hiện (hoặc vùng) của file và dữ liệu. Tính chất này quan hệ mật thiết với trong suốt đồng thời song được cụ thể hơn vì file và dữ liệu là loại đối tượng đặc biệt.

## **Trình bày tính trong suốt về lỗi trong các hệ thống phân tán.**

Trong suốt lỗi (Failure Transparency): Che giấu lỗi và phục hồi tài nguyên. Có lỗi xảy ra trên hệ thống nhưng người dùng không biết lỗi ở đâu Các vấn đề chính của hệ thống liên quan đến trong suốt lỗi là: Kiểm soát lỗi, Cấu hình Thu gọn VD: Nếu máy chủ được nhân bản, máy khách có thể gửi yêu cầu đến tất cả máy chủ nhưng sẽ chỉ nhận kết quả trả về từ 01 máy. Để đảm bảo tính trong suốt về lỗi, đối với lỗi truyền thông thì phần mềm trung gian trên máy khách có thể gửi lại một vài lần hoặc gửi yêu cầu đến máy chủ khác xử lý, trường hợp xấu nhất có thể lấy dữ liệu đã ghi nhớ (cache) của phiên liền trước

## **Trình bày tính trong suốt về qui mô của các hệ thống phân tán.**

- Tính trong suốt của 1 hệ phân tán được hiểu như là việc che khuất đi các thành phần riêng biệt của hệ đối với người sử dụng và những

người lập trình ứng dụng. - Tính trong suốt về quy mô của hệ thống:  
+ Người dùng sẽ không biết được quy mô hệ thống phân tán to hay nhỏ, và không biết được hệ thống bao gồm những gì. + Người sử dụng không cần biết về quy mô của dữ liệu mà có quyền truy cập đến cơ sở dữ liệu tại bất cứ nơi nào. + Các thao tác để lấy hoặc cập nhật một dữ liệu từ xa được tự động thực hiện bởi hệ thống tại điểm đưa ra yêu cầu. - Ví dụ: Người dùng truy cập vào 1 trang web và đăng ký tài khoản, họ sẽ không thể biết được hệ thống phân tán bên trang chủ bao gồm những gì, mà chỉ có quyền thao tác cơ bản đến cơ sở dữ liệu.

## **Liệt kê các chỉ số về hiệu năng của hệ thống thông tin.**

### **Thời gian đáp ứng**

Là lượng thời gian từ lúc hoàn thành quá trình nhập từ một đầu vào đến lúc bắt đầu quá trình xuất từ một đầu ra. Ví dụ, khi một yêu cầu xử lý sinh ra từ bàn phím máy tính, thời gian này là lượng thời gian cho đến kết quả được hiển thị trên khai báo hiển thị hoặc cho đến khi bắt đầu in ra. Đây cũng là yếu tố chính được sử dụng để đánh giá hiệu năng của một hệ thống thông tin.

### **Thông lượng (Khả năng xử lý)**

Là số lượng công việc có thể được xử lý bởi một hệ thống máy tính trong một đơn vị thời gian nhất định, hoặc là số lượng thời gian để xử lý một công việc nào đó

### **Thời gian quay vòng (TAT)**

Về mặt kỹ thuật, đây là lượng thời gian để tạo ra sự quay vòng của hệ thống. Trong xử lý theo khai báo, đây là lượng thời gian tồn tại từ sự xem xét của một chương trình tại cửa sổ đến khi đạt được các kết quả. Trong các hoạt động kinh doanh, đây là lượng thời gian từ khi một khách hàng đưa ra một yêu cầu đặt hàng đến khi sản phẩm đặt hàng được chuyển tới khách hàng.

## Hỗn hợp lệnh (Instruction Mix)

Hỗn hợp lệnh được sử dụng để so sánh hiệu năng phần cứng trong các hệ thống. Cho dù phần cứng tốt, nếu hiệu năng của hệ điều hành thấp thì hiệu năng của toàn bộ hệ thống cũng thấp. Một hỗn hợp lệnh dùng cho một chương trình mức độ trung bình để tính toán thời gian thực hiện lệnh trung bình trên một lệnh và giá trị MIPS, dựa trên tần xuất thực hiện của mỗi lệnh

## Nêu nguyên nhân về độ trễ trong các hệ thống phân tán.

Nguyên nhân của độ trễ trong các hệ thống phân tán thứ nhất là do độ trễ của mạng, từ hai nguồn: bộ định tuyến và khoảng cách địa lý. Mỗi bộ định tuyến có gói tin đi qua thì nó phải sao gói tin đó từ một giao diện mạng sang giao diện mạng tiếp theo. Việc này gây ra độ trễ nhỏ khoảng một vài mili giây. Tuy nhiên, giao thông trong mạng có thể phải đi qua rất nhiều bộ định tuyến khi thực hiện một round trip giữa hai máy tính, vì vậy độ trễ tăng lên nhiều lần. Bộ định tuyến hoặc mạng khi có quá nhiều gói tin cũng gây ra độ trễ đáng kể vì bộ định tuyến phải mất thêm thời gian đặt các gói tin vào hàng đợi và lấy các gói tin đó ra để xử lí.

Khoảng cách địa lý cũng gây ra độ trễ. Các gói tin di chuyển trên mạng với tốc độ nhỏ hơn tốc độ ánh sáng. Một ước tính sơ bộ cho thấy tốc độ gói tin là khoảng 100.000 dặm/h (160.000 km/h). Mặc dù tốc độ khá nhanh, nhưng nếu một gói tin phải đi qua phía bên kia của trái đất và trở lại thì ít nhất nó phải mất 250 mili giây độ trễ (không tính độ trễ tại các bộ định tuyến). Kết nối vệ tinh sẽ tốn thêm 500 mili giây độ trễ khi gửi gói tin đến hoặc nhận gói tin từ vệ tinh. Ngoài ra, đường dẫn trong mạng thường gián tiếp, các gói tin phải đi qua quãng đường xa hơn nhiều lần so với khoảng cách kết nối trực tiếp giữa hai máy tính đó.

Nguyên nhân thứ hai là ở việc hệ thống phân tán xử lí dữ liệu. Ví dụ nhân bản dữ liệu là việc tạo thêm các bản sao dữ liệu từ nơi khác tới vị trí xử lí nhằm mục đích tăng tốc độ truy nhập dữ liệu. Tuy nhiên nhân bản dữ liệu cũng đòi hỏi chi phí phải trả gồm thời gian sao dữ liệu và thời gian đảm bảo yêu tố nhất quán của dữ liệu được nhân bản.

## **Trình bày các kỹ thuật đảm bảo hiệu năng đối với các hệ thống phân tán có quy mô lớn.**

Các kỹ thuật đảm bảo hiệu năng đối với các hệ thống phân tán có quy mô lớn:

- Thực hiện truyền thông giữa các tiến trình: Trao đổi thông tin giữa các tiến trình là phần trọng tâm của tất cả các hệ thống phân tán do vậy chúng ta cần xem xét kĩ lưỡng cách thức truyền thông giữa các tiến trình trong một hệ thống phân tán.
- Loại bỏ tương tranh giữa các tiến trình: cho phép nhiều giao tác thực hiện đồng thời mà không xảy ra sự tranh chấp giữa các giao tác.
- Đồng bộ thời gian giữa các tiến trình.
- Áp dụng kĩ thuật đa luồng, ảo hóa, máy khách, máy chủ và di trú mã năng tự động phục hồi khi gặp lỗi mà ít ảnh hưởng tới sự vận hành chung của hệ thống.
- Tăng cường tính bảo mật của hệ thống. Bảo mật trong hệ thống phân tán gồm hai nhóm chính là bảo mật trong quá trình trao đổi thông tin và bảo mật việc lưu trữ thông tin tại các kho dữ liệu.
- Đảm bảo tính nhất quán và vấn đề nhân bản. Dữ liệu của hệ thống phân tán được lưu tại nhiều vị trí vật lý khác nhau và đồng thời cũng cần tạo ra các bản sao cho các dữ liệu đó vì vậy cần đảm bảo tính đồng nhất dữ liệu trong một hệ thống phân tán.

## **Trình bày tính trong suốt về truy nhập trong các hệ thống phân tán.**

- Tính trong suốt trong truy nhập được thể hiện ở việc ẩn đi cách thể hiện dữ liệu. Người dùng hoặc

ứng dụng sẽ không làm việc trên các quan hệ cơ sở dữ liệu của hệ thống qua đó sẽ không thấy được sự thể hiện các quan hệ của cơ sở dữ liệu, không thấy được cách cơ sở dữ liệu xử lý(phân mảnh ngang, dọc, kết hợp). Tính bảo mật qua được được tăng cao -Tính trong suốt trong truy cập còn được thể hiện ở việc hệ thống sẽ ẩn đi phương pháp truy cập đối với người dùng (truy cập đồng thời, truy cập tuần tự) -Việc truy cập tài nguyên địa phương hay cục bộ theo cùng một cách thức và hoàn toàn trong suốt đối với người dùng (người dùng không cảm nhận được sự tách rời tài nguyên hệ thống)

Trình bày tính nhất quán của hệ thống phân tán

## **KHÁI NIỆM VỀ TÍNH NHẤT QUÁN TRONG HỆ THỐNG PHÂN TÁN**

Nhất quán trong hệ thống phân tán được hiểu trong ngữ cảnh các thao tác đọc/ghi dữ liệu, cho dù đó là dữ liệu được lưu trong bộ nhớ, trên tập tin hay trong hệ quản trị cơ sở dữ liệu đều gọi chung là kho dữ liệu. Kho dữ liệu có thể được lưu trữ phân tán trên nhiều máy tính, mỗi tiến trình có thể truy nhập dữ liệu trong kho được coi là có bản sao của toàn bộ dữ liệu II.

### MÔ HÌNH NHẤT QUÁN 1. KHÁI NIỆM MÔ HÌNH NHẤT QUÁN

Mô hình nhất quán là thỏa thuận giữa các tiến trình, kho dữ liệu, nếu tiến trình tuân thủ một số quy tắc thì kho dữ liệu dựa hẹn sẽ làm việc chính xác. Thông thường, tiến trình thực hiện thao tác đọc mục dữ liệu bào đó và hi vọng sẽ nhận được giá trị phản ánh kết quả ghi cuối cùng được thực hiện trên mục dữ liệu đó.

**2. VAI TRÒ** Chúng ta sẽ định nghĩa một số mô hình nhất quán, mỗi mô hình sẽ hạn chế khả năng đọc giá trị của các khoản dữ liệu

**3. CÁC MÔ HÌNH \***

Các mô hình nhất quán lấy dữ liệu làm trung tâm - Nhất quán liên tục - Nhất quán thứ tự các thao tác +>Mô hình nhất quán chật +>Mô hình nhất quán tuần tự +>Mô hình nhất quán tuyến tính +>Mô hình nhất quán nhân quả +>Mô hình nhất quán FIFO +>Mô hình nhất quán yếu +>Mô hình nhất quán đi ra +>Mô hình nhất quán đi vào

\* Các mô hình nhất quán lấy máy khách làm trung tâm +>Mô hình nhất quán sau cùng +>Mô hình nhất quán đọc đều +>Mô hình nhất quán ghi đều +>Mô hình nhất quán đọc kết quả ghi +>Mô hình nhất quán ghi theo sau đọc ==>>>Tổng kết Tính nhất quán thường đi kèm với việc nhân bản dữ liệu trong hệ thống phân tán. Khi chúng ta nói lỏng tinh nhất thì mới hi vọng đạt được các giải pháp hiệu quả khi nhân bản dữ liệu. Với mỗi hệ thống phân tán cần áp dụng mô hình nhất quán phù hợp để đảm bảo hiệu năng tối đa.

### **Liệt kê các phương pháp nâng cao hiệu năng xử lý cho hệ thống phân tán**

Các phương pháp nâng cao hiệu năng xử lý cho hệ thống phân tán là:

- Lập trình đa luồng (ưu tiên số 1): Thư viện System.Threading - Áo hóa: bản chất là giải quyết vấn đề tương tranh - Phối hợp giữa máy khách và máy chủ + Máy khách: cung cấp giao diện cho người dùng tương tác với máy chủ, ngoài ra có thể thực hiện nhiệm vụ tiền xử lý các dữ liệu trước khi chuyển yêu cầu đến máy chủ. Máy khách cần giao thức: xây dựng giao thức ứng với từng trường hợp hoặc xây dựng một lớp trung gian cho toàn bộ các dịch vụ + Máy chủ: phụ thuộc vào từng tiến trình xử lý Máy chủ trạng thái: có yêu cầu gửi đến -> tiếp nhận -> xử lý xong -> tiếp nhận yêu cầu mới Máy chủ

tương tranh: 1 yêu cầu gửi đến -> tiếp nhận -> tiến trình -> luồng xử lý và tiếp nhận yêu cầu mới Máy chủ trạng thái: cho phép xử lý song song và ko phong tỏa hệ thống - Di trú mã: để giảm trao đổi thông tin trên mạng

## Các hệ thống phân tán

- Thời gian logic: Nếu ta coi chuỗi thời gian là một vector một chiều và hai sự kiện A, B xảy ra. Nếu sự kiện A xảy ra trước sự kiện B thì VT(A) < VT(B). Ngược lại nếu VT(B) < VT(A) thì sự kiện B xảy ra trước sự kiện A.
- Đồng hồ logic: Trong thực tế, không tồn tại đồng hồ vật lý với độ chính xác tương đối. Độ lệch thời gian giữa các máy tính luôn khác 0. Các hệ thống phân tán không cần đến đồng hồ vật lý nhưng vẫn phải đồng bộ với nhau. Do đó người ta đề xuất ra đồng hồ logic để duy trì sự đồng bộ trong hệ thống phân tán.
  - Đồng hồ logic Lamport: Lamport đã đưa ra mô hình đồng hồ logic đầu tiên cùng với khái niệm nhãn thời gian. Để đồng bộ đồng hồ logic, Lamport đưa ra định nghĩa mỗi quan hệ xảy ra -trước khi (happens - before),  $A \rightarrow B$  có nghĩa là sự kiện A xảy ra trước sự kiện B, có các tính chất:
    - A và B là hai sự kiện của cùng một tiến trình. Nếu A xảy ra trước B thì  $A \rightarrow B$  là đúng.
    - Nếu A là sự kiện bản tin được gửi bởi một tiến trình nào đó, còn B là sự kiện bản tin đó được nhận bởi một tiến trình khác thì quan hệ  $A \rightarrow B$  là đúng.
    - Quan hệ xảy ra trước có tính bắc cầu:  $A \rightarrow$

$B, B \rightarrow C$  thì  $A \rightarrow C$ . Để đo thời gian tương ứng với sự kiện  $x$  thì ta gán một giá trị  $C(x)$  (nhãn thời gian của  $x$ ) cho sự kiện đó và thỏa mãn các điều kiện sau: - Nếu  $A \rightarrow B$  trong cùng một tiến trình thì  $C(A) < C(B)$ . - Nếu  $A$  và  $B$  biểu diễn tương ứng việc gửi và nhận một thông điệp thì ta có  $C(A) < C(B)$ . - Với mọi sự kiện phân biệt (không có liên quan) thì  $C(A) \leftrightarrow C(B)$ .

Thuật toán Lamport:

- Cập nhật bộ đếm  $C_i$  cho tiến trình  $P_i$  để đo thời gian tương ứng với một sự kiện.
- Trước khi thực hiện gán  $C_i = C_i + 1$ .
- Mỗi khi thông điệp  $m$  gửi đi từ tiến trình  $P_i$  đến  $P_j$ , nó đặt nhãn thời gian của thông điệp  $ts(m) = C_i$ .
- Nhận được thông điệp  $m$ ,  $P_j$  điều chỉnh  $C_j = \max(C_j, ts(m))$ , và chuyển lên lớp ứng dụng.

o Đồng hồ vector:

- Do tồn tại  $N$  đồng hồ độc lập cho  $N$  tiến trình nên không thể rút ra quan hệ trước sau hay phụ thuộc nhân quả từ các nhãn thời gian.
- Nhưng trong một số trường hợp áp dụng thuật toán thời gian vector có thể đưa ra kết luận về tính trước sau của 2 sự kiện.

Tại mỗi tiến trình  $P_i$  duy trì một đồng hồ  $V_i$ :

- Mỗi đồng hồ  $V_i$  là một vector có kích thước  $N$  (tiến trình).
- $V_i[i]$  là số sự kiện xảy ra tới thời điểm hiện tại trên  $P_i$ .
- $V_i[j]$  chứa thông tin của  $P_i$  về số sự kiện đã xảy ra ở  $P_j$ .

Thuật toán nhãn thời gian vector:

- Khởi tạo vector  $V_i$  chứa  $N$  phần tử trên tiến trình  $P_i$ .
- Trước khi xảy ra một sự kiện giá trị  $V_i[i] = V_i[i] + 1$ .
- Mỗi khi thông điệp  $m$  gửi đi từ tiến trình  $P_i$  đến  $P_j$ , nó đặt nhãn thời gian của thông điệp  $ts(m) = V_i$ .
- Sau khi nhận thông điệp tiến trình  $P_j$  cập nhật  $V_j[k] = \max(V_j[k], ts(m)[k])$  với  $k$  trong khoảng 0

đến N-1 hoặc 1 đến N. Do đó nếu A → B khi và chỉ khi V(A) < V(B)

## **CÁC ĐẶC TRƯNG VÀ MỤC TIÊU THIẾT KẾ CƠ BẢN CỦA CÁC HỆ THỐNG PHÂN TÁN**

I. Kết nối người sử dụng và tài nguyên hệ thống  
Mục tiêu chính của hệ thống phân tán là kết nối người sử dụng và tài nguyên mạng. Nhiệm vụ chính của một hệ thống phân tán là cho phép người sử dụng được khai thác thông tin mà không phụ thuộc vị trí địa lý của người đó. Như vậy nảy sinh hàng loạt vấn đề liên quan đến việc khai thác và sử dụng thông tin: ai được phép truy nhập, truy nhập thông tin ở mức độ nào, thời gian nào được phép truy nhập, tần suất truy nhập thông tin. II. Trong suốt đối với người sử dụng Mục tiêu trong suốt đối với người sử dụng nhằm che giấu vị trí thực của thông tin đối với người sử dụng, người sử dụng không biết được thông tin được lưu trữ ở đâu và xử lý trên máy tính nào. Tính trong suốt đối với người sử dụng thể hiện ở các đặc điểm sau: - Truy nhập (Access): Ẩn cách thể hiện dữ liệu và phương pháp truy nhập. - Vị trí: Ẩn nơi lưu trữ thông tin. - Di chuyển: Ẩn quá trình chuyển vị trí lưu trữ dữ liệu - Đặt lại vị trí: Ẩn quá trình di chuyển dữ liệu mà không làm gián đoạn hoạt động của người sử dụng. - Nhân bản: Che giấu việc tạo ra bản sao dữ liệu - Tương tranh: Che giấu việc chia sẻ tài nguyên cho nhiều người sử dụng - Lỗi: Che giấu lỗi và phục hồi tài nguyên - Bên bì: Che giấu tài nguyên phần mềm được tải vào bộ nhớ hay ở trên ổ đĩa. III. Tính

mở của hệ thống Để có tính mở, hệ thống phân tán phải có chuẩn giao tiếp với hệ thống, như vậy sẽ dễ dàng hơn trong việc trao đổi tài nguyên. Một hệ thống mở phải tuân thủ các tiêu chuẩn giao tiếp nào đó đã được công bố, nghĩa là sản phẩm của các nhà sản xuất khác nhau có thể tương tác với nhau theo tập các luật và các qui tắc hoặc các tiêu chuẩn đã được công bố, ví dụ: ngôn ngữ IDL, XML, giao thức dịch vụ Web.... IV. Qui mô mở rộng hệ thống Hệ thống phân tán cần phải đảm bảo dễ dàng thêm các máy tính mà không cần phải sửa đổi hệ thống, như vậy chúng ta có thể mở rộng hay thu hẹp hệ thống phân tán theo yêu cầu thực tế, đây là một đặc tính quan trọng nhất của hệ thống phân tán. Khi mở rộng hệ thống, số lượng máy tính và số lượng người sử dụng tăng thêm nhưng không được phép giảm hiệu suất hoạt động của hệ thống. Tương tự như vậy, việc mở rộng hệ thống theo phạm vi địa lý cần đảm bảo ít ảnh hưởng tới hiệu suất hoạt động của hệ thống. Trong cả hai trường hợp mở rộng trên, cần phải đảm khả năng quản trị hệ thống.

### **Các trạng thái toàn cục**

I. Mở đầu Việc xác định trạng thái toàn cục rất có ích. Tại một thời điểm bất kì trạng thái toàn cục của một hệ thống phân tán được đặc trưng bởi trạng thái của tiến trình và các thông điệp đang lưu chuyển trong hệ thống. Ví dụ: Distributed garbage collection Còn tồn tại tham chiếu đến một đối tượng cho trước? Distributed deadlock detection Các tiến trình có đợi nhau theo vòng tròn không? Distributed termination detection Các tiến trình đã ngừng mọi hoạt động chưa? Việc xác định các trạng thái toàn cục được xác định trên phương pháp được đưa ra là khái niệm lát cắt. Lát cắt mô tả sự kiện cuối cùng mà sự kiện này được các tiến trình ghi lại cho mỗi tiến trình, bằng cách này nó có thể kiểm tra lại rằng tất cả các thông điệp

nhận đều tương ứng với các thông điệp gửi được ghi lại trên đường cắt. Ví dụ về lát cắt: II. Lát cắt nhất quán Xác định các tính chất toàn cục : - Kết hợp thông tin địa phương từ các nút khác nhau - Không có giờ toàn cục, làm sao biết các thông tin đó là nhất quán? - Thông tin trạng thái địa phương lấy tại các thời điểm tùy ý không nhất quán - Điều kiện nào để một tập hợp các thông tin địa phương được gọi là nhất quán về toàn cục Nếu đồng hồ được được đồng bộ tuyệt đối, một giải pháp được đưa ra mỗi tiến trình pi ghi nhận trạng thái xi tại thời điểm tương lai t nào đó sẽ gửi xi<sup>-</sup>(t) đến tất cả các tiến trình khác. Vấn đề nằm ở chỗ giải pháp không ghi nhận các trạng thái của các kênh truyền và đồng thời bị sai lệch bởi trễ đường truyền, do đó đồng bộ tuyệt đối là không thể được. Vì vậy cần phải nói lỏng điều kiện, thay cho trạng thái toàn cục thực tế của hệ thống sẽ bắt trạng thái toàn cục nhất quán. Cho tiến trình Pi trong đó xuất hiện các sự kiện ei0 , ei1 ,....: Lịch sử tiến trình (Pi)=hi= Tiền sử tiến trình (Pik)=hik= Trạng thái tiến trình Sik là trạng thái của Pi ngay trước khi sự kiện thứ k Lát cắt mô tả sự kiện cuối cùng mà sự kiện này được ghi lại cho mỗi tiến trình. Cho tập các tiến trình P1,...,Pi,...: Lịch sử toàn cục: H= UI (hi) Trạng thái toàn cục: S= Ui(Sik i) Lát cắt C⊆H=h1c1 ∪ h2c2 ∪...∪ hnccn Ranh giới của C={eic1 ,i=1,2,...,n} Lát cắt C là nhất quán khi và chỉ khi ∀ e ∈ C(nếu f-> e thì f ∈ C). Lát cắt được coi là nhất quán mạnh nếu trạng thái toàn cục nhất quán trong đó mỗi sự kiện gửi tương ứng với sự kiện nhận. Trạng thái S là nhất quán khi và chỉ khi nó tương ứng với lát cắt nhất quán

## Mô hình nhất quán đi ra

- 1) Tại sao phải đưa ra mô hình nhất quán? Trong một hệ thống phân tán kho dữ liệu có thể được đọc hay ghi bởi bất cứ một tiến trình nào. Tuy nhiên dữ liệu ghi vào một bản sao cục bộ phải đảm bảo cũng được truyền tới tất cả các bản sao ở xa vì thế nên các mô hình nhất quán ra đời. Một mô hình nhất quán có thể được coi như một bản hợp đồng giữa một kho dữ liệu của hệ phân tán với các tiến trình của nó. Nếu các tiến trình đồng ý với các điều khoản của hợp đồng thì kho dữ liệu sẽ hoạt động đúng như như tiến trình mong muốn. 2) Mô hình nhất quán đi ra - Sử dụng hai lệnh: lệnh acquired để báo muốn vào

vùng tới hạn (critical region) và lệnh release để báo giải phóng vùng tới hạn. Hai lệnh này cũng có hai cách thực thi khác nhau như: bằng một biến hoặc bằng một lệnh đặc biệt. Hai thao tác này chỉ thực hiện với các dữ liệu dùng chung chứ không áp dụng cho tất cả các dữ liệu. - Là một loại hình đặc biệt của các biến, gọi là biến đồng bộ hóa hoặc khóa. - Khóa không thể được đọc hoặc ghi vào. Nó có thể được acquire (vào) và release (ra). Đối với một khóa L những hoạt động được biểu thị tương ứng bởi Acquire(L) và Release(L). - Không có nhiều hơn 1 tiến trình có thể nắm 1 khóa L. Một tiến trình nắm khóa trong khi các tiến trình khác chờ đợi. Mô hình nhất quán đi ra thỏa mãn các điều kiện sau: - Trước khi thực hiện một thao tác đọc hay ghi lên dữ liệu chia sẻ thì tất cả các thao tác acquire do tiến trình này thực hiện trước đó phải hoàn tất. - Trước khi một thao tác release được phép thực hiện thì tất cả các thao tác đọc và ghi do tiến trình này thực hiện trước đó phải được hoàn tất. - Truy cập vào các biến đồng bộ hóa là nhất quán FIFO (Không yêu cầu nhất quán tuần tự). (Hình ảnh) Mô hình nhất quán đi ra - Hình trên mô tả cách hoạt động của mô hình nhất quán đi ra. Tiến trình P1 nắm khóa L (thao tác chiếm giữ Acq(L)) nó phong tỏa hệ thống và tất cả các tiến trình còn lại phải ở trạng thái chờ. Sau khi thực hiện xong nó giải phóng bằng thao tác Rel(L). Sau đó tiến trình P2 tiếp tục nắm giữ khóa L (Acq(L)) nó phong tỏa hệ thống để thực hiện xong tiến trình rồi giải phóng bằng thao tác giải phóng Rel(L), tiếp đó thì khóa mới do P3 chiếm giữ. - Mô hình nhất quán đi ra được xem là nhất quán lỏng. Do đó mô hình này có những ưu điểm là cho phép hoạt động đồng bộ hóa bộ nhớ ít hơn: + Kết quả này trong giáo tiếp ít hơn và do đó hiệu suất cao hơn. + Ghi có thể được trì hoãn, tổng hợp, loại bỏ.

## Mô hình nhất quán chặt

Trong hệ thống phân tán nhân bản giúp tăng độ tin cậy , đáp ứng tính sẵn sàng của hệ thống và giúp tăng hiệu năng hệ thống .Tuy nhiên nhân bản lại làm tính nhất quán bị suy giảm .Do đó chúng ta cần xây dựng các mô hình nhất quán để đảm bảo tính toàn vẹn của dữ liệu.Có rất nhiều mô hình nhất quán được xây dựng và trong bài này chúng ta sẽ tìm hiểu mô hình nhất quán chặt. Đầu tiên mô hình nhất quán chặt là gì ?Nhất quán chặt là mô hình xây dựng theo quan điểm lấy dữ liệu làm trung tâm .Mô hình nhất quán chặt phải đáp ứng được điều kiện :”Thao tác đọc bất kỳ trên mục dữ liệu x đều trả về một giá trị tương ứng với kết quả của thao tác ghi gần nhất trên x đó .Để đảm bảo được điều đó chúng ta cần làm rõ khái niệm thao tác ghi gần nhất tức là đưa ra định nghĩa về thời gian tuyệt đối .Thời gian tuyệt đối phải bao quát tổng cả hệ thống”. Tuy nhiên thời gian tuyệt đối khó khả thi trong hệ thống phân tán. Mô hình nhất quán chặt có ưu điểm gì ?Có thể dễ dàng nhận thấy mô hình này đảm bảo tính nhất quán lý tưởng và dễ hiểu .Mô hình nhất quán chặt cũng thoả mãn được mô hình nhất quán sau cùng trong quan điểm lấy client làm trung tâm . Tuy nhiên mô hình nhất quán chặt cũng có những nhược điểm nhất định của mình .Có thể thấy khái niệm thời gian tuyệt đối cho toàn bộ hệ thống phân tán là không thể thực hiện được vì quá trình đọc ghi được thực hiện trên nhiều máy khác nhau có vị trí địa lí khác nhau mặt khác lại có sự trễ do đường truyền không thể quản lý được .....Vì vậy đây chỉ là mô hình nguyên lý không thực hiện được trong hệ thống phân tán thực tế. Tóm lại mô hình nhất quán chặt là mô hình lý tưởng cho tính nhất quán , tuy nhiên nó không thể thực hiện với hệ thống phân tán thực tế . Mô hình này có thể là thước đo để so sánh với các mô hình nhất quán khác. Tất cả các mô hình nhất quán đều có hạn chế về khả năng đọc ghi nhất định . Do đó không có mô hình nào là toàn diện . Tuỳ vào yêu cầu thực tế , chúng ta cần quyết định sử dụng mô hình nào để hệ thống hiệu quả nhất .

## Mô hình nhất quá đọc ghi kết quả

mô hình nhất quán đọc kết quả ghi : + Định nghĩa : tác động của một thao tác ghi của một tiến trình lên mục dữ liệu X , sẽ luôn được nhìn thấy bởi một thao tác đọc lần lượt trên x của cùng tiến trình đó ". + Nói cách khác : 1 quá trình ghi luôn hoàn thành trước khi 1 quá trình đọc của cùng tiến trình thực hiện bắt kể việc đọc diễn ra ở đâu. + Ưu điểm : Đảm bảo người dùng luôn nhận được kết quả mới nhất. + Nhược điểm : khi thực hiện ghi có thể người dùng không thể truy cập được dữ liệu

## Loại trừ tương hỗ phân tán

Loại trừ tương hỗ là quá trình truy cập đồng thời của các tiến trình với một tài nguyên hoặc dữ liệu được chia sẻ, thực hiện theo cách loại trừ lẫn nhau. Trong hệ thống phân tán không có các biến chia sẻ có thể sử dụng để thực hiện loại trừ lẫn nhau. Thuật toán loại trừ tương hỗ phải đối phó với sự chậm trễ của thông điệp và thiếu hoặc không đoán trước thông tin về trạng thái của hệ thống. Hai cách tiếp cận cơ bản để loại trừ lẫn nhau trong hệ thống phân tán: + Phương pháp tiếp cận dựa trên Token. + Phương pháp tiếp cận không dựa trên Token. 1. Phương pháp tiếp cận dựa trên Token. • Một thẻ bài(token) được chia sẻ giữa các thành viên của hệ thống. • Thành viên được phép truy nhập tài nguyên nếu nó nắm giữ Token. • Loại trừ lẫn nhau được đảm bảo vì token là duy nhất. • Thuật toán : giải thuật thẻ bài, giải thuật không tập trung. 1.1 Giải thuật thẻ bài . Giải thiết tất cả các tiến trình được sắp xếp trên một vòng tròn logic, các tiến trình đều được đánh số và đều biết đến các tiến trình cạnh nó. Bắt đầu quá trình truyền, tiến trình 0 sẽ được trao một thẻ bài. Thẻ bài này có thể lưu hành xung quanh vòng tròn logic. Nó được chuyển từ tiến trình (k) đến tiến trình (k+1) để truyền thông điệp điểm – điểm. Khi một tiến trình nhận được thẻ bài từ tiến trình bên cạnh nó sẽ kiểm tra xem có cần thiết vào vùng tới hạn hay không. Nếu không cần thiết thì chuyển thẻ bài cho tiến trình khác, ngược lại sẽ vào vùng tới hạn. Sau khi hoàn thành phần việc của mình nó sẽ trả thẻ bài cho tiến trình kế tiếp. Vấn đề lớn nhất trong thuật toán truyền thẻ bài là thẻ bài có thể bị mất, khi đó hệ thống sẽ phải tạo lại thẻ bài bởi vì việc dò tìm lại thẻ bài là rất khó. 1.2 Giải thuật không tập trung. Mỗi tài nguyên sẽ được nhân bản N lần và có tiến trình điều phối riêng để kiểm soát truy nhập. Một tiến trình muốn truy nhập tài nguyên thì phải gửi yêu cầu đến N tiến trình điều phối đó (nếu bị hỏng thì bỏ qua) và sẽ được cấp quyền truy nhập khi và chỉ khi nhận được số

phiếu đồng ý lớn hơn N/2. 2.Phương pháp tiếp cận không dựa trên Token. •Sử dụng hai hoặc nhiều vòng liên tiếp các thông điệp được trao đổi giữa các thành viên để xác định thành viên nào sẽ được truy cập vào tài nguyên tiếp theo. •Các giải thuật: giải thuật tập trung, giải thuật phân tán. 2.1 Giải thuật tập trung . Một tiến trình được bầu chọn làm điều phối viên, cấp quyền truy cập tài nguyên. Để truy cập tài nguyên tiến trình gửi một yêu cầu tới điều phối viên và sau đó chờ trả lời.Trả lời từ điều phối viên cho quyền vào truy nhập. Sau khi hoàn thành công việc tiến trình thông báo cho điều phối viên. Ưu điểm: +Đảm bảo sự tồn tại duy nhất một tiến trình trong vùng giới hạn và chỉ cần 3 thông điệp để thiết lập là: Yêu cầu (Request), Cấp phép (Grant) và Giải phóng (Release). Nhược điểm: +Nếu tiến trình điều phối bị hỏng thì sẽ không có một tiến trình nào được cấp phép. +khi một tiến trình đang trong vùng giới hạn mà bị phong tỏa thì tài nguyên cũng không được giải phóng. +Trong một hệ thống lớn nếu có một tiến trình điều phối sẽ xuất hiện hiện tượng thắt cổ chai. 2.2 Giải thuật phân tán. Khi một tiến trình muốn vào vùng giới hạn, trước hết nó sẽ tạo ra một nhãn thời gian và gửi cùng với một thông điệp đến tất cả các tiến trình khác. Các tiến trình khác sau khi nhận được thông điệp này sẽ xảy ra ba tình huống: - Nếu bên nhận không ở trong vùng giới hạn và cũng không muốn vào vùng giới hạn thì nó sẽ gửi thông điệp chấp nhận(OK) cho bên gửi. - Nếu bên nhận đang ở trong vùng giới hạn thay vì trả lời nó sẽ cho vào hàng đợi yêu cầu này. - Nếu bên nhận cũng muốn vào hàng đợi thì nó sẽ so sánh timestamp ai thấp hơn sẽ thắng. Sau khi gửi đi thông điệp yêu cầu vào vùng giới hạn tiến trình sẽ đợi cho đến khi có trả lời càng sớm càng tốt. Khi đã vào vùng giới hạn rồi thì nó sẽ gửi thông điệp OK đến tất cả các tiến trình khác và xóa các tiến trình trong hàng đợi.

## Các mô hình dựa trên kiến trúc hướng dịch vụ

1. Khái niệm kiến trúc hướng dịch vụ Kiến trúc hướng dịch vụ - Service Oriented Architecture (SOA): là một chiến dịch cho biết dự định xây dựng tất cả các tài sản phần mềm của công ty đó bằng cách sử dụng phương pháp luận lập trình hướng dịch vụ. Kiến trúc hướng dịch vụ phân rã chức năng của hệ thống thành các dịch vụ, các dịch vụ được phân rã nhỏ hơn thành các dịch vụ con...Một quy trình ngược lại là kết hợp các dịch vụ con lại với nhau theo một chính

sách nào đó nhằm đạt được mục đích cuối cùng là phục vụ các mục tiêu nghiệp vụ của các doanh nghiệp. 2. Các mô hình kiến trúc dựa trên SOA Các mô hình kiến trúc dựa trên SOA thoạt nhìn khá phức tạp, nó tùy thuộc vào góc nhìn đối với hệ thống, có thể đó là góc độ kinh doanh hay kỹ thuật. Mô hình logic phân chia hệ thống thành các miền, mỗi miền đảm nhiệm vai trò và trách nhiệm riêng. Khái niệm miền ở đây phản ánh một thực thể nào đó, ví dụ đó là công ty, phòng/ban.... Đứng trên góc độ kỹ thuật có thể thấy trực dịch vụ doanh nghiệp ESB đóng vai trò trung tâm, các vùng chỉ cung cấp các dịch vụ cơ bản và dịch vụ tích hợp, các dịch vụ qui trình được tách biệt riêng rẽ. Với vai trò hỗ trợ rất nhiều mẫu tích hợp và làm trung tâm kiểm soát và phân bổ quá trình xử lý, Trục dịch vụ doanh nghiệp ESB đã gián tiếp kết nối các ứng dụng có sử dụng các dạng dịch vụ khác nhau để liên thông, chia sẻ dữ liệu cũng như quy trình tạo nên một kiến trúc tổng thể góp phần thúc đẩy mạnh việc xây dựng chính phủ điện tử toàn diện, minh bạch, tiết kiệm chi phí, thời gian. ESB hỗ trợ rất nhiều kiểu mẫu tích hợp với mục đích hỗ trợ đầy đủ các kiểu mẫu tích hợp khác nhau theo yêu cầu trong một kiến trúc SOA (ví dụ yêu cầu/phản hồi, xuất bản/đăng ký hay sự kiện). Trục dịch vụ doanh nghiệp ESB đóng vai trò trung tâm kiểm soát và phân bổ quá trình xử lý Trục dịch vụ doanh nghiệp ESB đôi khi được mô tả như một hạ tầng phân bổ với các giải pháp (như các công nghệ “môi giới”) và thường được mô tả như là hub - và - spoke. Hình bên dưới thể hiện sự mô tả này của ESB. Tuy nhiên, cách nhìn nhận về ESB không thực sự giúp ích gì được nhiều trong việc mô tả cách mà ESB được triển khai thực tế. Ví dụ, những thành phần hạ tầng nào xây dựng nên ESB mà được mô tả trong hình bên dưới như một đường thẳng? Hình 1 - Mô hình Trục dịch vụ doanh nghiệp ESB được thiết kế như một hạ tầng vật lý Ngược lại, các giải pháp tích hợp hub - và - spoke (hình 2) tìm cách tập trung kiểm soát các cấu hình: thông tin định tuyến, đặt tên dịch vụ... Hình 2 - Mô hình tích hợp hub - và - spoke Sự khác biệt giữa hai giải pháp phân bố Bus và hub - và - spoke tập trung thực ra lại là một sai lầm. Có hai vấn đề ở đây đó là sự tập trung về kiểm soát và sự phân bố của hạ tầng. Trong giai đoạn đầu tiên hoặc đối với các mô hình triển khai ở mức độ nhỏ của các giải pháp tích hợp thì hạ tầng vật lý có xu hướng tập trung: tập trung vào một cụm duy nhất, hoặc một Hub, hoặc tập trung vào các máy chủ. Tuy nhiên, trong quá trình triển khai, hạ tầng lại có xu hướng trở thành phân bố hơn là tập trung, như là một Bus. Điều quan trọng nữa trong việc triển khai công nghệ ESB đó là khả năng mở rộng các công việc hiện tại bằng việc bổ sung khả năng xử lý phân bổ mà

không ảnh hưởng đến hạ tầng hiện tại. Hình 3 - Trục dịch vụ doanh nghiệp ESB được thiết kế như một hạ tầng phân bổ với việc kiểm soát tập trung

## **Khái niệm chung về hệ thống phân tán**

1> Định nghĩa 2> Mục tiêu của hệ phân tán 3> Các khái niệm phần cứng 4> Các khái niệm phần mềm 5> Phân loại kiến trúc hệ thống phân tán \*\*\*\*\* 1. Định nghĩa. - Hệ thống phân tán bao gồm các máy tính (bao gồm cả các thiết bị khác như PDA, điện thoại di động...) được kết nối với nhau để thực hiện nhiệm vụ tính toán. - Hệ thống phân tán là tập hợp các máy tính tự trị được kết nối với nhau bởi một mạng máy tính và được cài đặt phần mềm hệ phân tán. - Hệ thống phân tán là một hệ thống có chức năng và dữ liệu phân tán trên các trạm (máy tính) được kết nối với nhau bởi một mạng máy tính. - Hệ thống phân tán là một tập các máy tính độc lập giao tiếp với người dùng như một hệ thống thống nhất, toàn vẹn. - Nếu vấn đề trao đổi thông tin giữa các máy tính không được thực hiện thì nhiệm vụ tính toán trên môi trường phân tán sẽ không thực hiện được. Những lý do để xây dựng các ứng dụng phân tán: - Yêu cầu tính toán phân tán. - Yêu cầu khả năng xử lý lỗi. - Chia sẻ tài nguyên. => Như vậy, có thể nói : Hệ phân tán = mạng máy tính + phần mềm hệ phân tán. Phân loại hệ phân tán: - hệ thống điện toán phân tán. + Hệ thống điện toán cụm + Hệ thống điện toán lưới. - Các hệ thống thông tin phân tán - Các hệ thống lan tỏa phân tán 2. Mục tiêu của hệ phân tán. a. Kết nối người sử dụng và tài nguyên - Giải quyết bài toán chia sẻ tài nguyên trong hệ thống. b. Tính trong suốt - Truy nhập (Access): Ân cách thể hiện dữ liệu và phương pháp truy nhập. - Vị trí: Ân nơi lưu trữ thông tin. - Di chuyển: Ân quá trình chuyển vị trí lưu trữ dữ liệu - Đặt lại vị trí: Ân quá trình di chuyển dữ liệu mà không làm gián đoạn hoạt động của người sử dụng. - Nhân bản: Che giấu việc tạo ra bản sao dữ liệu - Tương tranh: Che giấu việc chia sẻ tài nguyên cho nhiều người sử dụng - Lỗi (Failure): Che giấu lỗi và phục hồi tài nguyên - Bền bỉ: Che giấu tài nguyên phần mềm được tải vào bộ nhớ hay ở trên ổ đĩa. c. Tính mở (Openness) - Để có tính mở, hệ thống phân tán phải có chuẩn giao tiếp với hệ thống, như vậy sẽ dễ dàng hơn trong việc trao đổi tài nguyên. Một hệ thống mở phải tuân thủ các tiêu chuẩn giao tiếp nào đó đã được công bố, sản phẩm của các nhà sản xuất khác nhau có thể tương tác với nhau theo tập các luật và các qui tắc hoặc các tiêu chuẩn đã được công bố. - Liên tác: các cài đặt của các hệ

thống hoặc thành phần hệ thống từ các nhà sản xuất khác nhau có thể làm việc với nhau thông qua liên tác. - Chuyển mang: nhờ chuyển mang mà một ứng dụng được phát triển cho hệ phân tán A có thể thực hiện không cần thay đổi gì trên một hệ phân tán B khác, với điều kiện được cài đặt cùng giao diện như A d. Tính co giãn (Scalability) Một hệ phân tán được gọi là có tính co giãn nếu nó thích nghi với sự thay đổi quy mô của hệ thống. Thể hiện trên các khía cạnh sau: - Dễ bổ sung người sử dụng và tài nguyên hệ thống - Khi hệ thống thay đổi quy mô về mặt địa lý dẫn đến sự thay đổi về vị trí địa lý của người sử dụng và các tài nguyên. - Hệ thống có thay đổi quy mô về quản trị. Nếu hệ phân tán có tính co giãn thường ảnh hưởng đến hiệu năng của hệ thống (hiệu năng của hệ thống là hiệu quả năng lực hoạt động của đối tượng). Có ba giải pháp phổ dụng để giải quyết vấn đề co giãn của hệ phân tán: - Ân giấu - Phân tán: phân nhỏ thành phần hệ thống và phân bố chúng trên phạm vi của hệ thống (quản lý phân cấp). - Nhân bản: nhân bản một thành phần nào đó của hệ thống. Ví dụ tài nguyên dữ liệu đặt tại các vị trí khác nhau trong hệ thống. e. Các kiến trúc của hệ thống phân tán: - Hệ thống nhiều bộ vi xử lý (các bộ vi xử lý dùng chung bộ nhớ). - Hệ thống nhiều tính máy tính (mỗi bộ vi xử lý có bộ nhớ riêng). - Hệ thống nhiều máy tính được coi là đồng nhất nếu các máy tính cùng chung nền tảng(phần cứng, hệ điều hành, mạng) ngược lại gọi là hệ thống không đồng nhất. 3. Các khái niệm phần cứng. a. Phân loại máy tính. Có hai loại máy tính: - Các loại máy tính có chia sẻ bộ nhớ: các loại máy đa xử lý.

## Ảo hóa

1.Ảo Hóa là gì? Ảo hóa là tạo ra một phiên bản phụ (phiên bản ảo) của máy tính (có thể là một phần mềm hay linh kiện trong máy tính thậm chí là toàn bộ ) để tận dụng tài nguyên máy tính , tài nguyên mạng của hệ thống đem lại hiệu quả xử lý cao hơn. Ảo hóa cho hiệu quả cao hơn, tính sẵn sàng của dữ liệu cao hơn và chi phí thấp hơn  
Ảo hóa là cách hiệu quả nhất để giảm chi phí CNTT trong khi hiệu quả được tăng nhanh, không chỉ cho các doanh nghiệp lớn mà còn đối với các doanh nghiệp vừa và nhỏ. Ảo hóa cho phép bạn: • Chạy nhiều hệ điều hành và các ứng dụng trên một máy tính • Cung cấp phần cứng để năng suất được cao hơn từ các máy chủ có cấu hình thấp hơn • Tiết kiệm 50% trên tổng chi phí CNTT • Tăng tốc độ và đơn giản hóa quản lý CNTT, bảo trì và triển khai các ứng dụng mới

Ảo hóa hạ tầng CNTT là một thách thức cấp bách nhất hiện nay: sự mở rộng cơ sở hạ tầng buộc bộ phận IT phải chi hơn 70% ngân sách của họ vào việc bảo trì và để lại rất ít tài nguyên nguồn lực cho việc đổi mới cách kinh doanh hiện tại. Những khó khăn xuất phát từ những máy tính chạy hệ điều hành X86 ngày nay: chúng chỉ được thiết kế để chạy một hệ điều hành và ứng dụng tại một thời điểm. Và kết quả là ngay cả một trung tâm dữ liệu nhỏ cũng phải triển khai nhiều máy chủ, trong khi mỗi máy chỉ hoạt động ở mức từ 5% đến 15% công suất – không hiệu quả đối với bất kỳ tiêu chuẩn nào. Phần mềm ảo hóa giải quyết vấn đề này bằng cách cho phép nhiều hệ điều hành và các ứng dụng chạy trên một máy chủ vật lý hoặc “host” (trạm chủ, máy tính). Mỗi “máy ảo” khép kín được phân lập từ những máy khác, và sử dụng được nhiều tài nguyên mà nó yêu cầu hơn từ máy chủ.

2. Lợi ích của việc ảo hóa :

- Sử dụng đến 80% công suất của máy chủ Giảm yêu cầu về phần cứng với tỷ lệ 10:01 hoặc hơn thế nữa Chi phí cho vốn đầu tư và các hoạt động được cắt giảm một nửa, tiết kiệm hàng năm hơn 1,500 \$ cho mỗi máy chủ ảo hóa

Tính linh hoạt, sẵn sàng cao với mức giá phải chăng

3. Ảo hóa hoạt động như thế nào ? Trung tâm của ảo hóa là “máy ảo” (VM), một nơi chứa phần mềm chạy độc lập với các hệ điều hành và các ứng dụng bên trong đó. Bởi vì một máy ảo là hoàn toàn riêng biệt và độc lập nên nhiều máy ảo có thể chạy đồng thời trên một máy tính duy nhất. Một lớp phần mềm mỏng được gọi là hypervisor tách riêng các máy ảo từ máy chủ, và tự động phân bổ nguồn lực tính toán cho mỗi máy ảo khi cần thiết. Kiến trúc này định nghĩa lại phương trình tính toán để cung cấp:

- Nhiều ứng dụng trên một máy chủ. Mỗi một VM có thể đóng gói toàn bộ máy tính, và nhiều ứng dụng cùng với hệ điều hành có thể chạy trên một máy chủ cùng một lúc
- Sử dụng công suất máy chủ tối đa với số lượng máy chủ tối thiểu. Tất cả các máy tính vật lý được sử dụng hết công suất của nó, cho phép giảm đáng kể chi phí bằng cách triển khai tổng thể ít các máy chủ hơn.
- Ứng dụng nhanh hơn, dễ dàng hơn với nguồn lực dự phòng. Giống như một tệp tin khép kín, máy ảo có thể được thao tác như sao chép hay cắt dán một cách dễ dàng. Điều này mang lại sự đơn giản chưa từng có, tốc độ và sự linh hoạt để cung cấp và quản lý CNTT. Máy ảo thậm chí có thể được chuyển sang một máy chủ vật lý khác trong khi đang chạy, thông qua một quá trình được gọi là chuyển đổi trực tiếp. Bạn cũng có thể ảo hóa các ứng dụng kinh doanh quan trọng để cải thiện hiệu suất, độ tin cậy, khả năng mở rộng, và giảm chi phí.

Hợp nhất các máy chủ Ảo hóa một hoặc hai máy chủ chỉ là khởi đầu. Tiếp theo là tổng hợp một cụm máy chủ thành một nguồn lực tài

nguyên hợp nhất và duy nhất. Ví dụ, thay vì 20 máy chủ chạy với công suất 15% mỗi máy, bạn có thể giảm số lượng phần cứng và các chi phí liên quan đến 4 máy chủ để mỗi máy hoạt động với công suất 80%. Áo hóa máy tính để bàn Giải pháp của chúng tôi cho phép bạn cung cấp một máy tính để bàn áo an toàn như một dịch vụ quản lý cho nhân viên văn phòng từ xa và ở các chi nhánh với nhau. Các giải pháp máy tính để bàn áo của chúng tôi tăng tính linh hoạt kinh doanh, giảm chi phí. Lưu trữ hợp nhất Giải pháp của chúng tôi cu

## Các giao thức đảm bảo nhất quán

1.Giao thức ghi từ xa: Tất cả các thao tác ghi dữ liệu x được chuyển tiếp tới một máy chủ chính cho mục dữ liệu x .Máy chủ này tiếp nhận và xử lý tất cả các thao tác ghi-> cập nhật trên CSDL-> Báo cho các máy chủ khác tiến hành cập nhật dữ liệu-> Máy chủ dự phòng tiến hành cập nhật dữ liệu và xác nhận Các thao tác đọc dữ liệu có thể được xử lý cục bộ Đặc điểm: Máy khách yêu cầu ghi dữ liệu tồn nhiều thời gian để có thể thực hiện thao tác Nếu sử dụng giao thức ghi theo khối : đảm bảo tất cả các tiến trình đều nhìn thấy các thao tác ghi theo cùng một trật tự Sử dụng giao thức ghi không theo khối: tăng hiệu năng nhưng không đảm bảo sự nhất quán trong cập nhật dữ liệu trên các máy chủ dự phòng 2.Giao thức ghi cục bộ: Trong giao thức này một bản sao chính của dữ liệu sẽ di chuyển qua lại giữa các tiến trình yêu cầu thực hiện thao tác ghi. Các thao tác ghi được thực hiện cục bộ .Sau khi cập nhật bản sao chính được hoàn thành, yêu cầu cập nhật dữ liệu sẽ được gửi tới các máy chủ dự phòng.Sau khi tiến hành cập nhật các máy dự phòng gửi thông báo cho máy chủ chính xác nhận cập nhật. Tốn nhiều thời gian để định vị và di chuyển mục dữ liệu 3.Giao thức dựa trên đại biểu: chọn một tập con các bản sao nhỏ nhất đủ để thực hiện các thao tác.Số đại biểu lựa chọn phụ thuộc vào tỉ lệ thao tác đọc ghi và chi phí phương pháp truyền thông giữa các nhóm Định nghĩa số đại biểu đọc và số đại biểu ghi:Số đại biểu đọc phải lớn hơn  $\frac{1}{2}$  số bản sao; tổng đại biểu đọc và ghi phải lớn hơn tổng số đại biểu Thao tác đọc: Thu thập các yêu cầu đọc của nhóm. Nếu đạt đủ điều kiện ( $Nr > N/2$ ) thì đọc dữ liệu từ bản sao cập nhật mới nhất(dựa theo nhãn thời gian) Thao tác ghi: Thu thập các yêu cầu ghi của nhóm đại biểu.Nếu có một bản sao không đáp ứng đủ việc cập nhật -> thay thế bằng bản sao băng bản sao hiện tại Cập nhật tất cả các bản sao thuộc nhóm đại biểu ghi

# Truyền thông khách chủ tin cậy

Truyền thông khách/chủ tin cậy Tính chịu lỗi trong hệ thống phân tán không chỉ tập trung vào các lỗi ở các tiến trình mà còn phải chú ý đến các lỗi truyền thông. Lỗi truyền thông có thể do mất kênh truyền, thất lạc thông tin, quá thời gian, lặp thông điệp... Để khắc phục lỗi truyền thông, người ta thường sử dụng phương pháp truyền tin tin cậy giữa điểm với điểm (unicast) hoặc giữa điểm với nhiều điểm (Multicast). Truyền tin tin cậy giữa điểm với điểm được vận dụng dựa trên các giao thức truyền tin cậy, ví dụ sử dụng giao thức TCP.

1. Truyền thông điểm – điểm Trong hệ phân tán, truyền tin điểm - điểm tin cậy được thiết lập bằng cách sử dụng các giao thức truyền tin có liên kết như TCP. Giao thức TCP che giấu được lỗi bỏ sót bằng cách dùng cơ chế thông báo số tuần tự của đoạn tin và thực hiện truyền lại. TCP không khắc phục được treo hệ thống. Khi hệ thống bị treo thì liên kết nối TCP sẽ bị hủy bỏ. Cách duy nhất để che giấu lỗi đó là hệ thống phải có khả năng tự động tạo một liên kết mới.

2. Các tình huống lỗi trong gọi thủ tục từ xa Quá trình gọi thủ tục từ xa bao gồm nhiều công đoạn và có thể xảy ra các lỗi sau:

- 2.1. Máy trạm không thể xác định được máy chủ: Nguyên nhân gây lỗi có thể do máy chủ và máy trạm dùng các phiên bản khác nhau hoặc do chính máy chủ bị lỗi. Khắc phục bằng cách sử dụng tính năng bắt lỗi trong ngôn ngữ lập trình. Hạn chế của phương pháp này là không phải ngôn ngữ nào cũng hỗ trợ ngoại lệ hay điều khiển tín hiệu. Nếu tự viết một ngoại lệ hay điều khiển tín hiệu thì sẽ phá hủy tính trong suốt.
- 2.2. Mất thông điệp yêu cầu từ máy trạm gửi đến máy chủ: Đây là loại lỗi dễ xử lý nhất: hệ điều hành hay máy trạm stub kích hoạt một bộ đếm thời gian khi gửi đi một yêu cầu. Khi bộ đếm thời gian đã trở về giá trị 0 mà không nhận được thông điệp phản hồi từ máy chủ thì nó sẽ gửi lại yêu cầu đó. Nếu máy trạm nhận thấy có quá nhiều yêu cầu phải gửi lại thì nó sẽ xác nhận rằng máy chủ không hoạt động và sẽ quay lại thành kiểu lỗi "không xác định được máy chủ".
- 2.3. Máy chủ bị lỗi ngay sau khi nhận được yêu cầu từ máy trạm. Bao gồm hai loại: + Loại thứ nhất, sau khi thực hiện xong yêu cầu nhận được thì máy chủ bị lỗi. Khi đó, máy chủ sẽ gửi thông báo hỏng cho máy trạm. Loại thứ hai vừa nhận được yêu cầu từ máy trạm, máy chủ đã bị lỗi ngay, để khắc phục thì máy trạm chỉ cần truyền lại yêu cầu cho. Vấn đề đặt ra lúc này là máy trạm không thể nói cho máy chủ biết yêu cầu nào là yêu cầu được gửi lại. Khi gặp lỗi kiểu này, ở phía máy chủ sẽ thực hiện theo ba cách sau:

- Cách 1:

đợi đến khi nào máy chủ hoạt động trở lại, nó sẽ cố thực hiện yêu cầu đã nhận được trước khi lỗi đó. Như thế RPC thực hiện ít nhất một lần. - Cách 2: máy chủ sau khi được khôi phục nó sẽ không thực hiện yêu cầu nhận được trước khi bị lỗi mà sẽ gửi lại thông báo hỏng cho máy trạm biết để máy trạm gửi lại yêu cầu. Với cách này thì RPC thực hiện nhiều lần nhất. - Cách 3: không thực hiện gì để đảm bảo cá. Khi máy chủ bị lỗi, máy trạm không hề hay biết gì cả. Kiểu này, RPC có thể được thực hiện nhiều lần cũng có thể không thực hiện lần nào. Máy trạm có thể thực hiện theo 4 cách sau: - Cách 1: Máy trạm không thực hiện gửi lại các yêu cầu. Vì thế không biết bao giờ yêu cầu đó mới thực hiện được hoặc có thể không bao giờ được thực hiện. - Cách 2: Máy trạm liên tục gửi lại yêu cầu: có thể dẫn tới trường hợp một yêu cầu được thực hiện nhiều lần. - Cách 3: Máy trạm chỉ gửi lại yêu cầu nào đó khi không nhận được thông điệp xác nhận phản hồi từ máy chủ thông báo đã nhận thành công. Trường hợp này, máy chủ dùng bộ đếm thời gian. Sau một khoảng thời gian xác định trước mà không nhận được xác nhận thì máy trạm sẽ gửi lại yêu cầu đó. - Cách 4: Máy trạm gửi lại yêu cầu nếu nhận được thông báo lỗi từ máy chủ.

2.4. Mất thông điệp phản hồi từ máy chủ gửi trả về máy trạm: Phương pháp khắc phục: Thiết kế các yêu cầu có đặc tính không thay đổi giá trị. Máy trạm đánh số thứ tự cho các yêu cầu, máy chủ sẽ nhận ra được đâu là yêu cầu đã được gửi lại nhò

## Truyền thông nhóm tin cậy

### 1. Giới thiệu truyền thông nhóm tin cậy

§ Là một trong hai phương pháp truyền thông tin cậy để phát hiện lỗi đồng thời khắc phục lỗi truyền thông trong các hệ thống phân tán.  
§ Truyền thông nhóm tin cậy được hiểu là việc một tiến trình muốn gửi thông điệp đến tất cả các tiến trình khác trong nhóm của mình sau khi đã phân nhóm tiến trình. Phải có cơ chế để đảm bảo thông điệp đến được tất cả các thành viên trong nhóm.

### 2. Truyền thông nhóm tin cậy cơ bản

§

## Bài tập các hệ thống phân tán

**Các loại kiến trúc hệ thống phân tán:** - Mô hình phân tầng: là mô hình mà các tầng được phân chia theo chức năng, không có chức năng nào bị trùng lặp trong các tầng, tầng dưới phục vụ các yêu cầu của tầng trên đó và trả kết quả về theo chiều ngược lại. Mô hình này có nhược điểm là tốc độ xử lý chậm do phải qua nhiều tầng. - Mô hình dựa trên đối tượng: các phương thức xử lý nằm ngay tại đối tượng, đối tượng này gọi tới đối tượng kia, các đối tượng hoạt động tương đối độc lập với nhau, thời gian đáp ứng nhanh, tuy nhiên sẽ không phù hợp với các hệ thống phân tán có quy mô lớn mà chỉ phù hợp với hệ thống có mô hình khách/chủ. - Mô hình dựa trên sự kiện: khi có yêu cầu thì các tiến trình sẽ gửi yêu cầu lên một kênh và sẽ được xử lý và trả lại kết quả. - Mô hình dữ liệu tập trung: tất cả dữ liệu sẽ được tập trung tại một nơi, khi có yêu cầu cấp phát dữ liệu thì các tiến trình sẽ lấy dữ liệu ở đây.

### **Mô hình nhất quán FIFO**

Việc sử dụng các bản sao gây ra các hạn chế đó là tính nhất quán dữ liệu của hệ thống bị suy giảm. Do sử dụng bản sao nên có thể xảy ra trường hợp có sự thay đổi trên một dữ liệu mà không cập nhật trên các bản sao của nó. Mô hình nhất quán FIFO hay còn gọi là nhất quán PRAM. Đây là mô hình nhất quán yếu thuộc các mô hình nhất quán lấy dữ liệu làm trung tâm. Do nhất quán FIFO bỏ qua giới hạn về trật tự của bất kỳ thao tác đồng thời nào. Toàn vẹn FIFO thỏa mãn: → Các thao tác ghi bởi một tiến trình đơn phải được tất cả các tiến trình khác nhìn thấy theo cùng một trật tự mà chúng đề ra. → Thao tác ghi dữ liệu của các tiến trình khác nhau có thể được nhìn thấy theo những trật tự khác nhau bởi các tiến trình khác nhau. Ví dụ: Ưu điểm: Đơn giản, dễ cài đặt. Nhược điểm: Sai lệch giữa các bản sao có thể lớn.

## **Giới thiệu về kiến trúc hướng dịch vụ**

Giới thiệu về kiến trúc hướng dịch vụ 1, Khái niệm: Kiến trúc hướng dịch vụ (SOA—Service Oriented Architectural) là một cách tiếp cận để xây dựng các hệ thống phân tán cung cấp các chức năng ứng dụng dưới các dạng dịch vụ tới các ứng dụng người cuối cùng hoặc các dịch vụ khác. 2, Nguyên lý hình thành kiến trúc hướng dịch vụ: - Dịch vụ là nhân tố chủ yếu hình thành lên cấu trúc hướng dịch vụ. - Phân rã các chức năng của hệ thống thành các dịch vụ theo phương pháp từ trên xuống dưới (Top-Down). - Từ các quy trình, chính sách, nguyên lý hay phương pháp thực hiện trong SOA để hướng tới khái niệm dịch vụ. -

Các công cụ được lựa chọn bởi SOA hướng đến việc tạo và triển khai các dịch vụ, ngay cả cơ sở hạ tầng thực thi được cung cấp bởi SOA cũng hướng đến việc thực thi và quản lý các dịch vụ.

3, Mục đích của kiến trúc hướng dịch vụ:

- Hệ thống phần mềm phải luôn thay đổi để đáp ứng những yêu cầu của thị trường.
- Xóa bỏ những rào cản về mặt công nghệ thực hiện, các nền tảng hệ - thống.

Tân dụng nguồn tài nguyên sẵn có của các hệ thống cũ để giảm thiểu chi phí thực hiện.

- Đảm bảo phát triển cũng như bảo trì hệ thống.

4, Đặc điểm của kiến trúc hướng dịch vụ:

- Kiến trúc hướng dịch vụ là một kiến trúc dùng trong các chuẩn mở để biểu diễn các thành phần mềm như là các dịch vụ.
- Cung cấp một cách thức chuẩn hóa cho việc biểu diễn và tương tác với các thành phần phần mềm.
- Các thành phần phần mềm riêng lẻ trở thành các khối cơ bản để có thể sử dụng lại để xây dựng các ứng dụng khác.
- Được sử dụng để tích hợp các ứng dụng bên trong và bên ngoài tổ chức.

5, Các dịch vụ trong kiến trúc hướng dịch vụ:

- a, Phân loại dịch vụ :
  - Dịch vụ cơ bản: - Dịch vụ dữ liệu cơ bản.
  - Dịch vụ logic cơ bản.
  - Dịch vụ kết hợp.
  - Dịch vụ quy trình.
- b, Đặc điểm của dịch vụ :
  - Dịch vụ là yếu tố then chốt trong SOA.
  - Có thể hiểu dịch vụ như là hàm chức năng (modul phần mềm) thực hiện theo quy trình nghiệp vụ nào đó.
  - Các dịch vụ trong SOA có đặc điểm sau:
    - Các dịch vụ là có thể tìm kiếm được.
    - Các dịch vụ có tính liên thông.
    - Các dịch vụ không được gắn kết chặt chẽ với nhau.
    - Các dịch vụ là phức hợp, bao gồm nhiều thành phần, được đóng gói ở mức cao.
    - Các dịch vụ trong suốt về vị trí.
    - Các dịch vụ có khả năng tự hàn gắn

6, Ưu điểm:

- Tái sử dụng phần mềm.
- Nếu gói mã mà tạo thành một dịch vụ có quy mô và kích thước phù hợp sau đó nó có thể được tái sử dụng cho lần kế tiếp.
- Linh hoạt, các dịch vụ có thể được sử dụng trên nền tảng bất kỳ và được viết với ngôn ngữ bất kỳ (ví dụ, ứng dụng Java có thể liên kết với dịch vụ mạng .NET và ngược lại).
- Thậm chí nếu các dịch vụ sẽ không được tái sử dụng, thì họ có thể đưa ra nhiều giá trị nếu họ làm cho hệ thống CNTT chỉnh sửa dễ dàng hơn.

b, Nhược điểm:

- Sự phát triển không đồng bộ.
- Hiểu biết về SOA trong ngành CNTT chưa nhiều.
- Số lượng chuyên gia cao cấp về SOA còn hiếm.

## Mô hình nhất quán

1. Ví dụ Giả sử rằng các trường hợp sau đây xảy ra: • Hàng X được nhận rộng trên các nút M và N • Các khách hàng A viết hàng X đến nút N • Sau một khoảng thời gian t, khách hàng B lần đọc hàng X từ nút M Các mô hình nhất quán có thể xác định xem khách hàng B thấy ghi từ khách hàng A hay không. 2. Các loại Có hai phương pháp để xác định và phân loại các mô hình thông nhất; vấn đề và xem. Vấn đề: Phương pháp vấn đề mô tả các hạn chế để xác định làm thế nào một quá trình có thể phát hành các hoạt động. Xem: Xem phương pháp trong đó xác định thứ tự của các hoạt động có thể nhìn thấy quá trình. Ví dụ, một mô hình nhất quán có thể xác định rằng một quá trình không được phép phát hành một hoạt động cho đến khi tất cả các hoạt động ban hành trước đó được hoàn thành. Mô hình nhất quán khác nhau thực thi các điều kiện khác nhau. Một mô hình nhất quán có thể được coi là mạnh hơn so với người khác nếu nó đòi hỏi tất cả các điều kiện của mô hình đó và nhiều hơn nữa. Nói cách khác, một mô hình với những hạn chế ít hơn có thể được xem xét như là một mô hình nhất quán yếu. 2.1. Tính nhất quán nghiêm ngặt Nhất quán nghiêm ngặt là mô hình nhất quán mạnh nhất. Nó đòi hỏi rằng nếu một quá trình đọc bất kỳ vị trí nhớ, giá trị trả về bởi các hoạt động đọc là giá trị được viết bởi các hoạt động viết gần đây nhất cho vị trí đó. Đối với một hệ thống đơn CPU, mô hình này làm cho cảm giác hoàn hảo. Nhưng đó là hầu như không thể thực hiện các mô hình thông nhất chặt chẽ trong hệ thống phân phối bộ nhớ chia sẻ. Hãy xem xét một tình huống mà có hai bộ vi xử lý, A và B. Processor A viết một giá trị tại một trường thời gian cụ thể và xử lý B lần đọc rằng giá trị tại một thời gian sau đó. Hãy xem xét một hình nón ánh sáng có nguồn gốc ở bộ xử lý A. Nếu bộ xử lý A và xử lý B được đặt liền kề nhau trên một dòng thời gian, thời điểm mà một tia sáng từ nón ánh sáng này có thể chạm mốc thời gian xử lý của B xác định các trường hợp mà xử lý B có thể nhìn thấy giá trị mới của các dữ liệu được viết bởi bộ xử lý A. Nếu bộ xử lý B cố gắng để đọc dữ liệu trước khi thời gian này, nó sẽ đọc các giá trị trước đó của dữ liệu, mặc dù bộ xử lý A đã viết các giá trị mới. 2.2. Tuần tự nhất quán Các mô hình nhất quán tuần tự theo quy định của Lamport (1979) là một mô hình bộ nhớ yếu hơn so với tính nhất quán nghiêm ngặt. Linearizability (còn được gọi là nhất quán nguyên tử) có thể được định nghĩa là sự nhất quán tuần tự với các ràng buộc thời gian thực. 2.3. Tính nhất quán nhân quả Nhất quán quan hệ nhân quả có thể được coi là một mô hình suy yếu của sự nhất quán tuần tự bằng cách loại các sự kiện vào những quan hệ nhân quả có liên quan và những người không được. Nó định nghĩa mà chỉ ghi các hoạt động

có liên quan nhau phải được nhìn thấy trong cùng một thứ tự của tất cả các quy trình. 2.4. Bộ vi xử lý nhất quán Mô hình xử lý thống nhất là tương tự như xe đẩy mô hình nhất quán với một điều kiện mạnh rằng định nghĩa tất cả ghi vào trí nhớ cùng phải được nhìn thấy trong các trình tự tương tự của tất cả các quá trình khác. Quy trình thống nhất là yếu hơn so với tính nhất quán tuần tự nhưng mạnh rằng mô hình nhất quán PRAM. 2.5. PRAM Tính nhất quán (còn được gọi là nhất quán FIFO) Trong PRAM Tính nhất quán (pipeline RAM), tất cả các quá trình xem các hoạt động của một quá trình duy nhất trong thứ tự mà chúng được phát hành bởi quá trình đó, trong khi các hoạt động do các quá trình khác nhau có thể được xem trong thứ tự khác nhau từ các quá trình khác nhau. Nhất quán PRAM là yếu hơn so với quá trình thống nhất. 2.6. Cache Consistency Thống nhất bộ nhớ cache yêu cầu tất cả ghi các hoạt động đến vị trí nhớ cùng được thực hiện trong một số tuần tự. Nhất quán Cache là yếu hơn so với quá trình thống nhất và không thể so sánh với sự nhất quán PRAM. 2.7. Chậm nhất quán Trong quán chậm, nếu một quá trình đọc một giá trị trước bằng văn bản đến một vị trí bộ nhớ, nó có thể không phải sau đó đọc những giá trị trước đó từ vị trí đó. Ghi thực hiện theo một

## **Trình bày các tiêu chí về hiệu năng đối với hệ thống phân tán.**

Các tiêu chí về hiệu năng là:

- + Giảm trao đổi trên mạng
- + Dễ dàng xác định duy nhất thực thể khi chia sẻ tài nguyên trong hệ thống
- + Đồng bộ
- + Xử lý song song
- + Đảm bảo tính trong suốt
- + Giảm tương tranh
- + Hệ thống tự khắc phục lỗi
- + Đảm bảo tính bí mật, toàn vẹn
- + Đảm bảo tính đồng nhất dữ liệu, tăng độ tin cậy và tính sẵn sàng của hệ thống

## **Vì sao mỗi giao thức trong mô hình phân tầng phải có phần thông tin điều khiển riêng của giao thức đó.**

Vì:

1. Chức năng của các tầng trong 1 hệ thống là không trùng lắp nên để tránh đưa thông tin điều khiển nhầm tầng dẫn đến lỗi thì mỗi tầng có riêng 1 phần thông tin điều khiển của mình.
2. Do cấu trúc phân tầng không giới hạn về số tầng nên có nhiều tầng dẫn đến hệ thống chạy chậm mà dùng chung phần thông tin điều khiển sẽ dẫn tới hệ thống sẽ càng chậm hơn.
3. Do tính độc lập của mỗi tầng cao khi mà dùng chung phần thông tin điều khiển sẽ làm cho xung đột với tầng khác.
4. Dùng chung phần thông tin điều khiển dẫn tới tầng tốc độ xử lý cao sẽ làm tê liệt tầng tốc độ xử lý thấp.
5. Do mỗi tầng dùng đơn vị dữ liệu khác nhau nên khi dùng chung phần thông tin điều khiển cần phải đồng bộ dữ liệu sẽ làm chậm tốc độ xử lý.
6. Do đường truyền không hoàn hảo nên bên thu phải có khả năng phản hồi và bên nhận phải có khả năng trả lời gói tin đã đúng chưa dẫn đến cần có phần thông tin điều khiển để điều khiển trả lời.
7. Không phải mỗi tầng đều nhận gói tin có độ dài tùy ý nên cần phải có phần điều khiển riêng để chia nhỏ gói tin có độ dài phù hợp.
8. Để truyền thông gói tin chuyển xuống qua các tầng cần phải thêm thông tin điều khiển vào phần đầu và gửi về thì ngược lại nên đến nhận ra đặc trưng của mỗi tầng thì cần phải có phần thông tin điều khiển riêng (PDU = PCI +SDU).
9. Mỗi tầng không quan tâm đến chi tiết mà dịch vụ đó thực hiện nên để làm việc chính xác hơn cần phải có phần điều khiển riêng.

# **Single Instruction Multiple Data (SIMD) và Multiple Instruction Multiple Data (MIMD) là hai kiến trúc tính toán song song khác nhau sử dụng nhiều bộ xử lý và đôi khi là nhiều máy tính để xử lý dữ liệu.**

Kiến trúc SIMD và MIMD thực hiện các chức năng cơ bản giống nhau, nhưng chúng khác nhau cả thực tế và kỹ thuật.

Máy tính SIMD thực hiện một hành động đồng nhất cùng một lúc trên nhiều mảnh dữ liệu, trong đó có truy xuất, tính hoặc lưu trữ thông tin. Một ví dụ là truy xuất nhiều tập tin cùng lúc. Bộ xi xử lý với bộ nhớ cục bộ có chứa các dữ liệu khác nhau thực hiện lệnh giống nhau theo cách đồng bộ hóa.

Máy tính MIMD thực hiện nhiều hành động cùng một lúc trên nhiều mảnh dữ liệu. Một ví dụ là thực hiện các tính toán toán học khác nhau - chẳng hạn như phép cộng và phép nhân - đồng thời để giải quyết một vấn đề toán học phức tạp với nhiều thành phần riêng biệt. Máy tính MIMD có thể có hoặc không được đồng bộ và ngày càng phổ biến hơn so với máy tính SIMD.

Sự khác biệt thực tế:

SIMD thường được dùng cho vấn đề cần nhiều tính toán với bộ xử lý thực hiện thao tác tương tự song song. MIMD thường được dùng cho các vấn đề mà thuật toán chia thành các phần riêng biệt và độc lập, với mỗi phần giao cho một bộ xử lý khác nhau để giải quyết đồng thời.

Sự khác biệt kỹ thuật:

Bộ xử lý SIMD thường đơn giản, nhỏ hơn, rẻ hơn và nhanh hơn bộ xử lý MIMD, nhưng MIMD có khả năng thao tác phức tạp hơn rất nhiều. SIMD cũng có thể hoàn thành các thao tác của SIMD tuy nhiên nó sẽ tốn nhiều thời gian. Bộ xử lý SIMD phải thực hiện thao tác phức tạp liên tục, trong khi bộ xử lý MIMD có thể làm điều này đồng thời.

## =====Hệ PT

Hệ phân tán

Chương I : Tổng quan về hệ phân tán.

I.I Định nghĩa.

Có nhiều định nghĩa về hệ phân tán

Định nghĩa 1: Hệ phân tán là tập hợp các máy tính tự trị được kết nối với nhau

bởi một mạng máy tính và được cài đặt phần mềm hệ phân tán.

Định nghĩa 2: Hệ phân tán là một hệ thống có chức năng và dữ liệu phân tán

trên các trạm (máy tính) được kết nối với nhau bởi một mạng máy tính.

Định nghĩa 3: Hệ phân tán là một tập các máy tính độc lập giao tiếp VỚI nhau

dùng như một hệ thống thống nhất, toàn vẹn.

Như vậy, có thể nói : Hệ phân tán = mạng máy tính + phần mềm hệ phân tán.

Phân loại hệ phân tán:

Trước đây, hệ phân tán được chia thành ba loại : hệ điều hành hệ phân tán, cơ

sở dữ liệu hệ phân tán và các hệ thống tính toán hệ phân tán.

Ngày nay, hệ phân tán được phân chia như sau:

- Hệ phân tán mang tính hệ thống: hệ điều hành phân tán.
- Hệ phân tán mang tính ứng dụng: các hệ thống truyền tin phân tán.

1.2 Mục tiêu của hệ phân tán.

a. Kết nối người sử dụng và tài nguyên

CHia quyết bài toán chia sẻ tài nguyên trong hệ thống (resource sharing).

b. Tính trong suốt

Ấn giấu sự rời rạc và những nhược điểm nêu có của hệ phân tán đối với người

sử dụng (end-user) và những nhà lập trình ứng dụng (application programmer).

Theo tiêu chuẩn ISO cho hệ phân tán ISO / IS / 10746 tên là “Open distributed

processing reference model” 1995 đã cụ thể hóa tóm tắt dạng trong suÔt:

Trong suốt truy cập (Access transparency): che giấu sự khác biệt về cách biểu diễn và cách truy cập tài nguyên.

Trong suốt về VỊ trí (Location transparency): che giấu vị trí của tải nguyên. Hai dạng trong suốt vừa trình bày được gọi chung là trong suốt mạng (network transparency).

Trong suốt di trũ (Migration transparency): che giấu khả năng chuyên vị trí của tải nguyên.

Trong suốt về việc định vị lại (Relocation transparency): che giấu việc đi chuyên của tài nguyên khi đang được sử dụng.

Trong suốt nhân bản (Replication transparency): che giấu tình trạng tình trạng sử dụng bản sao của tài nguyên.

Che giấu sự che sẻ tài nguyên tương tranh (Concurrency transparency).

Trong suốt sự cố (Failure transparency): che giấu lỗi hệ thống nếu có.

Trong suốt khả năng di chuyên tài nguyên (Persistence transparency): che giấu việc di chuyên tài nguyên từ bộ nhớ ngoài vào bộ nhớ trong và ngược lại.

c. Lĩnh mở (Openness).

Hệ phân tán được gọi là mở nếu nó cung cấp các dịch vụ theo các quy tắc  
chuẩn mô tả cú pháp và ngữ nghĩa của dịch vụ đó.  
Thông thường trong hệ phân tán các dịch vụ thường đặc tả qua các giao diện

bằng ngôn ngữ đặc tả giao diện (Interface Definition Language-IDL). Vì thế chỉ quan tâm đến cú pháp. Nó cho phép các dịch vụ khác nhau cùng chung sống. Nếu các giao diện của hệ phân tán được đặc tả đầy đủ và đúng đắn.

Xét hai khái niệm của hệ phân tán là khái niệm liên tác (Interoperability) và khái niệm chuyên mang (portability).

Liên tác: các cài đặt của các hệ thống hoặc thành phần hệ thống từ các nhà sản xuất khác nhau có thể làm việc với nhau thông qua liên tác.

Chuyên mang: nhờ chuyển mang mà một ứng dụng được phát triển cho hệ phân tán A có thể thực hiện không cần thay đổi gì trên một hệ phân tán B khác, với điều kiện được cài đã cùng giao diện như A

#### d. Tính co giãn (Scalability)

Một hệ phân tán được gọi là có tính co giãn nếu nó thích nghi với sự thay đổi quy mô của hệ thống. Thể hiện trên các khía cạnh sau:

- Dễ bổ sung người sử dụng và tài nguyên hệ thống.
- Khi hệ thống thay đổi quy mô về mặt địa lý dẫn đến sự thay đổi về vị trí địa lý của người sử dụng và các tải nguyên.
- Hệ thống có thay đổi quy mô về quản trị.

Nếu hệ phân tán có tính co giãn thường ảnh hưởng đến hiệu năng của hệ thống (hiệu năng của hệ thống là hiệu quả năng lực hoạt động của đối tượng).

Có ba giải pháp phổ dụng để giải quyết vấn đề co giãn của hệ phân tán:

- Ân giầu

- Phân tán: phân nhỏ thành phần hệ thống và phân bổ chúng trên phạm vi của hệ thống (quản lý phân cấp). Ví dụ DNS xác định theo cách phân cấp miền lớn thành các miền con. Với phương pháp này sẽ giải quyết được vấn đề khi thêm người dùng hay tải nguyên vào hệ thống.
- Nhân bản: nhân bản một thành phần nào đó của hệ thống. Ví dụ tài liệu đặt tại các vị trí khác nhau trong hệ thống.

### 1.3 Các khái niệm phân cứng.

#### a. Phân loại máy tính.

Có hai loại máy tính:

- Các loại máy tính có chia sẻ bộ nhớ (Shared memory): các loại máy đa xử lý (multiprocessor).
- Các máy tính không chia sẻ bộ nhớ (Private memory): các hệ thống multicomputers

Trong mỗi loại lại chia tiếp theo mạng. kết nối bus - based chỉ có một đường kết nối và switch - base có nhiều đường kết nối từ máy này sang máy khác

Hình 1: Hai cách tổ chức vi xử lý và bộ nhớ trong hệ máy tính phân tán.

#### b. Hệ thuần nhất / hệ không thuần nhất.

Hệ thống thuần nhất: mạng máy tính cùng sử dụng một công nghệ, các bộ xử lý

là như nhau, truy cập đến cung một bộ nhớ giống nhau. Thường dùng trong hệ thống có tính toán song song.  
Hệ không thuần nhất: những máy tính khác nhau kết nối với nhau.

#### 1.4 Các khái niệm phần mềm.

##### a. DOS (distributed OS).

Là hệ điều hành cho các hệ multiprocessor và các hệ homogenous multicomputer.

Mục tiêu là ân giấu và cung cấp các dịch vụ quản trị tải nguyên.

Đặc điểm là các dịch vụ có thể được thực hiện bởi các lời triệu gọi từ xa.

##### Hình 2. Cấu trúc chung của DOS

##### b. NOS (Network OS).

Là hệ điều hành cho các hệ thống heterogenous multicomputer (LAN, WAN).

Mục tiêu của NOS là cung cấp các dịch vụ từ xa.

##### Hình 3 . Cấu trúc chung của NOS

##### c. Middleware.

Lớp tảng phụ nằm giữa tảng dịch vụ của NOS và tảng ứng dụng phân tán.

##### Hình 4. Cấu trúc chung của một hệ middleware

#### 1.5 Mô hình client - server

##### a. Tổng quan về mô hình Client - server.

Mô hình client - server trong một hệ phân tán được phân chia thành hai nhóm chính là nhóm các server và nhóm các client. Nhóm các server chứa các dịch

vụ đặc biệt. Nhóm các client là nhóm gửi yêu cầu đến server để được sử dụng các dịch vụ đó trên server.

Mô hình tương tác tổng quát giữa client và server:

Hình 5 Mô hình tương tác chung giữa một client và một server

b Phân tầng các ứng dụng.

Việc phân định rạch rời chức năng của client và server đến Ø81Ò cũng rất khác biệt và không thuận nhất. Do đó người ta đưa ra ý tưởng là chia thành ba mức chức năng:

User - interface level: bao gồm các chương trình cung cấp giao diện cho phép người sử dụng tương tác với chương trình ứng dụng.

Processing level: làm nhiệm vụ xử lý các tác vụ của người dùng trên cơ sở dữ liệu

Data level: gồm các chương trình duy trì các dữ liệu mà các chương trình ứng dụng xử lý.

Chương 2: Truyền thông.

(Communication)

2.1 Các giao thức phân tầng (Layered protocols).

Một trong những mô hình phân tầng thông dụng nhất hiện nay là mô hình OSI 7 tầng. Mỗi tầng có các giao thức riêng cho nó.

- Tầng ứng dụng.

- Tầng trình diễn.

- Tầng phiên.
- Tầng vận chuyển.
- \_ Tầng mạng.
- Tầng liên kết dữ liệu.
- Tầng vật lý:

Một cải tiến trong hệ phân tán là gộp tầng trình diễn và tầng phiên thành một tầng mới là tầng middle ware. Do đó ta cũng phải xây dựng các giao thức tương ứng cho tầng middleware này.

Có 4 mô hình dịch vụ middleware mà ta sẽ xét lần lượt sau đây:

- Gọi thủ tục từ xa RPC (Remote Procedure Call).
- Triệu gọi đối tượng từ xa (Remofte ObJect Invocation)
- Middleware hướng thông điệp (Message - orlented Middleware)
- Middleware hướng dòng (Stream - orIented Middleware)

## 2.2 Gọi thủ tục từ xa (Remote procedure call - RPC).

TT Tông quan về RPC.

Khi một tiến trình trên máy A muốn thực hiện một thủ tục nào đó nằm trên một máy B khác thì nó sẽ thực hiện một lời gọi thủ tục từ xa tới máy B. Thủ tục đó sẽ được thực hiện ở máy B dựa trên các tham SỐ được truyền đến từ máy A và kết quả sẽ được truyền trở lại cho máy A tương ứng.

Trong mô hình client - server thì lời gọi thủ tục từ xa được thực hiện qua các bước sau:

- Tiên trình muốn thực hiện thủ tục ở máy client sẽ gọi client stub.
- Client stub sẽ tạo một bản tin và có lời gọi đến hệ điều hành của client đó.
- Hệ điều hành của máy client sẽ gửi bản tin đó tới hệ điều hành của máy SGTVENT.
- Hệ điều hành của server sẽ gửi bản tin tới server stub.
- Server stub lấy các thông tin của gói tin và gọi server tương ứng.
- Server thực hiện công việc được yêu cầu và trả kết quả về cho server stub.
- Server stub đóng gói kết quả đó vào bản tin rồi gọi hệ điều hành của server đó.
- Hệ điều hành của máy server này sẽ gửi bản tin kết quả đó hệ điều hành của máy client.
- Hệ điều hành của máy client sẽ gửi bản tin cho client stub.
- Client stub sẽ mở gói tin kết quả và trả về cho client.

Trong đó, client stub và server stub ở máy client và server là thành phần nhằm giảm nhẹ công việc cho client và server, làm cho hệ thống hoạt động một cách trong suốt hơn.

Hình 6. RPC giữa một client và server

#### 2.2.2 Xét chi tiết các thao tác RPC.

Đóng gói các tham số: VIỆC đóng gói các tham số để chuẩn bị truyền đi do

client stub thực hiện. Client stub sẽ sắp xếp các tham số và đưa vào hàng đợi và quá trình này được gọi là parameter marshaling. Các tham số được truyền đi giúp cho server hiểu được công việc mình cần thực hiện tương ứng là gì để xác định lời gọi đến thủ tục thích hợp.

Truyền tham số: Việc truyền tham số từ client tới . Có hai cách truyền: truyền tham biến và truyền tham trị.

- Truyền tham trị: các tham số được truyền đi là các giá trị cụ thể. Các thủ tục được gọi đến sẽ coi các tham biến được truyền kiêm tham trị như là các biến được khởi tạo cục bộ, có thể thay đổi giá trị nhưng lại không ảnh hưởng tới giá trị gốc trong lần gọi Sau. Vấn đề đặt ra khi truyền tham trị là yêu cầu giữa các máy phải có sự đồng nhất về VIỆC biêu diễn dữ liệu và các kiểu dữ liệu.

- Truyền tham biến: các tham số được truyền đi là con trỏ hay biến chứa địa chỉ của nơi chứa giá trị thực của chúng. Các thủ tục được gọi sẽ căn cứ vào địa chỉ này để tham chiếu đến giá trị khi tính toán. Khi giá trị này bị thay đổi trong khi thực hiện thủ tục thi sẽ được thông báo cho client và các lần gọi sau sẽ dùng giá trị mới đó.

## 2.3 Các mô hình RPC mở rộng.

### 2.3.1 RPC dị bộ (Asynchronous RPC).

Tư tưởng thực hiện là: client gửi tới server lời gọi thủ tục và chờ bản tin chấp nhận từ server. Phía server sẽ gửi bản tin chấp nhận về cho client thông báo đã

nhận được yêu cầu và bắt đầu thực hiện yêu cầu RPC đó. Lúc này client sẽ tiếp tục thực hiện công việc của mình mà không chờ kết quả từ server như ở RPC truyền thống.

Hình 7. RPC dị bộ.

#### 2.3.4.6. đồng bộ trễ (Deferred synchronous RPC):

Thực hiện hai lời gọi, một từ client và một từ server.

Client gửi tới server lời gọi thủ tục và chờ bản tin chấp nhận từ server. Phía server sẽ gửi bản tin chấp nhận về cho client thông báo đã nhận được yêu cầu và bắt đầu thực hiện yêu cầu RPC đó. Lúc này client sẽ tiếp tục thực hiện công việc của mình. Khi thực hiện thủ tục xong, server sẽ thực hiện lời gọi tới client để báo nhận lũy kết quả. Client thực hiện ngắt, nhận kết quả và gửi lại cho server bản tin thông báo đã nhận kết quả thành công.

Hình §. RPC đồng bộ trễ.

#### 2.3.3 RPC đơn tuyến (one-way RPC).

Sau khi thực hiện lời gọi thủ tục từ xa tới server, client không chờ đợi thông báo nhận yêu cầu thành công từ server mà tiếp tục thực hiện ngay các công việc khác của mình. Đó là RPC đơn tuyến.

#### 2.4 Triệu gọi đối tượng từ xa (Remote Object Invocation).

##### 2.4.1 Đối tượng phân tán (Distributed object ).

Một đối tượng phân tán gồm các thành phần sau:

- State: là các dữ liệu đã được đóng gói.

- Method: là các thao tác có thể thực hiện trên dữ liệu.
- Interface: là nơi để giao tiếp với các phương thức của đối tượng.  
Nói cách khác, các phương thức sẵn sàng thông qua Interface.

Một đối tượng có thể thực thi nhiều interface và cũng có thể có nhiều đối tượng cùng thực thi một Interface giống nhau.

Sự độc lập giữa các interface và các đối tượng thực thi interface cho phép ta có thể đặt một interface vào một máy nào đó trong khi chính bản thân đối tượng có thể cư trú ở máy khác.

Hình 9. Đối tượng phân tán.

2.4.2 Các bước thực hiện triệu gọi đối tượng từ xa.

Hình 10. Triệu gọi đối tượng từ xa.

Khi cần triệu gọi các phương thức từ xa, client sẽ gửi yêu cầu đến proxy - một thể hiện của interface.

Proxy sẽ marshal (sắp xếp và đưa vào hàng theo thứ tự) các phương thức được yêu cầu vào một bản tin rồi gửi cho hệ điều hành của máy client.

Hệ điều hành của client sẽ gửi bản tin yêu cầu đó đến hệ điều hành của server.

Hệ điều hành server nhận bản tin và chuyên cho skeleton (giống server stub của KEC).

Skeleton sẽ unmarshal bản tin nhận được để gửi đến interface của đối tượng có phương thức tương ứng.

Đối tượng thực thi phương thức rồi trả kết quả về cho skeleton.

Skeleton marshal kết quả nhận được rồi gửi trả về cho hệ điều hành của client.

Hệ điều hành của client nhận bản tin kết quả rồi chuyên tới cho proxy.

Proxy unmarshal bản tin đó rồi chuyên kết quả về cho client.

Chú ý là cả client và server đều sử dụng interface giống nhau.

Một số các đối tượng

Compile - time object: là các đối tượng trong các ngôn ngữ lập trình hướng đối tượng. Nó được định nghĩa như là một mẫu của class.

Runtime object

Persistent Object - đối tượng kiên trì: là đối tượng vẫn tồn tại ngay cả khi nó không tồn tại trong không gian địa chỉ của tiến trình nào trên SFTVFI.

Transient object - đối tượng tức thời: là đối tượng chỉ tồn tại khi server gọi đến

nó, sau khi dùng xong nó sẽ được giải phóng.

Triệu gọi phương thức từ xa (EMI - remote method Invocation)

Sau khi đã triệu gọi một đối tượng từ xa, client có thể triệu gọi từ xa phương thức của đối tượng đó.

Có hai phương pháp triệu gọi phương thức từ xa là: triệu gọi phương thức từ xa động và triệu gọi phương thức từ xa tĩnh.

Triệu gọi phương thức từ xa động: khi cần gọi đến một phương thức mới xác định Interface đang dùng trong lời triệu gọi từ xa đó. Vì thế khi interface thay đổi, các chương trình ứng dụng không cần phải biên dịch lại.

Triệu gọi phương thức từ xa tính: các interface được xác định trước.

Các

chương trình ứng dụng không thích ứng được khi interface hiện hành thay đổi.

Nếu interface hiện tại có sự thay đổi thì các chương trình ứng dụng

phải được

bên dịch lại mới có thể hiểu

ME on) Truyền thông hướng thông điệp (Message - oriented communication).

### 2.5.1 Các loại truyền thông cơ bản

Truyền thông kiên trì (Persistent communication): Thư điện tử là một ví dụ

minh họa rõ nét cho khái niệm truyền thông kiên trì. Khi một trạm muốn gửi

bản tin đi trên mạng, nó sẽ gửi bản tin đó đến interface của máy mình. Qua bộ

nhớ đệm, bản tin đó được truyền đi trong mạng cục bộ để đến mail server cụ

bộ. Mail server này tạm thời lưu trữ bản tin đó vào bộ nhớ đệm của mình, xác

định địa chỉ trạm đích, rồi gửi tới server cục bộ của trạm đích tương ứng (có

thể đi qua nhiều mail server trung gian khác). Tới mail server cuối cùng, bản

tin lúc này sẽ được lưu lại trước khi phát cho trạm đích tương ứng.

Truyền thông nhất thời (Transient communication): bản tin gửi đi chỉ được lưu

lại trong phiên trao đổi đó. Khi phiên trao đổi đã hoàn thành hoặc khi kết nối bị

hủy bỏ thì các bản tin đó cũng bị hủy bỏ trên các server. Do đó, vì một lý do

nǎo đó mà một server trung gian không thể chuyển tiếp bản tin đi được thì bản

tin này sẽ bị hủy bỏ.

Truyền thông đồng bộ (Synchronous communication): khi trạm gửi

gửi đi một

bản tin thì nó sẽ ở trạng thái khóa (blocked) cho đến khi nhận được thông báo  
bản tin đó đã đến đích thành công.

Truyền thông dị bộ (Asynchronous communication): khi trạm gửi gửi đi bản tin, nó sẽ tiếp tục thực hiện công việc của mình. Điều này cũng có nghĩa là bản tin đó được lưu lại trên bộ nhớ đệm của trạm gửi hoặc của server cục bộ.

### 2.5.2 Một số loại truyền thông hỗn hợp.

Truyền thông dị bộ, kiên trì: bản tin được lưu trữ lâu dài hoặc là ở bộ nhớ đệm của trạm gửi hoặc là trên server truyền thông đầu tiên mà bản tin đó tới. Ví dụ hệ thống thư điện tử.

Truyền thông đồng bộ, kiên trì: bản tin được lưu trữ lâu dài ở trạm nhận, trạm gửi sẽ ở trạng thái blocked cho đến khi bản tin được lưu trữ ở bộ nhớ đệm trạm nhận.

Truyền thông dị bộ, nhất thời: sau khi lưu trữ bản tin cần gửi ra bộ nhớ đệm của máy mình, trạm gửi sẽ tiếp tục thực hiện công việc của mình. Cùng lúc, bản tin sẽ được truyền tới trạm nhận. Khi bản tin đến được trạm nhận đó lại không làm việc, khi đó quá trình truyền thông bị hủy bỏ.  
Truyền thông đồng bộ, nhất thời: bản tin không được lưu trữ lâu dài. Khi gửi đi một bản tin, trạm gửi sẽ chờ bản tin bảo đã nhận thành công của trạm nhận gửi về mới thực hiện tiếp công việc của mình.

## Hình 11. Một số dạng truyền thông.

### 2.6 Truyền thông hướng dòng (stream-oriented communication).

### 2.6.1 Một số khái niệm cơ bản.

Medium (số nhiều là media) : chỉ các phương tiện dùng để truyền thông tin như các thiết bị lưu trữ, đường truyền, các phương tiện hiển thị...

Continuous media: quan hệ thời gian giữa các mục là yếu tố cơ bản để thông dịch đúng ngữ nghĩa thực sự của dữ liệu.

Discrete media: quan hệ thời gian không còn là yếu tố cơ bản để thông dịch đúng dữ liệu.

Data stream: là một chuỗi các đơn vị dữ liệu. Với data stream thì thời gian là yếu tố quyết định. Để kiểm soát thời gian người ta đưa ra ba phương thức truyền sau:

Truyền dị bộ (asynchronous transmission mode): các mục dữ liệu truyền tuần tự và không có ràng buộc thời gian đối với việc truyền.

Truyền đồng bộ (synchronous transmission mode): quy định trước độ trễ tối đa cho mỗi đơn vị dữ liệu trong data stream.

Truyền đăng thời (Isochronous transmission mode): quy định độ trễ lớn nhất và nhỏ nhất cho mỗi đơn vị dữ liệu trong data stream. Cách truyền này đóng một vai trò quan trọng trong việc trình diễn audio và video.

Dòng đơn (simple stream) là dòng chỉ gồm một chuỗi đơn vị dữ liệu.

Đồng phức (complex stream): bao gồm nhiều chuỗi đơn vị dữ liệu khác nhau.  
Mỗi chuỗi này được gọi là một dòng con (sub stream).

## 2.6.2 QoS - chất lượng dịch vụ.

Chất lượng dịch vụ QoS liên quan đến các vấn đề sau:

Bảng thông yêu cầu, tốc độ truyền, trễ..

LoSS SEnSItIVIty: kết hợp cùng với loss interval cho phép ta xác định được tốc độ mất mát thông tin có thể chấp nhận được.

Burst loss sensitivity: cho phép xác định bao nhiêu đơn vị dữ liệu liên tiếp có thể bị mất.

Minimum delay noticed: xác định giới hạn thời gian trễ trên đường truyền cho phép để bên nhận không nhận biết được là có trễ.

Maximum delay variation: xác định độ trễ (jitter) rung lớn nhất cho phép.

Quality of guarantee: chỉ số lượng các dịch vụ yêu cầu cần phải có.

## 2.6.3 Đồng bộ các dòng.

Có hai loại đồng bộ:

Đồng bộ đơn giản: thực hiện đồng bộ giữa dòng trễ và dòng liên tục.  
Ví dụ

trong việc trình diễn slide có kèm âm thanh. Dòng hình ảnh slide là dòng trễ  
còn dòng âm thanh là dòng liên tục, phải đồng bộ hai dòng này để thu được kết  
quả trình diễn như ý muốn.

Đồng bộ phức tạp: là việc đồng bộ giữa các dòng dữ liệu liên tục. Ví  
dụ trong  
việc xem phim trực tuyến, cả dòng âm thanh và dòng hình ảnh đều là  
các dòng  
liên tục cần phải được đồng bộ.

Các kỹ thuật đồng bộ: có hai kỹ thuật đồng bộ

Kĩ thuật đơn giản: dựa trên việc đồng bộ các thao tác đọc ghi trên các dòng dữ liệu sao cho phù hợp với các yêu cầu thời gian cho trước và các ràng buộc về đồng bộ.

Hình 12. Đồng bộ đơn giản.

Kĩ thuật phức tạp: đồng bộ trên mỗi trường mạng dựa trên cả việc đồng bộ giữa bên nhận và bên gửi.

Hình 13. Đồng bộ phức tạp

Chương 3 : Tiến trình

(Processes)

3.1 Luồng (Thread).

Tiến trình (Process) là chương trình đang được thực hiện, nó coi tính trong suốt là quan trọng.

Luồng (Thread): là một hay một phần chương trình đang thực hiện, nó coi hiệu năng là quan trọng.

Lời gọi hệ thống (System call): là tập lệnh mở rộng do hệ điều hành cung cấp xác định giao diện giữa hệ điều hành và các chương trình người sử dụng.

Blocking System call: là lời gọi hệ thống mà sau khi được gọi bởi tiến trình người sử dụng thì tiến trình này bị dừng lại cho đến khi thực hiện xong lời gọi hệ thống.

Non - Blocking System call: sau khi gọi, điều khiển được trả lại cho tiến trình

gọi và tiến trình này tiếp tục thực hiện song song với lời gọi hệ thống.

Đa luồng (Multi thread): áp dụng cho mô hình client/server được gọi là multithread server và multithread client. Với mô hình này giúp đơn giản hóa khi lập trình cho server đồng thời cũng tăng khả năng xử lý song song, làm tăng hiệu năng của hệ thống.

Có ba phương pháp tiếp cận để xây dựng một server:

Đơn luồng (single - threaded server) : non - parallelism, blocking system call.

Đa luồng (multi - threaded server) : parallelism, blocking system call.

Máy trạng thái hữu hạn (Finite State Machine): parallelism, non - blocking system call.

### 3.2 Di trú mã.

Lý do cần phải di trú mã: để tăng hiệu năng và độ linh hoạt của hệ thống do

việc di chuyển của các tiến trình đang thực hiện là rất khó khăn.  
Một tiến trình bao gồm :

Phần mã (Code Segment): chứa tập các lệnh của tiến trình đang thực hiện.

Phần tài nguyên (Resource Segment): chứa các tham chiếu đến tất cả các tài nguyên bên ngoài mà tiến trình đang cần

Phần thực thi (Execution segment): chứa các trạng thái thực thi hiện hành của tiến trình.

Các mô hình di trú mã:

## Hình 14 Các mô hình di trú mã.

Weak mobility: chỉ truyền phần mã và một số các dữ liệu khởi động của tiến trình. Đặc tính của mô hình này là một chương trình được truyền đi luôn được bát đầu từ trạng thái khởi động, chỉ yêu cầu máy đích có thể thực thi yêu cầu (code) đó

Strong mobility: truyền cả phần mã và phần thực thi. Đặc điểm của mô hình này là một tiến trình đang chạy có thể được dừng lại rồi chuyển đến một máy khác và tiếp tục thực hiện tiếp tiền trình đó —> khó thực thi hơn.

Sender Initiated migration (di trú được khởi tạo từ phía gửi) : Di trú được khởi động từ máy mà phần code của tiến trình được lưu trữ hoặc đang thực hiện. Di trú này hoàn thành khi upload chương trình.

Receiver Initiated migration (di trú được khởi tạo từ phía nhận) : Di trú mã ban đầu từ máy tính.

Di trú được khởi tạo từ phía nhận thực thi đơn giản hơn di trú được khởi tạo từ phía gửi.

### 3.3 Tác tử mềm.

#### 3.3.1 Định nghĩa và phân loại:

Định nghĩa: Tác tử là một tiến trình tự trị có khả năng phản ứng, trao đổi, cộng tác với các tác tử khác trong môi trường của nó.

Phân loại theo khái niệm di trú hóa:

Tác tử di động (mobile agent): là một tác tử đơn giản có khả năng di chuyển giữa các máy khác nhau. Trong di trú mã, các tác tử di động thường yêu cầu hỗ trợ cho mô hình di động mạnh mặc dù là không cần thiết. Yêu cầu này đến từ thực tế là các tác tử là tự trị và có ảnh hưởng lẫn nhau và với môi trường của chúng. Sự di chuyển một tác tử đến máy khác khó có thể được thực hiện nếu không xét đến trạng thái thực thi của nó. Tính di động là đặc tính chung của các tác tử.

Tác tử thông minh (Intelligent agent): là tác tử dùng để quản lý thông tin từ nhiều nguồn khác nhau. Việc quản lý thông tin bao gồm việc sắp xếp, lọc, thu thập... Vì các tác tử này thao tác trên thông tin từ những nguồn vật lý khác nhau nên chúng đóng vai trò rất quan trọng.

### 3.3.2 Công nghệ tác tử.

#### Hình 15. Mô hình agent platform của FIPA

ACL (Agent Communication Language): Truyền thông giữa các tiền trình tuân thủ theo giao thức truyền thông mức ứng dụng ACL. ACL message bao gồm phần header và nội dung. Phần header chứa trường để xác định mục đích của thông điệp, cùng với trường để xác định người gửi và người nhận. Cũng như các giao thức truyền thông, phần nội dung được tách riêng. ACL không định khuôn dạng hay ngôn ngữ thể hiện nội dung thông điệp.

ACC: Một thành phần quan trọng trong nền tác tử là kênh truyền thông tác tử - ACC. Trong hầu hết các mô hình cho hệ thống đa tác tử, các tác tử truyền

thông bằng cách trao đổi thông điệp. Mô hình FIPA cũng để cho một ACC quản lý việc truyền thông giữa các agent platform khác nhau. Cụ thể, ACC là nguyên nhân cho việc truyền thông điểm tới điểm với các nền khác một cách xác thực.

## Chương 4: Định danh.

### (Naming)

#### 4.1 Các thực thể định danh (Naming Entities).

##### 4.1.1 Tên, định danh và địa chỉ.

Tên (name): là xâu các bit hoặc kí tự dùng để tham chiếu đến một thực thể trong hệ phân tán.

Địa chỉ (address): khi truy cập đến thực thể ta sử dụng điểm truy cập (access point). Các điểm truy cập này cũng phải được đặt tên và tên đó chính là địa chỉ của nó. Như vậy địa chỉ của thực thể chính là tên của điểm truy cập thực thể tương ứng.

Định danh (Identifiers): đây cũng là một kiêu tên đặc biệt. Việc định danh một tên phải thỏa mãn ba tính chất sau:

- Mỗi thực thể chỉ được tham chiếu bởi duy nhất một định danh ID
- Mỗi ID tham chiếu tới một thực thể.
- ID đó không được gắn cho một thực thể khác.

Không gian tên (Name Space): dùng để biểu diễn tất cả các tên. Nếu xét về mặt

hình học thì đây là một đồ thị có hướng, gồm các nút và các cung, gọi là đồ thị

tên (naming graph). Đồ thị có cấu trúc: Mỗi nút là miêu tả một thực thể.

Mỗi nút directory gắn với nhiều nút khác; lưu trữ trong bảng directory, bảng này là tập các cặp (label, identifier).

Tên thân thiện (Human-friendly name): là các tên được đặt một cách dễ hiểu, thân thuộc với con người.

#### 4.1.2 Độ phân giải tên.

Không gian tên đưa ra kỹ thuật lưu trữ và tìm kiếm các tên trên nó một cách dễ

đảng. Một trong những phương pháp hay dùng là sử dụng đường dẫn tên (path

name). Quá trình tìm kiếm tên trong không gian tên được gọi là phân giải tên

(name resolution). Quá trình phân giải tên trả về định danh một nút.

Closure mechanism: là kỹ thuật cho ta biết quá trình tìm kiếm tên được bắt đầu

như thế nào và bắt đầu ở đâu.

Linking: kỹ thuật này sử dụng bí danh (alias) - tên giống với tên của thực thể.

Với kỹ thuật này cho phép nhiều đường dẫn cùng tham chiếu đến cùng một nút

trên đồ thị tên. Một cách tiếp cận khác là dùng một nút lá không phải để lưu trữ

địa chỉ hay trạng thái của thực thể mà để lưu trữ đường dẫn tuyệt đối tới thực

thê đó.

Mounting: là kỹ thuật được thực hiện khi tìm kiếm trên hai không gian tên. Một

nút thư mục được gọi là một mount point (điểm gắn kết) lưu giữ 1d (hoặc các

thông tin cần thiết cho việc xác định và truy nhập) một nút thư mục bên phía không gian tên cần gắn kết được gọi là mounting point.

Hình 10. Mouting một không gian tên từ xa nhờ một giao thức truy cập

Thông thường, nếu 2 không gian tên NS1, NS2 - để gắn kết một thực thể bên

ngoài trong hệ phân tán, chúng ta cần tối thiêu những thông tin sau:

- tên của giao thức truy nhập (được xác định để thực hiện giao thức truyền thông)
- Tên của server (xác định địa chỉ server)
- tên của mounting point (xác định 1d của nút trong không gian tên bên ngoài)

#### 4.1.3 Thực hiện một không gian tên.

##### Phân phối không gian tên

Trong hệ phân tán, việc quản lý tên được thực hiện bằng cách phân thành các mức:

Mức Global: Chứa những nút thư mục ở mức cao (gốc và con của nó). Trong lớp này các nút thư mục ít thay đổi. Khả năng sẵn sàng ở lớp Global được yêu cầu cao hơn so với các lớp còn lại. Nếu name server của lớp này bị lỗi thì việc phân giải tên không thể thực hiện.

Mức Administrational: Chứa những nút thư mục ở mức trung gian, nó có thể được nhóm thành các nhóm, và mỗi nhóm có thể được chia cho những khu vực

quản trị khác nhau. Các nút ở trong nhóm này cũng ít khi thay đổi.

#### Khả năng

sản sàng của name server trong lớp administrational là rất quan trọng đối với

các client do name server quản lý. Vì nếu server này lỗi thì có rất nhiều các tài

nguyên không thể truy cập

Mức Managerial: Chứa những nút thư mục ở mức thấp. Các nút trong mức này

thay đổi khá thường xuyên. Ví dụ như các host trong một mạng LAN. Yêu cầu

đối tính sẵn sàng của name server của lớp managerial ít khắt khe hơn so với 2

lớp trên. Song về hiệu năng thì yêu cầu đối với lớp này cao hơn do phải thường

xuyên cập nhật các thay đổi.

Hình17 . Phân phối không gian tên

#### Thực hiện phân giải tên

Cách 1: phân giải tên tương tác (interactive name resolution), việc phân giải tên

thực hiện bằng cách truyền và nhận qua lại giữa client và các name server ở các

mức khác nhau. Theo cách này thì các server không trao đổi trực tiếp với nhau,

mỗi server chỉ phân giải nhãn tương ứng với lớp để xác định địa chỉ của server

tiếp theo, kết quả trả lại cho client là địa chỉ của name server tiếp theo, và việc

liên kết với server tiếp theo là do client đảm nhiệm.

Hình 18. Phân giải tên tương tác

Cách 2: phân giải tên đệ quy (recursive name resolution), theo cách này thì mỗi

name Server sẽ gửi kết quả đến name server tiếp theo mà nó tìm thấy. Và cứ

như vậy cho đến khi hoàn thành phân giải toàn bộ đường dẫn.

## 4.2 Định vị các thực thể di động.

### 4.2.1 Tên và việc định vị các thực thể.

Mỗi thực thể đều có tên và địa chỉ tương ứng, việc ánh xạ từ tên đến địa chỉ của

thực thể được thực hiện theo hai phương pháp: theo mô hình một lớp và theo mô hình hai lớp.

Theo mô hình một lớp: chỉ có một mức ánh xạ giữa tên và thực thể.  
Mỗi lần

thực thể thay đổi vị trí, ánh xạ cần phải được thay đổi theo

Theo mô hình hai lớp: phân biệt tên và địa chỉ nhờ Entity ID. Gồm hai quá

trình: quá trình tìm Entity ID tương ứng từ tên của thực thể được thực hiện

bằng dịch vụ tên (naming service) và quá trình xác định vị trí của thực thể từ

ID được thực hiện bởi dịch vụ định vị (Location service).

Hình 20 (a). Mô hình một lớp (b). Mô hình hai lớp.

### 4.2.2 Các giải pháp định vị thực thể.

Broadcasting và multicasting: gửi ID cần tìm tới tất cả các máy. Máy nào có

thực thể đó thì gửi lại một thông báo chứa địa chỉ của aCCES point.

Với phương

pháp này, yêu cầu tất cả các trình điều khiển lắng nghe yêu cầu gửi đến.

Dùng con trỏ (forwarding pointer): với một thực thể di động rời khỏi vị trí A

của nó đến vị trí B thì nó sẽ để lại một tham chiếu tới vị trí mới của nó. Nhờ đó,

khi định vị được thực thể, client có thể xác định ngay được địa chỉ hiện tại của

thực thể này nhờ vết địa chỉ đó.

Home-based approaches: cấp phát cho mỗi thực thể một vị trí gốc (home)

Với phương pháp này sẽ tạo ra một home location để lưu giữ địa chỉ hiện tại của các thực thể (thường là nơi thực thể được tạo ra ).

Địa chỉ của home được đăng ký tại naming service.

Home đăng ký địa chỉ ngoại của các thực thể

Client luôn đến home trước tiên, và sau đó tiếp tục với các vị trí bên ngoài.

Hình .

Hierarchical approaches: xây dựng một cây tìm kiếm phân cấp và thực hiện phân miền ở các mức khác nhau. Mỗi domain hình dung như một nút thư mục riêng biệt dir(d). Nút gốc biết tất cả các thực thể. Mỗi thực thể trong một domain D tương ứng với một location record trong nút thư mục dir(D), nó là địa chỉ hiện tại của thực thể hoặc một con trỏ.  
Hình 22 .Hierarchical approaches

Địa chỉ của một thực thể được lưu trong một nút lá, hoặc một nút trung gian.

Nút trung gian chứa một con trỏ đến một nút con nếu và chỉ nếu cây con năm tại nút con lưu trữ một địa chỉ của thực thể. Một thực thể có thể có nhiều địa chỉ (ví dụ trong trường hợp tạo bản sao).

Hình 23. Cấu trúc nút

Nguyên lý cơ bản: Bắt đầu tìm kiếm ở các nút lá cục bộ. Nếu nút đó biết thực

thê, tiếp theo sẽ đi xuống phía dưới theo con trỏ, ngược lại đi lên trên. Tìm kiếm lên mức cao nhất là root

#### Hình 24 .Nguyên lý tìm kiếm

4.3 Xóa bỏ các thực thể không còn được tham chiếu (Unreferenced Entities).

4.3.1 Đếm các tham chiếu (Reference Counting).

Mỗi lần client tạo(xóa) một tham chiếu đến một đối tượng O, một bộ đếm tham chiếu sẽ tăng thêm (giảm đi).

4.3.2 Lê danh sách các tham chiếu (Reference Listing).

Skeleton duy trì một danh sách tất cả các proxy trả về nó.

Ở đây đưa ra khái niệm Idempotent operation là một thao tác nó có thể lặp đi lặp lại nhiều lần mà không ảnh hưởng đến kết quả ( ví dụ  $1\#]=1$ ).

Thông điệp để thêm/xóa một proxy của danh sách cũng gần giống như  
tăng/giảm bộ đếm tham chiếu.

Các thực thể chuyên tham chiếu cho các thực thể khác nhưng không  
thể lấy  
được từ root.

Tập hợp loại bỏ dựa trên cơ sở truy nguyên: kiểm tra những phương  
thức có thể  
lấy được từ root và remove

### Chương 5 : Đồng bộ hóa (Synchronization)

5.1 Đồng bộ hóa đồng hồ (Clock Synchronization).

Trong hệ phân tán,mỗi máy tính là một đồng hồ nên việc đồng bộ  
các đồng hồ

này là rất cần thiết và rất khó khăn.

### 5.1.1 Đồng hồ vật lý (Physical Clock).

Chúng ta có nhiều cách để xác định thời gian. Phổ biến nhất là các hệ  
đếm thời  
gian theo thiên văn và ở đây là mặt trời. Có 23h một ngày và 3600  
giây. Một  
giây mặt trời được tính là 1/5600 của một ngày mặt trời. Một trong  
những mô  
hình để tính thời gian áp dụng phương pháp trên là Internatinal  
Atomic Time  
viết tắt là TAI. Tuy nhiên, TAI lại có một vấn đề là cứ 86400TAIs sẽ  
có 3ms  
chậm hơn so với đồng hồ mặt trời.

Để thông nhất thời gian vật l người ta đã đưa ra khái niệm thời gian  
phối hợp  
toàn cầu UCT (Universal Coordinate Time). Viện chuẩn quốc gia  
Mỹ đã lập ra  
trạm phát radio sóng ngắn WWV để gửi UTC khi cần hoặc định kì.

### 3.1.2 Các giải thuật đồng bộ hóa vật lý (Clock synchronization algorithm).

Nếu tất cả các máy tính đều có WWV Receiver thì việc đồng bộ  
chúng là dễ  
đảng vì tất cả đều cung đồng bộ với giờ chuẩn quốc tế UTC. Tuy  
nhiên khi  
không có WWV thì việc đồng bộ được thực hiện bằng các giải thuật  
đồng bộ  
sau.

#### a. Cải thuật Cristian

Cả sử trong hệ phân tán có một máy có WWYV (gọi là Time server )  
và chúng  
ta sẽ tiến hành đồng bộ các máy khác với máy này. Trong khoảng  
thời gian  $\delta/2p$   
mỗi máy sẽ gửi một thông điệp đến máy chủ hỏi thời gian hiện tại.  
Máy chủ

nhanh sẽ phản hồi bằng một thông điệp mang giá trị thời gian C(utc). Bên gửi nhận được phản hồi nó sẽ thiết lập lại clock thành C(uet).

Hình 25. Xác định thời gian trong time server

Đánh giá: giải thuật này có 2 vấn đề :

- Một là nếu clock bên gửi chạy nhanh thì lúc này C(uct) sẽ nhỏ hơn thời gian hiện tại C của bên gửi.. Có thể giải quyết bằng cách thay đổi nhịp ngắt lại nhanh hơn hoặc chậm hơn cho đến lúc khớp nhau.
  - Hai là sự chênh lệch từ lúc C(uct) được gửi cho đến lúc nhận được có thể gây lỗi. Giải quyết bằng cách ghi nhận khoảng thời gian giữa lúc gửi và nhận
- b. Giải thuật Berkeley.

Tư tưởng của giải thuật:

Server sẽ chủ động cho các máy khác biết thời gian chuẩn của mình CUTC sau đó sẽ yêu cầu thông tin về thời gian của các client.

Client sẽ trả lời khoảng thời gian chênh lệch giữa nó và server.

Server sẽ tính khoảng thời gian mà các client so với thời gian chuẩn của server lúc đó và gửi cho các máy khách cách điều chỉnh thời gian cho phù hợp.

Hình 26 . Đồng bộ theo giải thuật Berkeley

c. Cải thuật trung bình

Giải thuật này thực hiện chia thời gian thành những khoảng đồng bộ cố định.

Khoảng thời gian I sẽ bắt đầu từ thời điểm ( $To + \frac{1}{2}R$ ) và chạy đến khi  $To +$

(r1) R với To là thời điểm xác định trước và R là một biến hệ thống.

Vào thời điểm bắt đầu của mỗi lần đồng bộ tất cả các máy của mạng sẽ broadcast thời gian của mình .

Sau khi broadcast nó sẽ bắt đầu thu thập thời gian mà các máy khác gửi đến trong khoảng thời gian S. Sau đó bỏ đi giá trị lớn nhất và nhỏ nhất rồi tính trung bình của các giá trị thời gian còn lại.

## T2 Đồng hồ logic (Logical Clock)

Trong nhiều trường hợp, giữa các tiến trình không nhất thiết phải phù hợp theo thời gian thực tế mà chỉ cần khớp với nhau về thời gian. Do đó người ta đưa ra khái niệm đồng hồ

logic.

### 3.2.1 Nhãn thời gian Lamport (Lamport timestamps).

Lamport đã đưa ra mô hình đồng hồ logic đầu tiên cùng với khái niệm nhãn thời gian.

\a. Xét định nghĩa mỗi quan hệ "xảy ra trước" (

B:A xảy ra trước B thì tất cả các tiến trình trong hệ phân tán thỏa thuận sự kiện A xảy ra trước rồi đến sự kiện B.\ Khi có A

B là đúng.\ A và B là hai sự kiện của cùng một tiến trình. Nếu A xảy ra trước B thì A

Nếu A là sự kiện bản tin được gửi bởi một tiến trình nào đó, còn B (là sự kiện bản tin đó được nhận bởi một tiến trình khác thì quan hệ A  $\rightarrow$  B là đúng.

C.\ C thì A( B., B(Quan hệ xảy ra trước có tính bắc cầu: A

b. Tem thời gian (Time Stamps)

Đề đo thời gian tương ứng với 4 sự kiện x thì ta gán một giá trị  $C\{x\}$  cho sự kiện đó và thỏa mãn các điều kiện sau:

B trong cùng một tiền trình thì  $C(A)|Nếu A < \epsilon(B)$ .

Nếu A và B biểu diễn tương ứng việc gửi và nhận một thông điệp thì ta có

$$C(A) < C(B)$$

Với mọi sự kiện phần biệt (không có liên quan) thì  $C(A) \leq C(B)$

### 5.2.2 Vector thời gian (Vector Timestamps)

Cải thuật vector timestamp đưa ra một vector timestamp VT(a) gắn cho sự kiện

a có thuộc tính là nếu  $V_{tt}(a) < V_{tt}(b)$  thì sự kiện là nguyên nhân của b.

Trong vector thời gian mỗi tiền trình Pi lưu giữ một Vi với giá trị N (các tiền

trình khác nhau thì N khác nhau)

- VIH] là số các sự kiện đã xảy ra tại PI

- Nếu  $Vi[j] = k$  nghĩa là Pi biết đã có k sự kiện đã xảy ra tại Pj

Yêu cầu: mỗi khi có sự kiện mới xảy ra ở tiền trình Pi thì phải tăng  $VI[I]$  và

phải đảm bảo vector này được gửi cùng thông điệp suốt trong quá trình.

Nhờ đó bên nhận sẽ biết được đã có bao nhiêu sự kiện xảy ra tại PI .Quan trọng

hơn phía nhận sẽ báo cho biết là đã có bao nhiêu sự kiện ở các tiền trình khác

đã xảy ra trước khi PI gửi thông điệp m. Nói cách khác timestamp VT của n nói cho bên nhận biết bao nhiêu sự kiện đã xảy ra trong các tiền trình khác trước m.

#### Luật cập nhật vector

- Thiết lập  $V_i[j] = 0$  với mọi  $j.i$
- Sự kiện xảy ra ở PI là nguyên nhân tăng  $V_i[1]$
- PI gắn một timestamp  $t=V_i[i]$  vào mọi thông điệp gửi đi
- Khi PI nhận được một thông điệp có f nó sẽ thiết lập  $VII|EMax(VH[J], t[[]])$  và tăng  $VIII$

#### 5.3 Trạng thái tổng thể (Global state).

Việc xác định trạng thái tổng thể của hệ thống rất có ích. Một trong những phương pháp được đưa ra là Chụp Nhanh Phân Tán (Distributed Snapshot) cùng khái niệm lát cắt (cut).

#### Hình 28 .(a) Lát cắt nhất quán. (b) Lát cắt không nhất quán

Một lát cắt nhất quán được biểu diễn là đường chấm gạch trong hình a. Lát cắt mô tả sự kiện cuối cùng mà sự kiện này được ghi lại cho mỗi tiền trình. Bằng cách này nó có thể kiểm tra lại rằng tất cả các thông điệp nhận đều tương ứng với các thông điệp gửi được ghi lại trên đường cắt. Ngược lại là lát cắt không nhất quán như hình vẽ b: Thời điểm tiền trình P-3 nhận thông điệp m2 được ghi vào lát cắt nhưng việc ghi lại này không tương ứng với sự kiện gửi!.

#### 5.4 Các giải thuật bầu chọn (Election Algorithm).

khi tiến trình điều phối gặp lỗi thì sẽ phải có quá trình bầu chọn để chọn ra một tiến trình khác làm điều phối thay cho nó. Có hai giải thuật bầu chọn hay được sử dụng là:

#### 5.4.1 Giải thuật áp đảo (Bully Algorithm)

Với giả thiết:

Mỗi một tiến trình đều có một ID duy nhất. Tất cả các tiến trình khác đều có thể biết được số ID và địa chỉ của mỗi tiến trình trong hệ thống.

Chọn một tiến trình có ID cao nhất làm khóa. Tiến trình sẽ khởi động việc bầu chọn nếu như nó khôi phục lại sau quá trình xảy ra lỗi hoặc tiến trình điều phối bị trục trặc.

Các bước của giải thuật:

- 1.P gửi thông điệp ELEC đến tất cả các tiến trình có ID cao hơn
- 2.Nếu không có tiến trình nào phản hồi thì P sẽ trở thành tiến trình điều phối
- 3.Nếu có một tiến trình có ID cao hơn phản hồi thì nó sẽ đảm nhiệm vai trò điều phối.

Hình 29 .Ví dụ theo giải thuật áp đảo

#### 5.4.2 Giải thuật vòng (Ring Algorithm)

VỚI giả thiết :

Các tiến trình có một ID duy nhất và được sắp xếp trên 1 vòng tròn Logie. Mỗi một tiến trình có thể nhận biết được tiến trình bên cạnh mình.

Các bước thuật toán:

Một tiến trình bắt đầu gửi thông điệp ELEC tới các nút còn tồn tại gần nhất,  
quá trình gửi theo ì hướng nhất định. Thăm dò liên tiếp trên vòng  
cho đến khi  
tim được ì nút còn tồn tại.

Mỗi một tiến trình sẽ gắn ID của mình vào thông điệp gửi.

Cuối cùng sẽ chọn ra Ì tiến trình có [D cao nhất trong số các tiến  
trình còn hoạt  
động và gửi thông điệp điều phối cho tiến trình đó.

5.5 Loại trừ nhau (Mutual Exclusion).

Tổ chức các "vùng tối hạn" (critical section region).

Có nhiều giải thuật được xây dựng để cài đặt cơ chế loại trừ nhau  
qua  
những  
các vùng tối hạn. Có ba giải thuật phổ biến là:

3.5.1 Giải thuật tập trung (Centralized Algorithm)\_

Giả thiết: mỗi tiến trình có một số ID duy nhất. Tiến trình được bầu  
chọn làm  
điều phối là tiến trình có số hiệu ID cao nhất.

Nội dung thuật toán: Khi một tiến trình nào đó cần vào vùng giới  
hạn nó sẽ gửi  
một thông điệp xin cấp quyền Nếu không có một tiến trình nào đang  
trong  
vùng giới hạn thi tiến trình điều phối sẽ gửi phản hồi cho phép. Còn  
nếu có một  
tiến trình khác đang ở trong vùng giới hạn rồi thi tiến trình điều phối sẽ  
gửi thông  
điệp từ chối và đưa tiến trình này vào hàng đợi cho đến khi không có  
tiến trình  
nào trong vùng giới hạn nữa.

Khi tiến trình một tiến trình rời khỏi vùng giới hạn nó sẽ gửi một  
thông điệp

đến tiền trình điều phối thông báo trả lại quyền truy cập.Lúc này tiền trình điều

phối sẽ gửi quyền truy cập cho tiền trình đầu tiên trong hàng đợi truy cập.

Đánh giá : Thuật toán này có đảm bảo sự tồn tại duy nhất một tiền trình trong

vùng tới hạn và chỉ cần 3 thông điệp để thiết lập là: Request -Grant -

Release .Nhược điểm duy nhất là nếu tiền trình điều phối bị hỏng thì hệ thống

SẼ SỰP ĐỒ .Vì nếu một tiền trình đang trong trạng thái Block nó sẽ không thể

biết được tiền trình điều phối có bị DEAD hay không .Trong một hệ thống lớn

nếu chỉ có một tiền trình điều phối sẽ xuất hiện hiện tượng thắt cõ chai

Hình 30 .ví dụ theo giải thuật tập trung

### 3.3.2 Giải thuật phân tán (DIistributed Algorithm)

Khi một tiền trình muốn vào vùng giới hạn, trước hết nó sẽ tạo ra một nhãn

thời gian và gửi cùng với một thông điệp đến tất cả các tiền trình khác. Các tiền

trình khác sau khi nhận được thông điệp này sẽ xảy ra ba tình huồng:

Nếu bên nhận không ở trong vùng giới hạn và cũng không muốn vào vùng giới

hạn thì nó sẽ gửi thông điệp OK cho bên gửi

Nếu bên nhận đang ở trong vùng giới hạn thay vì trả lời nó sẽ cho vào hàng đợi yêu cầu này.

Nếu bên nhận cũng muốn vào hàng đợi thì nó sẽ so sánh timestamp ai thấp hơn sẽ thắng.

Sau khi gửi đi thông điệp yêu cầu vào vùng giới hạn tiền trình sẽ đợi cho đến

khi có trả lời cảng sớm cảng tốt .Khi đã vảo vùng giới hạn rồi thì nó sẽ gửi

thông điệp OK đến tất cả các tiến trình khác và xóa các tiến trình trong hàng đợi đi.

Hinh 3L. Ví dụ theo giải thuật phần tán

3.5.3 Giải thuật vòng với thẻ bài (TokenRing Algorithm).

Giả thiết tất cả các tiến trình được sắp xếp trên một vòng tròn logic, các tiến trình đều được đánh số và đều biết đến các tiến trình cạnh nó.

Bắt đầu quá trình truyền, tiến trình 0 sẽ được trao một thẻ bài. Thẻ bài này có thể lưu hành xung quanh vòng tròn logic. Nó được chuyên từ tiến trình k đến tiến trình ( $k+1$ ) bằng cách truyền thông điệp điểm - điểm. Khi một tiến trình giành được thẻ bài từ tiến trình bên cạnh nó sẽ kiểm tra xem có thẻ vào vùng tới hạn hay không. Nếu không có tiến trình khác trong vùng tới hạn nó sẽ vảo vùng tới hạn. Sau khi hoàn thành phần việc của mình nó sẽ nhả thẻ bài ra, thẻ bài có thể di chuyển tự do trong vòng tròn. Nếu 1 tiến trình muốn vào vùng tới hạn thì nó sẽ giữ lấy thẻ bài, nếu không nó sẽ để cho thẻ bài truyền qua. Vẫn đê lớn nhất trong thuật toán truyền thẻ bài là thẻ bài có thẻ bị mất, khi đó chúng ta phải sinh lại thẻ bài bởi vì việc đỗ tìm lại thẻ bài là rất khó.

Hinh 32. Ví dụ theo giải thuật vòng với thẻ bài

3.0 Các giao tác phân tán (DIistributed Transactions).

Bốn tính chất của giao tác đối với thẻ giới bên ngoài: ACID

Tính nguyên tử (Atomic): mọi giao tác diễn ra không thể phân chia được.

Tính nhất quán (Consistency): giao tác không xâm phạm các bất biến của hệ thống.

Tính cô lập (Isolated): các giao tác đồng thời không gây trở ngại cho nhau.

Tính lâu bền (Durable): khi giao tác đã cam kết thì các thay đổi đối với nó không phải là tạm thời mà là kéo dài.

### 5.6. Phân loại các giao tác

Cao tác được chia thành các loại sau:

Limitation of Flat Transaction.

Nested Transaction

Distributed Transaction.

#### 5.6.2 Điều kiện tương tranh:

Là quá trình cho phép nhiều giao tác thực hiện đồng thời mà không xảy ra sự tranh chấp giữa các giao tác. Có hai loại tương tranh:

Tương tranh bị quan.

Tương tranh lạc quan.

Chương 6 : Nhất quán và nhân bản

(Consistency & replication)

#### 6.1 Đặt vấn đề.

Có hai lý do để sử dụng bản sao:

Dùng bản sao để tăng độ tin cậy và tính sẵn sàng của hệ thống: khi dữ liệu bị

lỗi hay vì một nguyên nhân nào đó mà không thể dùng được, ta có thể dùng ngay bản sao dữ liệu đó để hệ thống không phải dừng lại và tránh được tình trạng sử dụng các dữ liệu không chính xác.

Dùng bản sao để tăng hiệu năng của hệ thống: có thể tăng quy mô hệ thống cả về Số lượng lẫn phạm vi địa lý.

Tuy nhiên việc sử dụng nhân bản cũng phải trả giá, đó là tính nhất quán dữ liệu của hệ thống bị suy giảm. Do sử dụng bản sao nên có thể xảy ra trường hợp có sự thay đổi trên một dữ liệu mà không cập nhật trên các bản sao của nó. Điều này sẽ gây ra các sai sót trong hệ thống. Do đó phải tốn nhiều công sức để xây dựng các mô hình đảm bảo tính nhất quán của dữ liệu.

## 6.2 Các mô hình nhất quán lấy dữ liệu làm trung tâm.

### 6.2.1 Mô hình nhất quán chặt (StrIcf consistency).

Là mô hình thỏa mãn điều kiện sau: Thao tác đọc bất kỳ trên mục dữ liệu x đều trả về một giá trị tương ứng với kết quả của thao tác ghi gần nhất trên x đó.

Sử dụng khái niệm thời gian tuyệt đối. Thời gian tuyệt đối này là tông thê cho cả hệ thống để xác định đúng khái niệm "gần nhất". Điều này là khó khả thi với hệ phân tán.

Các kí hiệu:

- WI(x)a: thao tác ghi được thực hiện bởi tiến trình  $P(j)$  lên mục dữ liệu x với giá trị a.

-  $R_i(x)b$ : thao tác đọc được thực hiện bởi tiến trình  $P()$  lén mục dữ liệu  $x$  cho kết quả  $b$ .

Giả thiết  $x$  có giá trị ban đầu là null.

Hình 33 (a). Mô hình nhất quán chặt. (b) Không phải là mô hình nhất quán chặt

Do việc lan truyền cục bộ của  $P_1$  chưa tới  $P_2$  nên  $P_2$  đọc dữ liệu  $x$  vẫn là giả  $fri$   
null ban đầu.

Mô hình này là không khả thi nên đưa ra mô hình giảm nhẹ hơn.

#### 6.2.2 Mô hình nhất quán tuần tự và mô hình nhất quán tuyến tính.

##### a. Mô hình nhất quán tuần tự.

Là mô hình lỏng lẻo hơn, yêu hơn mô hình nhất quán chặt. Nó thỏa mãn các yêu cầu sau:

Kết quả của sự thực hiện bất kỳ là như nhau nếu thao tác đọc và ghi do các tiến trình thực hiện trên mục dữ liệu một cách tuần tự và các thao tác của mỗi tiến trình xuất hiện trong chuỗi thao tác này chỉ ra bởi chương trình của nó.

Khi các tiến trình chạy đồng thời trên các máy khác nhau thì cho phép sự đan xen của các thao tác nhưng tất cả các tiến trình đều phải nhận biết được sự đan xen của các thao tác đó là như nhau.

Hình 34 (a). Mô hình nhất quán tuần tự. (b) Không là mô hình nhất quán tuần tự.

##### b. Mô hình nhất quán tuyến tính.

Là mô hình yếu hơn mô hình nhất quán chặt nhưng mạnh hơn mô hình nhất quán tuần tự. Mô hình này thỏa mãn điều kiện sau: "Kết quả của bất kì sự thực hiện nào là như nhau nếu các thao tác (đọc và ghi) của tất cả các tiến trình lén dữ liệu được thực hiện một cách tuần tự và các thao tác của mỗi tiến trình xuất hiện trong chuỗi thao tác này phải theo thứ tự đã được chỉ ra trong chương trình của nó. Thêm vào đó, nếu  $tsopl(x) < tsop2(y)$  thì thao tác  $opl(x)$  phải được thực hiện trước  $op2(y)$  trong chuỗi thao tác"

#### 6.2.3 Mô hình nhất quán nhân quả.

Đây là mô hình lỏng lẻo hơn mô hình nhất quán tuần tự. Mô hình này phân biệt các sự kiện có quan hệ nhân quả và các sự kiện không có quan hệ nhân quả. Nếu sự kiện b được gây ra hoặc bị tác động bởi một sự kiện a xảy ra sớm hơn thì tính nhân quả đòi hỏi mọi thực thể khác phải "nhìn" thấy a trước rồi mới thấy b sau.

Mô hình nhất quán nhân quả thỏa mãn các điều kiện sau: các thao tác ghi có quan hệ nhân quả tiềm năng phải được nhận biết bởi tất cả các tiến trình khác trong cùng một thứ tự. Các thao tác ghi đồng thời có thể nhận biết được theo thứ tự khác nhau trên các máy khác nhau.

#### Hình 35 Mô hình nhất quán nhân quả.

#### 6.2.4 Mô hình nhất quán FIFO .

Nhất quán FIFO còn được gọi là nhất quán PRAM. Đây là mô hình yêu nhất vì

mô hình này bỏ qua giới hạn về trật tự của bất kì thao tác đồng thời nào. Nhất quán FIFO thỏa mãn : "Các thao tác ghi bởi một tiến trình đơn phải được tất cả các tiến trình khác nhìn thấy theo cùng một trật tự mà chúng đề ra. Nhưng thao tác ghi bởi nhiều tiến trình khác nhau có thể được thấy theo những trật tự khác nhau bởi các tiến trình khác nhau".

Hình 36 Mô hình nhất quán FIFO.

#### 6.2.5 Mô hình nhất quán yếu (Weak consistency).

Mô hình nhất quán yếu không tập trung vào các thao tác trên dữ liệu như các mô hình trên mà chúng quan tâm đến trật tự các nhóm lệnh bằng việc sử dụng các biến được đồng bộ.

Mô hình nhất quán yếu có ba đặc tính sau:

s Việc truy cập đến một biến đồng bộ hóa được kết hợp với kho dữ liệu là một nhất quán tuần tự.

◦ Không có thao tác nào lên các biến đồng bộ hóa được phép thực hiện cho đến khi tất cả các thao tác ghi trước đó được hoàn thành ở mọi nơi.

◦ Không có thao tác đọc hay ghi dữ liệu lên các mục dữ liệu nào được phép thực hiện cho đến khi tất cả các thao tác trước đó lên các biến đồng bộ hóa được thực hiện.

Hình 37 (a) Mô hình nhất quán yếu. (b) Không là mô hình nhất quán yếu

6.2.6 Mô hình nhất quán đi ra (Release COHERENCY ).

Sử dụng thêm hai lệnh: lệnh acquired để báo muốn vào vùng tới hạn (critical region) và lệnh release để báo giải phóng vùng tới hạn. Hai lệnh này cũng có hai cách thực thi khác nhau như: bằng một biến hoặc bằng một lệnh đặc biệt.

Hai thao tác này chỉ thực hiện với các dữ liệu dùng chung chứ không áp dụng cho tất cả các dữ liệu.

Mô hình nhất quán đi ra thỏa mãn các điều kiện sau:

- Trước khi thực hiện một thao tác đọc hay ghi lên dữ liệu chia sẻ thì tất cả các thao tác acquire do tiên trình này thực hiện trước đó phải hoàn tất.
- Trước khi một thao tác release được phép thực hiện thì tất cả các thao tác đọc và ghi do tiên trình này thực hiện trước đó phải được hoàn tất.
- Truy cập vào các biến đồng bộ hóa là nhất quán FIFO (Không yêu cầu nhất quán tuần tự).

Hình 38 Trình tự sự kiện theo mô hình nhất quán đi ra

#### 6.2.7 Mô hình nhất quán đi vào (Entry consistency).

Cũng giống mô hình nhất quán đi ra, mô hình nhất quán đi vào cũng sử dụng hai lệnh acquired và release khi muốn sử dụng vào vùng tới hạn. Nhưng các lệnh này thao tác trên từng mục dữ liệu của vùng dữ liệu chia sẻ. Tiên trình nào muốn sử dụng mục dữ liệu thì phải đợi cho tất cả các tiên trình khác giải phóng mục dữ liệu đó.

Hình 39 .Trình tự sự kiện theo mô hình nhất quán đi vào.

Để ghi lên một mục dữ liệu, client phải có được biến đồng bộ hóa của mục đó trong chế độ dành riêng. Điều đó có nghĩa là không client nào khác có thể sử dụng biến đó. Khi client cập nhật xong mục dữ liệu, thì nó giải phóng biến đó. Khi client muốn đọc một mục dữ liệu nào đó, nó phải có được biến đồng bộ hóa kết hợp ở chế độ không dành riêng. Nhiều client có thể giữ một biến đồng bộ hóa ở chế độ không dành riêng.

Khi thực hiện một thao tác acquire, client lấy về phiên bản mới nhất của mục dữ liệu từ tiền trình cuối cùng thực hiện thao tác acquire trên biến đó.

Nhất quán đi vào phải thỏa mãn các điều kiện sau:

- Một thao tác acquire để truy cập vào một biến đồng bộ hóa không được phép thực hiện trong một tiền trình cho đến khi tất cả các cập nhật lên mục dữ liệu trong tiền trình đó được thực hiện.
- Trước khi một truy cập trong chế độ dành riêng của một tiền trình tới một biến đồng bộ hóa được phép thực hiện thì không tiền trình nào khác còn được giữ các biến đồng bộ hóa, trong chế độ không dành riêng thi không cần yêu cầu như vậy.
- Sau khi một truy cập trong chế độ dành riêng lên một biến đồng bộ hóa được thực hiện thì bất kì sự truy cập của tiền trình nào khác trong chế độ không, dành riêng lên biến đó cũng không được thực hiện cho đến khi chủ nhân của biến

đồng bộ thực hiện xong việc truy cập của mình.

### 6.3 Các mô hình nhất quán lấy client làm trung tâm.

#### 6.3.1 Nhất quán cuối cùng (eventual consistency).

Khi một dữ liệu có nhiều bản sao thì yêu cầu đưa ra là sau các thao tác cập nhật  
thì tất cả các bản sao cuối cùng là phải bằng nhau. Yêu cầu này sẽ  
được thực hiện tốt nếu mỗi client luôn chịu khó cập nhật cho các bản sao.

Nếu các client là di động thì việc thực hiện yêu cầu trên gặp khó khăn hơn.

Phải luôn đảm bảo rằng cả khi client thay đổi về vị trí vật lý thi  
việc sử dụng các bản sao cũng phải chính xác. Tức là các bản sao luôn luôn  
là nhất quán.

#### 6.3.2 Nhất quán đọc đều (monofonic - write consistency).

Mô hình nhất quán đọc đều phải đảm bảo điều kiện sau:

Một tiến trình thực hiện thao tác đọc trên một mục dữ liệu thì phải  
đảm bảo bất kì thao tác đọc nào cũng đều cho cùng một kết quả hay kết quả gần  
đầy nhất.

Mô hình nhất quán đọc đều đảm bảo rằng một client sẽ luôn nhìn  
thấy những dữ liệu mới hơn và không bao giờ phải nhìn thấy những dữ liệu cũ  
hơn những gì mà mình đã đọc trước đó. Điều đó có nghĩa là khi một client thực  
hiện một thao tác đọc trên một bản sao rồi tiếp theo lại đọc trên một bản sao  
khác thì bản sao thứ hai kia ít nhất cũng phải được ghi giống với bản sao đầu tiên.

Hình 41 (a) Kho dữ liệu theo mô hình nhất quán đọc đều (b) Kho dữ  
liệu không

theo mô hình nhất quán đọc đều

Về bản chất thì mô hình này là phiên bản hướng người dùng của mô hình nhất quán FIFO (điểm khác biệt ở chỗ nó chỉ áp dụng đối với một client).

### 6.3.3 Nhất quán ghi đều (monotonic - read consistency).

Mô hình nhất quán đọc đều phải đảm bảo điều kiện sau:

Thao tác ghi trên mục dữ liệu x của một tiến trình phải được hoàn thành trước bất kỳ một thao tác ghi nào khác trên x bởi cùng một tiến trình.

Nói cách khác thì các thao tác ghi lên một mục dữ liệu sẽ được sắp xếp một cách có trật tự.

Hình 42 (a) Kho dữ liệu theo mô hình nhất quán ghi đều (b) Kho dữ liệu không theo mô hình nhất quán ghi đều.

6.3.4 Nhất quán đọc kết quả ghi (Read - your - write consistency)  
Trong mô hình nhất quán này, người dùng được đảm bảo rằng sẽ luôn được

nhìn thấy những kết quả ghi mới nhất.

“Tác động của một thao tác ghi của một tiến trình lên mục dữ liệu x sẽ luôn được nhìn thấy bởi một thao tác đọc lần lượt trên x của cùng tiến trình đó”.

Hình 43 (a) Kho dữ liệu theo mô hình nhất quán đọc kết quả ghi (b) Kho dữ liệu không theo mô hình đọc kết quả ghi

### 6.3.5 Nhất quán ghi theo sau đọc (write - follow - read consistency).

Mô hình nhất quán này ngược với nhất quán đọc kết quả ghi, nó đảm bảo rằng một người dùng sẽ luôn thực hiện thao tác ghi lên một phiên bản dữ liệu mà ít

nhất cũng phải mới bằng phiên bản cuối cùng của nó.

Tác động bởi một thao tác ghi của một tiến trình lên mục dữ liệu x sẽ luôn được nhìn thấy bởi một thao tác đọc liên tiếp lên x của cùng tiến trình đó".

Hình 44 (a) Kho dữ liệu theo mô hình nhất quán ghi theo sau đọc  
(b) Kho dữ liệu không theo mô hình ghi theo sau đọc.

#### 6.4 Các giao thức phân phối (distribution protocols).

##### 6.4.1 Xếp đặt các bản sao (replica placement).

Có 3 kiểu bản sao:

Các bản sao thường trực: trong tiến trình hay trên máy luôn có một bản sao. Số lượng các bản sao thường xuyên này rất ít, thường được tập hợp lại thành nhóm các máy trạm (COWS) hoặc trong các hệ thống phản chiếu (mirrored), thường là các Web server hay là các server có chứa cơ sở dữ liệu dự phòng.

Bản sao khởi đầu từ server: Các bản sao này được sử dụng để làm tăng hiệu năng. Các bản sao này được xếp đặt động dựa vào yêu cầu của server khác. Một ví dụ điển hình là chúng được các công ty web hosting sử dụng để định vị vị trí địa lý của các bản sao gần nhất khi họ cần.

Các bản sao khởi đầu từ client: Các bản sao này được tạo ra từ yêu cầu của client, chẳng hạn như việc cache dữ liệu của một trình duyệt. Chúng được xếp đặt động dựa vào yêu cầu của client.

Hình 45 Tô chức logic của các loại bản sao.

#### 6.4.2 Lan truyền các cập nhật

Có 3 khả năng lan truyền các cập nhật

Chỉ thông báo là có cập nhật: Thường dùng trong việc cache dữ liệu. Thông báo về việc mất hiệu lực của một giao thức. Phương pháp này tốt khi tỉ lệ các thao tác đọc so với thao tác ghi nhỏ.

Truyền dữ liệu cập nhật từ bản sao này tới một bản sao khác. Thực hiện tốt khi có nhiều thao tác đọc. Ghi lại các thay đổi và tập hợp các cập nhật lại để truyền đi (chỉ truyền đi các thay đổi chứ không truyền cả dữ liệu đã bị thay đổi,反之). Tiết kiệm được băng thông).

Lan truyền các thao tác cập nhật tới các bản sao khác (nhân bản chủ động). Tốn ít băng thông nhưng đòi hỏi năng lực xử lý cao vì trong nhiều trường hợp thì các thao tác là rất phức tạp.

Các giao thức kéo và đẩy

Đẩy cập nhật: là giao thức do server khởi tạo, trong giao thức này các cập nhật được lan truyền mỗi khi có một server khác yêu cầu.

Kéo cập nhật: là giao thức do client khởi tạo khi client muốn được cập nhật.

#### 6.4.3 Các giao thức bệnh dịch (epidemic protocol).

Đây là một giao thức có thể dùng để thực hiện mô hình nhất quán sau cùng.

Ý tưởng cơ bản của thuật toán bệnh dịch là:

- Giả sử rằng không xảy ra xung đột giữa các thao tác ghi - ghi.

- Các thao tác cập nhật ban đầu được thực hiện chỉ trên một hay một vài bản sao (càng ít càng tốt).
- Một bản sao chỉ gửi các cập nhật của nó tới một số hữu hạn các hàng xóm.
- Việc lan truyền các cập nhật xảy ra chậm chạp và không phải ngay lập tức.
- Cuối cùng thì mỗi cập nhật cũng đến được từng bản sao.

Dựa trên thuật toán bệnh dịch mã có các mô hình lan truyền cập nhật. Điều đáng lưu tâm trong mô hình này là các cập nhật được lan truyền tới các bản sao với càng ít thông điệp càng tốt và càng nhiều bản sao bị "nhiễm" các lan truyền càng nhanh thì càng tốt. Đến cuối cùng nếu bản sao nào mà không lan truyền được cập nhật của mình thì nó sẽ bị loại bỏ. Tuy nhiên việc loại bỏ có thể sẽ không dễ dàng.

Một trong những mô hình lan truyền cập nhật được gọi là: anti entropy. Trong mô hình này, mỗi bản sao cứ định kì lại chọn ngẫu nhiên một bản sao khác và trao đổi những trạng thái khác nhau của mình, sau một thời gian thi cả 2 phía sẽ có những trạng thái giống hệt nhau.

Một mô hình khác là gossiping. Trong mô hình này một bản sao đã được cập nhật sẽ "kê" cho một số bản sao khác về những cập nhật của mình vì thế sẽ làm cho những bản sao đó bị nhiễm những cập nhật của mình.

## 6.5 Các giao thức nhất quán.

### 6.5.1 Giao thức Primary-based

### a. Các giao thức ghi từ xa

Với giao thức này, tất cả các thao tác ghi được thực hiện chỉ trên một SCTV€T từ xa. Giao thức này thường được kết hợp với các hệ thống chủ khách truyền thông.

Một dạng giao thức ghi từ xa là giao thức Primary-Backup.

Nhược điểm của giao thức này là vân đề hiệu năng. Tất cả các thao tác ghi trong giao thức đều chiếm khá nhiều thời gian, đặc biệt là khi có giao thêm giao thức ghi theo khối được sử dụng.

Ưu điểm của giao thức này là

- Sử dụng giao thức ghi không theo khối để xử lý các cập nhật.
- Tất cả các thao tác ghi có thể được gửi đến các bản sao dự phòng theo cùng, một thứ tự, điều này tạo điều kiện thuận lợi khi thực thi mô hình nhất quán tuần tự.

### b. Các giao thức ghi cục bộ

Trong giao thức này một bản sao của mục dữ liệu được duy trì. Khi có yêu cầu thao tác shl, mục dữ liệu được nhân bản từ server ở xa chuyên đến server cục bộ. Việc này được gọi là tiếp cận theo kiểu di trú hoàn toàn.

Một vấn đề được đặt ra cho các trình sử dụng giao thức này để đọc hoặc ghi lên các mục dữ liệu là: thời gian để thật sự định vị được một mục dữ liệu có thể còn lớn hơn cả thời gian tiền trình sử dụng nó.

Một dạng của giao thức ghi cục bộ là là giao thức Primary-backup.

Trong giao

thức này bản chính được di trú đến tiền trình đang muốn thực hiện việc cập

nhật, rồi sau đó bản dự phòng sẽ được cập nhật.

### 6.5. 2. Các giao thức Replicated-write.

Trong các giao thức này, thao tác ghi có thể được tiến hành tại bất kì

bản sao

nào.

Ví dụ:

Một tiền trình đặc biệt sẽ mang các thao tác cập nhật tới từng bản sao. Một tem

thời gian Lamport có thể được sử dụng để lấy các thao tác vế, tuy nhiên

phương pháp này không được linh hoạt cho các hệ phân tán.

Một phương pháp khác là sử dụng một bộ sắp xếp dãy, là một tiền

trình để gán

các số ID duy nhất cho mỗi cập nhật, sau đó truyền các cập nhật tới tất cả các

bản sao.

### Cao thức nhân bản chủ động

Trong giao thức này, các thao tác ghi được truyền đến tất cả các bản sao, trong

khi các thao tác đọc được thực hiện cục bộ. Giao thức ghi có thể

được truyền

sử dụng giao tiếp point-to-point hay multicast. Ưu điểm của giao

thức này là tất

tất cả các bản sao đều nhận được các thao tác cùng lúc và theo cùng một

trật tự, và

nó cũng không cần đánh dấu một bản chính hay phải gửi tất cả các

thao tác tới

một Server.

Tuy nhiên giao thức này lại đòi hỏi phải truyền theo kiểu multicast động hoặc phải có một bộ sắp xếp dãy tập trung mã cả 2 phương pháp này đều khó có thể tiếp cận một cách linh hoạt.

Trong giao thức này có một vấn đề cần quan tâm là "triệu gọi bản sao". Để tránh cho một bản sao bị gọi quá nhiều lần, một bộ điều phối được gắn ở mỗi bên (client và server), điều này đảm bảo cho việc chỉ có một lời gọi và một lời đáp được gửi đi.

### Cao thức Quorumbased

Với giao thức này, các thao tác ghi được thực hiện trên một tập nhỏ nhất các bản sao. Khi thực hiện một thao tác đọc, người dùng cũng phải liên hệ với một tập các bản sao để tìm ra phiên bản mới nhất của dữ liệu.

Trong giao thức này tất cả các mục dữ liệu được kết hợp với một SỐ phiên bản (version number). Mỗi lần một mục bị sửa đổi thì số phiên bản của nó cũng được tăng lên.

Giao thức này định nghĩa ra số đại biểu đọc và số đại biểu ghi, hai đại biểu này sẽ xác định số bản sao phải được liên hệ trước khi thực hiện thao tác đọc và ghi. Số đại biểu đọc phải lớn hơn  $1/2$  tổng số bản sao, vì thế tổng của số đại biểu đọc và ghi phải lớn hơn tổng số bản sao. Bằng cách này, một người muốn thực hiện một thao tác đọc thì phải đảm bảo việc liên hệ với ít nhất một bản sao có chứa phiên bản mới nhất của mục dữ liệu. Việc lựa chọn SỐ lượng đại biểu dựa

vào tỉ lệ giữa thao tác đọc và ghi cùng với cosf (bandwidth...) khi thực hiện phương pháp giao tiếp giữa các nhóm.

### Cache-coherence protocols

Cache là một dạng đặc biệt của nhân bản, nó được điều khiển bởi client thay vì được điều khiển bởi server. Có nhiều giải pháp cho việc cache dữ liệu.

Với chiến lược phát hiện sự cố kết, là chiến lược để xác định khi nào thì sự không nhất quán thật sự bị phát hiện và từ đó loại bỏ những dữ liệu gây ra sự không nhất quán, thì có 2 giải pháp khác nhau.

- Với giải pháp tĩnh, tại thời điểm biên dịch chương trình thì những chỉ thị phụ sẽ được thêm vào để phát hiện những dữ liệu không nhất quán.
- Với giải pháp động thì tại thời điểm chạy chương trình thì có những đoạn mã để kiểm tra tính không nhất quán của dữ liệu cache với dữ liệu của server.

Với chiến lược ép buộc sự cố kết, là chiến lược để xác định xem dữ liệu cache được giữ nhất quán với dữ liệu lưu trên server như thế nào, thì có 2 cách để ép buộc giữ liệu phải cố kết với nhau. Đó là:

- Đề cho server gửi đi một thông điệp về sự không hợp lệ mỗi khi dữ liệu bị thay đổi.
- Cập nhật các kỹ thuật lan truyền.

Các thao tác ghi dữ liệu vào cache được tiến hành như sau:

- Với cache chỉ đọc ra (Read-only Cache) thì các cập nhật được thực hiện bởi S $\overline{E}$ T $\overline{V}\overline{E}$ T (bằng giao thức đẩy) hoặc bởi client (bằng giao thức kéo mỗi khi client nhận thấy dữ liệu cache đã cũ).
- Với cache ghi thẳng (Write- Through Cache) thì client sẽ thay đổi nội dung của cache, sau đó sẽ gửi các cập nhật đến cho server.
- Với cache ghi lại (Write-Back Cache) thì client trì hoãn sự lan truyền các cập nhật, cho phép nhiều cập nhật được tạo ra cục bộ sau đó gửi những cập nhật mới nhất cho server (việc này có khả năng ảnh hưởng mạnh đến hiệu năng).

## Chương 7: Chịu lỗi

### (Fault Tolerance)

#### 7.1 Chịu lỗi và một số khái niệm liên quan

##### 7.1.1 Một số khái niệm cơ bản.

###### a. Các khái niệm.

Tính chịu lỗi liên quan nhiều tới khái niệm hệ có thể tin cậy được (dependable system). Thuật ngữ "có thể tin cậy được" bao gồm các thuộc tính sau:

Tính sẵn sàng (availability): hệ thống có tính sẵn sàng là hệ thống luôn sẵn sàng hoạt động tốt ở mọi thời điểm.

Tính tin cậy (Reliability): một hệ thống có tính tin cậy là hệ thống có khả năng hoạt động trong một thời gian dài mà không bị gián đoạn, không xảy ra lỗi.

Tính an toàn (Safety): hệ thống có tính an toàn là hệ thống mà khi xảy ra lỗi

cũng không dẫn tới thảm họa. Các hệ thống cần phải có độ an toàn cao là các hệ thống điều khiển.

Khả năng bảo trì (Maintainability): hệ thống có khả năng bảo trì là hệ thống có khả năng phục hồi lại được sau khi có lỗi. Nếu sự phục hồi này diễn ra tự động thì có thể nói hệ thống này cũng có tính sẵn sàng cao.

Tính chịu lỗi còn có liên quan tới khái niệm điều khiển lỗi (Fault control). Điều khiển lỗi bao gồm ngăn ngừa lỗi, loại bỏ lỗi và dự báo lỗi với mục tiêu xây dựng thành công khả năng chịu lỗi cho hệ thống.

### b. Phân loại lỗi.

Lỗi được phân chia thành các loại sau:

Lỗi nhất thời (Transient faults): Là loại lỗi xuất hiện một lần rồi biến mất. Cách khắc phục: thực hiện lại hoạt động có lỗi này

Lỗi lặp (Intermittent faults): Là loại lỗi mà chúng xuất hiện, rồi biến mất, sau đó lại xuất hiện lại và cứ tiếp tục như thế. Lỗi này thường gây ra các hậu quả trầm trọng vì chúng rất khó xác định được.

Cách khắc phục: sử dụng bộ sửa lỗi cho hệ thống (fault doctor) để khắc phục lỗi.

Lỗi lâu dài (Permanent faults): Là loại lỗi vẫn tồn tại ngay cả khi thành phần gây lỗi đó đã được sửa chữa.

#### 7.1.2 Các mô hình lỗi.

Lỗi sụp đổ (crash failure): khi server gặp lỗi này thì nó sẽ bị treo, trước đó server vẫn hoạt động tốt cho đến khi ngừng hoạt động. Khi server gặp lỗi này, nó sẽ không thể làm gì được nữa. Một ví dụ hay gặp lỗi này là hệ điều hành của các máy cá nhân. Khi hệ điều hành ngừng hoạt động thì chỉ còn cách duy nhất là khởi động lại.

Lỗi bỏ sót (omission failure): là lỗi mà một server không thể đáp ứng được yêu cầu gửi tới nó. Người ta chia nó thành hai loại:

Lỗi khi nhận thông điệp gửi tới: gặp lỗi này, server không nhận được yêu cầu ngay cả từ client gần nó nhất và mặc dù kết nối giữa server với client đã được thiết lập. Lỗi khi nhận thông điệp chỉ làm cho server không nhận biết được các thông điệp gửi tới nó mà không ảnh hưởng đến trạng thái của server.

Lỗi khi gửi thông điệp: server vẫn nhận được các yêu cầu, vẫn hoàn thành yêu cầu đó nhưng vì một lý do nào đó lại không thể gửi kết quả tới máy đã yêu cầu. Một trong những lý do thường gặp là do bộ nhớ đệm gửi đầy. Trong trường

hợp gặp lỗi này, server cần chuẩn bị tình huống client sẽ gửi lại yêu cầu đã gửi

đó.

Lỗi thời gian (timing failure): là lỗi xảy ra khi server phản ứng lại quá chậm, sau cả thời gian cho phép. Trong một hệ thống luôn có các ràng buộc về mặt thời gian. Nếu bên gửi gửi đến bên nhận nhanh quá, bộ nhớ đệm của bên nhận

không đủ đê chúa thì sẽ gây ra lỗi. Tương tƯ, S€TV€T phản ứng lại chậm quá,

vượt quá khoảng timeout quy định sẵn cũng sẽ gây ra lỗi, ảnh hưởng đến hiệu năng chung của hệ thống.

Lỗi đáp ứng (Response failure): là lỗi khi server trả lời không đúng.

Đây là một

kiểu lỗi rất nghiêm trọng và được phân chia thành hai loại:

Lỗi về mặt giá trị: là lỗi khi server trả lời lại yêu cầu của client với giá {T<sub>i</sub>} không chính xác. Ví dụ khi sử dụng các máy tìm kiếm, kết quả trả về không hề liên quan gì tới yêu cầu của người sử dụng.

Lỗi về chuyển trạng thái: là lỗi khi server hoạt động trêch hướng khỏi luồng

điều khiển. Có nghĩa là server trả lời các yêu cầu được gửi tới một cách không theo như mong đợi.

Lỗi bất kì (Arbitrary failure): một server có thể tạo ra một lỗi bất kì ở bất kì

thời gian nào. Đây là loại lỗi nguy hiểm nhất. Có thể có hai khả năng xảy ra:

Thứ nhất: một server tạo ra một kết quả sai mà không thể phát hiện ra được.

Thứ hai: server bị lỗi có liên kết với các server khác tạo ra một kết quả sai.

Ta có thể xét một vǎo lỗi bất kì hay gấp sau : lỗi fail-stop, lỗi fail-silent và lỗi

fail-safe. Với fail-stop, server bị treo, ngừng hoạt động và có thông báo tới các

tiến trình khác. Với fail-silent, server đột ngột hoạt động chậm lại vì thê lầm

cho các tiến trình không thê kết thúc được, ảnh hưởng đến hiệu năng của hệ

thống. Lỗi fail-safe là lỗi mà khi server tạo ra kết quả ngẫu nhiên nhưng các

tiên trình nhận dạng các kết quả này là không có giá trị.

## 7.2 Các phương pháp che giấu lỗi.

### 7.2.1 Che giấu lỗi bằng dư thừa.

Có ba loại dư thừa: dư thừa thông tin, dư thừa thời gian và dư thừa vật lý.

Dư thừa thông tin : bô sung thêm các bit dư thừa để phát hiện lỗi và phục hồi lỗi. Ví dụ trong việc truyền dữ liệu thường thêm vào các bit kiểm tra chẵn lẻ, mã Haming, CRC... để phát hiện lỗi và khôi phục lỗi.

Dư thừa thời gian: khi một hoạt động đã được thực hiện, nếu dư thừa thời gian nó có thể được thực hiện lại. Kĩ thuật dư thừa thời gian phù hợp khi lỗi là ngắn và không liên tục. Ví dụ: khi một giao tác bị hủy giữa chừng, nó có thể được thực hiện lại mà không gây nguy hại gì.

Dư thừa vật lý: bô sung thêm tải nguyên

### 7.2.2 Khôi phục tiên trình.

#### a. Các vấn đề khi thiết kế.

Nguyên tắc: tổ chức các tiên trình giống nhau vào cùng một nhóm.

Hoạt động: khi nhóm nhận được thông báo thi thông báo này sẽ được gửi tới tất cả các thành viên trong nhóm. Nếu có tiên trình nào trong nhóm bị lỗi thì sẽ có tiên trình khác thay thế.

Đặc điểm: các nhóm này có thể là động. Tính động thể hiện ở các mặt sau:

Số lượng các nhóm là không cố định: có thể tạo thêm hay hủy bỏ một nhóm.

Số lượng các tiến trình trong cùng một nhóm là không cố định: một tiến trình có thể gia nhập hay rời khỏi nhóm.

Một tiến trình có thể là thành viên của nhiều nhóm trong cùng thời điểm.

Do tính động đó mà cần phải đưa ra các cơ chế quản lý nhóm: quản lý mỗi quan hệ giữa các nhóm và quản lý thành viên trong một nhóm.

Phân loại nhóm: dựa trên cấu trúc bên trong thì nhóm được phân thành hai loại:

Nhóm ngang hàng:

- Tất cả các tiến trình trong nhóm là ngang hàng nhau.
- Khi thực hiện một công việc nào đó sẽ phải có một quá trình bâu cử (vofte) để xác định xem tiến trình nào phù hợp để thực hiện công việc đó.
- **Ưu điểm:** khi một tiến trình bị lỗi thì chỉ làm cho kích thước của nhóm giảm đi chứ không ảnh hưởng đến hoạt động của cả nhóm.
- **Nhược điểm:** do phải có quá trình bâu cử nên tốn thời gian (delay & overhead).

Hình 40. Nhóm ngang hàng.

Nhóm phân cấp:

- Trong mỗi nhóm sẽ có một tiến trình giữ vai trò quản lý gọi là **coordinator**, còn các tiến trình khác đóng vai trò thực hiện (**worker**). Các tiến trình thực hiện chịu sự điều khiển của coordinator.
- Khi có yêu cầu gửi đến nhóm, yêu cầu này sẽ được gửi tới coordinator.

Coordinator sẽ quyết định xem tiến trình nào trong nhóm đảm nhiệm công việc đó một cách phù hợp nhất và chuyên yêu cầu nhận được đến tiến trình đó.

- **Ưu điểm:** không bị trễ như kiến trúc ngang hàng.
- **Nhược điểm:** khi coordinator gặp sự cố thi toàn bộ hoạt động của nhóm sẽ bị dừng lại.

#### Hình 47. Nhóm ngang hàng

Các phương pháp quản lý thành viên trong nhóm:

Phương pháp 1: dùng một server gọi là Ø8roUD S€TVeT

Server này chứa tất cả các thông tin về các nhóm và các thành viên của từng nhóm.

**Ưu điểm:** hiệu quả, dễ sử dụng

**Nhược điểm:** nếu server bị lỗi thì không thể quản lý được toàn bộ hệ thống và các nhóm có thể phải xây dựng lại từ đầu các công việc mình đã thực hiện.

Phương pháp 2: phương pháp phân tán.

Khi tiến trình muốn gia nhập hay rời khỏi nhóm thì nó phải gửi bản tin thông báo tới tất cả các tiến trình khác.

Phương pháp 3: yêu cầu việc gia nhập/ rời khỏi nhóm phải đồng bộ với bản tin gửi hay nhận.

Khi một tiến trình gia nhập nhóm nó sẽ nhận tất cả các bản tin từ nhóm đó.

Khi một tiến trình rời khỏi nhóm thì nó sẽ không được nhận bất kì bản tin nào

từ nhóm đó nữa và không một thành viên nào của nhóm cũ nhận được các bản tin từ nó

b. Che giấu lỗi và nhân bản.

Có hai phương pháp nhân bản : bảng giao thức primary-based và bảng giao thức replicated-write

Bảng giao thức primary-based: Các tiến trình trong nhóm tổ chức theo mô hình phân cấp. Nếu coordinator của nhóm chính dừng hoạt động thì coordinator của các nhóm sao lưu sẽ thực hiện các giải thuật để lựa chọn nhóm chính mới (mặc dù nó có thể đảm nhiệm công việc đó).

Bảng giao thức replicated-write : Các tiến trình trong nhóm tổ chức theo mô hình nhóm ngang hàng. Vấn đề là cần nhân bản với số lượng là bao nhiêu

7.2.3 Che giấu lỗi trong truyền thông client/server tin cậy.  
Việc che giấu lỗi trong hệ phân tán tập trung vào trường hợp có tiến trình bị lỗi.  
Nhưng ta cũng phải xét đến trường hợp các giao tiếp bị lỗi. Thông thường, một kênh giao tiếp có thể gặp các lỗi: lỗi sụp đổ, lỗi bỏ sót, lỗi thời gian và lỗi tùy ý. Việc xây dựng một kênh truyền thông tập trung vào che giấu lỗi sụp đổ và lỗi tùy ý.

a. Truyền thông điểm - điểm .

Trong hệ phân tán, truyền thông điểm - điểm tin cậy được thiết lập bằng cách sử dụng các giao thức truyền tin cậy như TCP. TCP che giấu được lỗi bỏ sót

bằng cách dùng cơ chế thông báo ACK/NACK và việc thực hiện truyền lại.

TCP không che giấu được lỗi SUP < đồ. Khi Xây Ta lỗi sụp đồ thì kết nối TCP sẽ

bị hủy. Chỉ có một cách để che giấu lỗi sụp đồ là hệ thống phải có khả năng tự động tạo một kết nối mới.

### b. RPC khi xảy ra lỗi và cách khắc phục

Với hệ thống RPC, năm lớp lỗi có thể xảy ra là:

- Client không định vị được server: Nguyên nhân gây lỗi là do server và client dùng các phiên bản khác nhau hoặc do chính server bị lỗi.  
Khắc phục bằng cách sử dụng các ngoại lệ (exception) để bắt lỗi như ở ngôn ngữ java và điều khiển tín hiệu (signal handle) như ở ngôn ngữ C. Hạn chế của phương pháp này là không phải ngôn ngữ nào cũng hỗ trợ ngoại lệ hay điều khiển tín hiệu. Nếu tự viết một ngoại lệ hay điều khiển tín hiệu thì sẽ phá hủy tính trong suốt.

- Bị mất bản tin yêu cầu từ client gửi đến server: Đây là loại lỗi dễ xử lý nhất: hệ điều hành hay client stub kích hoạt một bộ đếm thời gian (timer) khi gửi đi một yêu cầu. Khi timer đã trở về giá trị 0 mà không nhận được bản tin phản hồi từ server thì nó sẽ gửi lại yêu cầu đó. Nếu bên client nhận thấy có quá nhiều

yêu cầu phải gửi lại thì nó sẽ xác nhận rằng server không hoạt động và sẽ quay lại thành kiêng lỗi không định vị được server"

- Server bị lỗi ngay sau khi nhận được yêu cầu từ client: Lúc này lại phần chia thành hai loại:

Loại 1: Sau khi thực hiện xong yêu cầu nhận được thì server bị lỗi.  
Phương pháp khắc phục: sau đó server sẽ gửi thông báo hỏng cho client

Loại 2: Vừa nhận được yêu cầu từ client server đã bị lỗi ngay.  
Phương pháp khắc phục: client chỉ cần truyền lại yêu cầu cho. Vấn đề đặt ra lúc này là client không thể nói cho server biết yêu cầu nào là yêu cầu được gửi lại.

Khi gặp lỗi kiểu này, ở phía máy server sẽ thực hiện theo 3 kĩ thuật sau:

Kĩ thuật 1: đợi đến khi nào server hoạt động trở lại, nó sẽ có thể thực hiện yêu cầu đã nhận được trước khi lỗi đó. Như thế RPC thực hiện ít nhất một lần.

Kĩ thuật 2: server sau khi được khôi phục nó sẽ không thực hiện yêu cầu nhận được trước khi bị lỗi mà sẽ gửi lại thông báo hỏng cho client biết để client gửi lại yêu cầu. Với kĩ thuật này thì RPC thực hiện nhiều lần nhất.

Kĩ thuật 3: không, thực hiện gì để đảm bảo cả. Khi server bị lỗi, client không hề hay biết gì cả. Kiểu này, RPC có thể được thực hiện nhiều lần cũng có thể không thực hiện lần nào.

Còn ở client thì có thể thực hiện theo 4 chiến lược sau:

Một là: Client không thực hiện gửi lại các yêu cầu. Vì thế không biết bao giờ yêu cầu đó mới thực hiện được hoặc có thể không bao giờ được thực hiện.

Hai là: Client liên tục gửi lại yêu cầu: có thể dẫn tới trường hợp một yêu cầu  
được thực hiện nhiều lần.

Ba là: Client chỉ gửi lại yêu cầu nào đó khi không nhận được bản tin ACK phản hồi từ server thông báo đã nhận thành công. Trường hợp này, server dùng bộ đếm thời gian. Sau một khoảng thời gian xác định trước mà không nhận được ACK thì client sẽ gửi lại yêu cầu đó.

Bốn là: Client gửi lại yêu cầu nếu nhận được thông báo hỏng từ server.

- Mất bản tin phản hồi từ server gửi trả về client: Phương pháp khắc phục: thiết kế các yêu cầu có đặc tính không thay đổi giá trị (idempotent). Client đánh SỐ thứ tự cho các yêu cầu, server sẽ nhận ra được đâu là yêu cầu đã được gửi lại nhờ các số thứ tự này. Do đó server sẽ không thực hiện lặp lại các yêu cầu. Tuy nhiên server vẫn phải gửi trả về bản tin thông báo yêu cầu nào bị thất lạc. Hoặc ta có thể sử dụng một bit ở phần header của yêu cầu để phân biệt yêu cầu nào là yêu cầu đã được gửi lại.

- Client bị lỗi ngay sau khi gửi yêu cầu tới server: Client gửi yêu cầu tới server rồi bị lỗi trước khi nhận được trả lời từ Server gửi về. Công việc mà server thực hiện nhưng không có đích nào đợi để nhận được gọi là một “orphan”. Như thế sẽ gây lãng phí chu kỳ CPU.

Có 4 giải pháp được đưa ra trong trường hợp này là:

Một là: trước khi gửi đi yêu cầu nào đó, client stub sẽ tạo ra một bản ghi xác

định công việc cần thực hiện này và lưu lại. Như thế, khi được phục hồi sau khi lỗi, client sẽ lấy lại bản ghi đó và và việc thực hiện các orphan đang diễn ra sẽ dừng lại. Phương pháp này có nhiều nhược điểm: Chi phí để trang bị đĩa để lưu lại mỗi bản ghi cho mỗi RPC. Orphan có thể tự mình thực hiện RPC tạo ra một grandorphan nên rất khó xác định.

Hai là: chia thời gian hoạt động liên tục của client thành các số liên tục gọi là các thời kì. Mỗi khi các client khôi phục trở lại thì số chỉ thời kì này lại tăng lên một đơn vị. Lúc này client sẽ gửi thông báo đến tất cả các máy khác thông báo số thời kì mới của mình. Khi nhận được thông báo này thì các orphan sẽ dừng lại

Ba là: khi nhận được bản tin thông báo thời kì mới, mỗi máy sẽ kiểm tra xem mình có đang thực hiện một tính toán từ xa nào đó không. Nếu có, máy đó sẽ xác định xem client nào đã gửi yêu cầu này. Nếu không xác định được thì quá trình tính toán này sẽ bị hủy bỏ.

Bốn là: quy định mỗi RPC chỉ có một khoảng thời gian xác định T để thực hiện, sau khi gặp lỗi, client sẽ phả đợi thêm một khoảng thời gian T trước khi khởi động lại để nhận các orphan. Vẫn đề đặt ra là phải lựa chọn giá trị T như thế nào cho hợp lý.

#### 7.2.4 Che giấu lỗi trong truyền thông nhóm tin cậy (dùng multicasting)

a. Multicasting tin cậy cơ bản (Basic Reliable-multicasting).

Sau khi các tiên trình đã được phân nhóm thì một tiên trình khác muốn thực

hiện multicast tức là sẽ gửi bản tin tới tất cả các tiên trình trong nhóm đó.

Multicast tin cậy là phải có cơ chế để đảm bảo bản tin đó đến được tất cả các thành viên trong nhóm. Khi xảy ra lỗi thì sẽ áp dụng phương pháp sau để che giấu lỗi:

Phương pháp: đánh số các bản tin cần gửi. Các bản tin được lưu tại một buffer

của bên gửi và vẫn lưu ở đó cho đến khi nhận được bản tin ACK báo về từ bên

nhận. Nếu bên nhận xác định là bị mất một bản tin nào đó thì nó sẽ gửi về một

bản tin NACK để yêu cầu gửi lại. Và thông thường, bên gửi sẽ tự động gửi lại

bản tin sau trong khoảng thời gian xác định nào đó mà nó không nhận được bản

tin ACK báo về.

Hình 48. (a). Truyền bản tin (b). Bản tin phản hồi

b. Multicast tin cậy mở rộng.

Để tăng hiệu quả công việc khi làm việc với một SỐ lượng lớn các tiên trình thì

đưa ra mô hình multicast tin cậy mở rộng. Với mô hình này sẽ không gửi trả về

bản tin ACK báo nhận thành công mà chỉ gửi trả về cho tiên trình nhận bản tin

NACK thông báo khi có lỗi truyền. Việc này được thực hiện bằng giao thức

SRM (Scalable Reliable Multicasting).

Để có thể thực hiện multicast tin cậy cho một nhóm lớn các tiên trình thì thực

hiện tổ chức các nhóm theo cấu trúc dạng cây. Cấu trúc của cây :

- Gốc là nhóm chứa tiên trình gửi.

- Các nút là các nhóm có chứa tiên trình nhận.

#### Hình 49. Multicast tin cậy dạng cây

Việc thực hiện multicast được thực hiện cho các nhóm nhỏ đó. Việc chia thành các nhóm nhỏ hơn này cho phép sử dụng các kịch bản multicast tin cậy cho từng nhóm nhỏ đó.

Trong mỗi nhóm nhỏ sẽ đề cử một tiền trình làm coordinator. Coordinator có khả năng điều khiển việc truyền lại khi nhận được thông báo truyền lỗi. Coordinator của mỗi nhóm sẽ có bộ đệm (history buffer) riêng.

- Nếu Coordinator của mỗi nhóm không nhận được bản tin m thì nó sẽ gửi yêu cầu truyền lại tới coordinator của nút cha nó.
- Trong kịch bản truyền tin cậy sử dụng bản tin ACK thì khi coordinator nhận thành công một bản tin m nó sẽ gửi bản tin ACK tới coordinator của nút cha nó.
- Nếu coordinator của một nhóm nhận được bản tin ACK báo nhận thành công bản tin m của tất cả các tiền trình trong nhóm gửi về thì nó sẽ xóa bản tin m khỏi bộ đệm của nó.

Đánh giá: với phương pháp phân cấp này thì xảy ra vấn đề về cấu trúc cây. Rất nhiều trường hợp yêu cầu cây phải có cấu trúc động nên phải có một cơ chế tìm đường cho cây này

#### c. Multicast nguyên tử (Atomic multicast ).

Tư tưởng chính: khi một tiền trình muốn gửi bản tin cho một tập các tiền trình khác theo kiểu multicast, nó sẽ không gửi bản tin tới tất cả các tiền trình của nhóm chứa các tiền trình nhận mà chỉ gửi đến một nhóm nhỏ các tiền trình cần nhận bản tin đó.

Vấn đề đặt ra: phải đảm bảo gửi được bản tin tới tất cả các tiền trình trong nhóm hoặc không được gửi tới bất kỳ tiền trình nào nếu một tiền trình trong nhóm bị lỗi sụp đồ.

Một SÓ thuật ngữ:

Group view (khung nhìn nhóm): ý tưởng chính của atomic multicast là một tiền trình thực hiện multicast bản tin m thi chỉ thực hiện liên kết tới một danh sách các tiền trình cần nhận bản tin m đó chứ không phải toàn bộ nhóm. Danh sách các tiền trình này tương ứng với một khung nhìn nhóm (group view)- một tập nhỏ các tiền trình của một nhóm lớn.

View change (thay đổi khung nhìn): khi đang thực hiện multicast tới một group view G mà có một tiền trình xin gia nhập nhóm hay xin ra khỏi nhóm thì sự thay đổi ve này sẽ được gửi tới tất cả các thành viên còn lại trong nhóm. Do đó, các tiền trình còn lại trong G sẽ nhận được hai bản tin:

m: bản tin cần nhận

vc: bản tin thông báo có thay đổi trong G.

Nếu tất cả các tiền trình trong G đều chưa nhận được ve thi thao tác multicast bản tin m được thực hiện.

Nếu một trong số các tiến trình trong G đã nhận được ve thì pháo  
đảm bảo rằng  
không một tiến trình nào khác trong G được nhận m nữa

Đồng bộ ảo (Virtual synchronous).

Tư tưởng chính: đảm bảo bản tin chỉ được multicast tới tất cả các  
tiến trình  
không có lỗi. Nếu tiến trình gửi bị sụp đồ trong quá trình multicast  
thì quá trình  
này bị hủy ngay dù bản tin đó đã được gửi tới một vài tiến trình khác  
trong  
nhóm rồi.

Hình 50 Nguyên lý đồng bộ ảo

1: PI tham gia vào nhóm đã có sẵn ba thành viên: P2.P3, P4.

2: P2 thực hiện multicast bản tin tới tất cả các tiến trình còn lại.

3: P1 thực hiện multicast bản tin tới tất cả các tiến trình còn lại.

4: P3 multicast tới tiến trình P2, P4 thành công nhưng P1 chưa nhận  
được thì

P3 bị sụp đồ. Lúc này đồng bộ ảo sẽ hủy tất cả các bản tin đã được  
gửi trước đó

cho P2, P4, thiết lập trạng thái trước khi sụp đồ của P3 là chưa gửi  
bản tin đó.

5: nhóm lúc này chỉ còn PI, P2, P4 và P4 thực hiện multicast bản tin,

6: P3 được khôi phục và xem gia nhập lại nhóm.

7: P3 gia nhập nhóm thành công.

### 7.3 Cam kết phân tán.

Mô hình thiết lập cam kết phải là mô hình phân cấp và coordinator  
lãnh trách  
nhiệm thiết lập cam kết phân tán. Ở cam kết một pha đơn giản,  
coordinator

thông báo với tất cả các thành viên còn lại hoặc là thực hiện hoặc là không thực hiện một thao tác nào đó. Nếu thành viên nào đó không thực hiện được cũng không thể báo lại cho coordinator biết. Do đó người ta đưa mô hình mới đó là cam kết hai pha và cam kết ba pha

#### 7.3.1 Cam kết hai pha.

Xét một giao dịch phân tán với các thành viên là một tập các tiến trình chạy ở một máy khác với giả thiết không có lỗi xảy ra.

Cam kết hai pha gồm hai: Pha bầu cử (voting phase) và pha quyết định (Decision phase).

Với pha bầu cử: bao gồm hai bước thực hiện:

- Coordinator gửi một bản tin thông báo yêu cầu bầu cử VOTE\_REQUEST tới tất cả các thành viên trong nhóm.
- Sau khi nhận được bản tin VOTE\_REQUEST của coordinator, nếu có thành viên nào đó sẽ gửi lại cho coordinator thông báo chấp nhận bầu cử VOTE\_COMMITT, nếu không, sẽ gửi lại cho coordinator thông báo từ chối VOTE\_ABORITT.

Pha quyết định: gồm hai bước thực hiện:

- Coordinator tập hợp tất cả các bầu cử của các thành viên. Nếu tất cả đều đồng ý chấp nhận giao dịch thì coordinator sẽ gửi một bản tin

GLOBAL COMMIT tới tất cả các thành viên. Tuy nhiên, chỉ cần một thành viên gửi thông báo từ chối thì coordinator quyết định hủy giao dịch trên và sẽ

gửi một bản tin GLOBAL \_ ABORT cho tất cả các thành viên trong nhóm.

- Các thành viên sau khi đã gửi thông báo chấp nhận tới coordinator sẽ đợi phản hồi từ coordinator. Nếu nó nhận về thông báo GLOBAL \_ COMMIT thì giao dịch sẽ được chấp thuận. còn nếu nhận được GLOBAL ABORT thi giao dịch sẽ bị hủy.

Cam kết hai pha đưa ra một sơ đồ các trạng thái hữu hạn như hình 7.17 a, b (394).

Các trạng thái của một coordinator là: INIT, WATT, ABORET, COMMITIT. Còn các trạng thái của một thành viên bất kì là : INIT, READY, ABORFT, COMMILIT.

Hình ŠT (a) Máy trạng thái hữu hạn cho coordinator trong cam kết 2 pha (b).

Máy trạng thái hữu hạn cho thành viên

Nhược điểm của cam kết hai pha: Nhược điểm chính của cam kết hai pha là tốn nhiều thời gian chờ đợi. Cả coordinator và các thành viên còn lại đều phải chờ một bản tin nào đó được gửi đến cho mình.

Nhược điểm thứ hai là nếu coordinator bị lỗi thì hoạt động của cả hệ thống sẽ bị ảnh hưởng.

### 7.3.2 Cam kết 3 pha

Để khắc phục nhược điểm của cam kết hai pha trong trường hợp coordinator bị lỗi, người ta đưa ra mô hình cam kết ba pha. Các trạng thái khá giống hai pha

nhưng thêm một trạng thái PRECOMMIT.

Hình 52 (a) Máy trạng thái hữu hạn cho coordinator trong cam kết 2 pha (b).

Máy trạng thái hữu hạn cho thành viên

#### 7.4 Phục hồi.

Phục hồi là các phương pháp đưa trạng thái bị lỗi sang trạng thái lành (fault

Iree). Có hai cách tiếp cận cho phục hồi lỗi: phục hồi lùi (backward recovery) và

phục hồi tiến (forward recovery).

Phục hồi lùi: khi thực hiện phục hồi lùi sẽ thực hiện phục hồi trạng thái lành

của hệ thống trước khi có lỗi và cho hệ thống chạy lại từ điểm đó.

Để có thể

thực hiện được điều này phải sử dụng các điểm checkpoint. Tại các

điểm này

sẽ sao lưu trạng thái hiện hành của hệ thống để khi khôi phục sẽ cho

chạy

từ điểm checkpoint gần nhất. Việc thực hiện theo phương pháp này là

rất tốn kém.

Phục hồi tiến: chuyên hệ thống từ trạng thái lỗi sang trạng thái mới

với các

thông tin để tiếp tục thực hiện

Chương 8: An toàn - An ninh.

(Security)

8.1 Đặt vân đè.

§.1.1 Các mối đe dọa, chính sách và cơ chế an toàn , an ninh.

a. Các mối đe dọa.

Hệ thống máy tính luôn bị đe dọa bởi các nguy cơ mất an toàn. Một

trong

những công việc để bảo vệ hệ thống là làm sao giúp hệ thống tránh

khỏi các

nguy cơ đó. Có 4 loại các mối đe dọa an toàn:

Interception (chặn bắt): chỉ thành phần không được phép cũng có thể truy cập

đến các dịch vụ hay các dữ liệu, "nghe trộm" thông tin đang được truyền đi.

Interruption (đứt đoạn): là mối đe dọa mà làm cho dịch vụ hay dữ liệu bị mất

mất, bị hỏng, không thể dùng được nữa..

Modification (thay đổi): là hiện tượng thay đổi dữ liệu hay can thiệp vào các

dịch vụ làm cho chúng không còn giữ được các đặc tính ban đầu.

Fabrication (giả mạo): là hiện tượng thêm vào dữ liệu ban đầu các dữ liệu hay

hoạt động đặc biệt mà không nhận biết được để ăn cắp dữ liệu của hệ thống.

### b. Các cơ chế an toàn, an ninh.

Có 4 cơ chế an toàn, an ninh được đưa ra:

Mật mã (Cryptography): là việc thực hiện chuyển đổi dữ liệu theo một quy tắc

não đó thành dạng mới mà kẻ tấn công không nhận biết được.

Xác thực (Authentication): là các thao tác để nhận dạng người dùng, nhận dạng

client hay server..

Ủy quyền (tifHgiizsifgBij" chính là việc phân định quyền hạn cho mỗi thành

phần đã đăng nhập thành công vào hệ thống. Quyền hạn này là các quyền Sử

dụng dịch vụ, truy cập dữ liệu...

Kiểm toán (Auditing): là các phương pháp để xác định được client đã truy cập

đến dữ liệu nào và bằng cách nào.

### 8.1.2 Các vấn đề khi thiết kế.

#### a. Điều khiển (focus of control).

Có ba cách tiếp cận:

Chống các thao tác bất hợp lệ: việc này thực hiện bằng cách bảo đảm toàn vẹn  
chính các dữ liệu đó mà không quan tâm đến việc phân tích sự hợp  
lệ của thao  
tác.

Hình 53 Chống các thao tác bất hợp lệ

Chống các triệu gọi thao tác không được ủy quyền.: không bảo đảm  
toàn vẹn

dữ liệu mà tập trung vào các thao tác. Thao tác nào là bất hợp lệ sẽ bị  
hủy bỏ  
ngay.

Hình 54. Chống các triệu gọi thao tác không được ủy quyền

Chống người sử dụng không được ủy quyền: ở cách tiếp cận này lại  
tập trung  
vào quản lý người dùng. Xác định người dùng và các vai trò của họ  
trong hệ  
thống cứ không quan tâm đến đảm bảo dữ liệu hay quản lý các thao  
tác của  
người dùng.

Hình 55. Chống người sử dụng không được ủy quyền

#### b. Phân tầng các cơ chế an toàn (Layer Of security mechanism)

Một vấn đề quan trọng trong việc thiết kế một hệ thống an toàn là  
quyết định  
xem cơ chế an toàn an ninh được đặt ở tầng nào. Việc xác định vị trí  
đặt đó phụ  
thuộc rất nhiều vào yêu cầu của client về các dịch vụ an toàn, an  
ninhanh của từng  
tầng.

Trong một hệ phân tán, cơ chế an toàn, an ninh được đặt ở tầng middleware.

### c. Phân tán các cơ chế an toàn (Distribution of security mechanism)

Xét khái niệm CB (Trusted Computing Base): là tập hợp tất cả các cơ chế an

toàn, an ninh trong hệ phân tán, các cơ chế này phải tuân theo một ràng buộc an toàn nǎo đó.

#### S.1.3 Mật mã (Cryptography)

Một cơ chế an toàn, an ninh cơ bản trong hệ phân tán đó là mã mật.

Tư tưởng

cơ bản là: bên gửi mã hóa bản tin cần truyền, truyền bản tin đã mã hóa đi, bên nhận sẽ giải mã bản tin nhận được thành bản tin ban đầu.

CIOI:

Bản tin ban đầu là P.

Khóa mã hóa là Ek.

Bản tin được mã hóa theo khóa Ek là C:  $C=Ek(P)$ .

Khóa giải mã là Dk.

Bản tin được giải mã theo khóa giải mã:  $P=Dk(C)$ .

Có hai loại hệ thống mật mã: mật mã đối xứng (symmetric cryptosystem) và  
mật mã bất đối xứng (asymmetric cryptosystem)).

##### a. Mật mã đối xứng: dùng khóa bí mật..

Với mật mã đối xứng: khóa mã hóa và khóa giải mã là giống nhau.

Ta có:

$P=Dk(Ek(P))$ . Cả bên nhận và bên gửi đều phải có khóa trên, khóa phải được

giữ bí mật.

Nguyên lý chung của giải thuật DES (Data Encryption Standard):

Hình 56 nguyên lý chung của DES

Thực hiện trên các khôi dữ liệu 64 bit. Mỗi khôi này được mã hóa qua 16 vòng lặp, mỗi vòng có một khóa mã hóa 48 bit riêng. 16 khóa này được sinh ra từ 56

bit khóa chính.

Đầu vào của vòng lặp mã hóa thứ i là dữ liệu đã được mã hóa của vòng lặp thứ

(i-1). 64 bit dữ liệu qua mỗi vòng lặp được chia thành hai phần bằng nhau: Li-1 và Ri-I, cùng bằng 32 bit . Phần dữ liệu bên phải Ri-I được lấy làm trái của dữ liệu cho vòng sau: Ri-I= Li. Hỗn hợp với đầu vào là Ri-I và khóa Ki sinh ra khôi 32 bit được XOR với

Li-1 để sinh ra Ri.

Hình 57 .Một vòng mã hóa

Phương pháp sinh khóa của giải thuật DES:

Hình 58. Sinh khóa theo giải thuật DES

Mỗi khóa 48 bit cho mỗi vòng lặp được sinh ra từ khóa chính 56 bit như sau:

hoán vị khóa chính, chia đôi thành hai phần 28 bit. Tại mỗi VÒNØ, mỗi một nửa đó sẽ quay trái một hoặc hai bit, sau đó lấy ra 24 bit kết hợp với 24 bit của nửa còn lại tạo ra khóa.

b. Mật mã bắt đối xứng: dùng khóa công khai.

Mật mã bắt đối xứng: khóa mã hóa và khóa giải mã là khác nhau. Ta có:

$P=DkD(EKkD(P))$ . Trong hệ thông này, một khóa sẽ được giữ bí mật còn một khóa sẽ được công khai.

Xét giải thuật RAS (được đặt theo tên của các nhà phát minh ra nó: Rivest, Shamir, Adleman) :

Cách sinh khóa của giải thuật RAS: thực hiện theo 4 bước:

- Chọn 2 số chính lớn:  $p, q$
- Lĩnh  $n = p \cdot q$  và  $z = (p-1) \cdot (q-1)$
- Chọn một số  $d$  liên quan đến  $z$
- Lĩnh toán  $e$  sao cho  $e \cdot d = 1 \pmod{z}$ .

Như thế  $d$  có thể dùng để giải mã còn  $e$  dùng để mã hóa. Ta có thể công khai một trong hai SỐ này, tùy thuật toán.

Nguyên lý chung của giải thuật RAS:

Coi bản tin được truyền đi là một dãy các số nhị phân. Chia bản tin m đó thành các khối có kích thước cố định mi sao cho  $0 \leq mi \leq m$ . Ở bên gửi, với mỗi khối mi sẽ tính một giá trị  $c_i = mei \pmod{n}$  rồi gửi đi. Bên nhận sẽ giải mã bằng cách tính:  $mi = cdi \pmod{n}$ .

Như vậy, để mã hóa cần biết  $e$  và  $n$  còn để giải mã thi cần biết  $d$  và  $n$ .

## 8.2 Kênh an toàn (Secure channels).

### 8.2. Xác thực (Authentification).

#### a. Xác thực dựa trên khóa bí mật.

Nguyên lý chung: bên gửi muốn giao tiếp với bên nhận sẽ gửi một yêu cầu A tới bên nhận. Bên nhận trả lời bằng một yêu cầu RB. Bên gửi sẽ mã hóa yêu cầu RB bằng khóa bí mật KA,B và gửi về cho bên nhận. Bên nhận xác thực được bên gửi nhờ nhận biết được yêu cầu RB mình đã gửi trong gói tin vừa nhận. Lúc này bên gửi muốn xác thực bên nhận sẽ tiếp tục gửi yêu cầu RA tới bên nhận. Bên nhận sẽ lại mã hóa RA bằng khóa bí mật KA,B đó và gửi về cho bên nhận. Và như thế bên nhận đã xác định được bên gửi, sau đó quá trình trao đổi sẽ được thực hiện.

Hình 59 Xác thực dựa trên khóa bí mật  
Một mô hình cải tiến hơn là thu gọn số lượng bản tin chỉ còn lại 3 bản tin giữa bên nhận và bên gửi.

Hình 60. Xác thực dựa trên khóa bí mật nhưng dùng 3 bản tin

Nhưng hiện nay, giao thức hay được dùng là "reflection attack" như được mô tả trong hình vẽ sau:

Hình 61. Reflection Atfack

b. Xác thực dựa trên trung tâm phân phối khóa.

Nếu hệ thống gồm N host, mỗi host phải chia sẻ một khóa mật với N-I host khác thì hệ thống cần quản lý  $N \cdot (N-1)/2$  khóa, và mỗi host phải quản lý  $N-I$  khóa. Như vậy nếu N lớn sẽ rất khó khăn trong việc quản lý. Do đó, để khắc phục hiện tượng trên ta sử dụng trung tâm phân phối khóa KDC (Key Distribution Center).

Tư tưởng chính: bên gửi sẽ gửi bản tin tới trung tâm phân phối khóa thông báo  
mình muốn giao tiếp với bên nhận. KDC sẽ gửi cho cả bên gửi và  
bên nhận  
một bản tin có chứa khóa bí mật KA,B. Bản tin gửi cho bên nhận sẽ  
được mã  
hóa bằng KA,KDC.. Bản tin gửi cho bên gửi sẽ được mã hóa bằng  
KB,KDC.

Hình 62 Nguyên lý của KDC

Cách tiếp cận thứ hai là KDC sẽ gửi cả hai bản tin chứa khóa bí mật KA,KDC  
(KA,B) và KB,KDC (KA,B ) cho bên gửi và bên gửi có nhiệm vụ  
gửi cho bên  
nhận khóa đã được KDC mã hóa KB,KDC (KA,B ) đó.

Hình 63 Dùng ticket

c. Xác thực dựa trên khóa công khai.

Hình 64. Xác thực dựa trên khóa công khai.

Bên gửi mã hóa yêu cầu bằng khóa công khai K+B của bên nhận.  
Bên nhận  
này là nơi duy nhất có thể giải mã bản tin đó bằng K-B. Bên nhận sẽ  
mã hóa  
yêu cầu của bên ØrI cùng với yêu cầu của chính mình và khóa  
KA,B vừa tạo ra  
bằng khóa công khai K+A của bên gửi nhằm xác thực bên gửi. Cuối  
cùng, bên  
gửi sẽ gửi lại cho bên nhận yêu cầu RB của bên nhận đã gửi đi để  
xác thực.

S.2.2 Tính toán vẹn và tính mật của thông điệp.

a. Chữ kí số.

Chữ kí số đê đảm bảo tính toàn vẹn của thông điệp.  
Có nhiều cách thiết lập chữ kí số cho thông điệp:  
Cách 1: dùng hệ mật mã khóa công khai là RSA.

Hình 65 Chữ kí số cho một bản tin dùng khóa công khai  
Bên gửi sẽ mã hóa bản tin bằng khóa riêng K-AA của mình, sau đó  
sẽ mã hóa  
tiếp nội dung bản tin và phiên bản chữ kí bằng khóa công khai K+B  
của bên  
nhận. Bản tin được mã hóa này sẽ được truyền đi cùng bản tin m.  
Bên nhận sau  
khi nhận được bản tin sẽ giải mã gói tin, lấy phiên bản chữ kí của m  
và so sánh  
với m để xác thực rằng bản tin này được gửi từ bên gửi đó và cũng  
để kiểm tra  
xem có thay đổi trên đường truyền hay không.

Cách 2: dùng hàm băm.

Hàm băm H dùng để tính toán một bản tin có độ dài cố định là một  
chuỗi bit h  
từ một bản tin có độ dài tùy ý m. Nếu giá trị m thay bằng giá trị m'  
thì  $H(m)$ )  
cũng có giá trị khác giá trị  $h = H(m)$ , do đó ta có thể dễ dàng xác  
định được  
những thay đổi trên bản tin m trên đường truyền.

Hình 66. Chữ kí số cho một bản tin dùng message digest

Bên gửi sẽ tính toán các bản tin có độ dài cố định từ bản tin m và mã  
hóa bằng  
khóa riêng của mình. Bản tin được mã hóa này sẽ được truyền đi  
cùng bản tin  
m. Khi nhận, bên nhận giải mã bản tin và thực hiện so sánh với bản  
tin m đã  
được truyền đi để xác định được rằng bản tin này gửi từ bên gửi đó  
và đã được  
kiểm tra chữ kí số.

b. Khóa phiên

Trong một kênh trao đổi an toàn, sau pha xác thực sẽ tiến hành  
truyền thông.

Mỗi kênh truyền thông đó được xác định bởi một khóa phiên tương ứng. Khi phiên truyền kết thúc thì khóa phiên tương ứng cũng bị hủy bỏ.

### 8.2.3 Truyền thông nhóm an toàn

#### a. Truyền thông nhóm bí mật

Mô hình đơn giản là tất cả các thành viên trong nhóm sẽ cùng có một khóa bí mật để mã hóa và giải mã các bản tin. Điều kiện tiên quyết cho mô hình này là phải đảm bảo rằng tất cả các thành viên trong nhóm phía bên bia mật khóa đó.

Mô hình thứ hai là dùng một khóa bí mật cho từng cặp thành viên trong nhóm.

Khi một trong hai thành viên kết thúc phiên truyền thì thành viên còn lại vẫn sẽ dùng khóa đó để giao tiếp với thành viên khác trong nhóm. Với mô hình này phải duy trì tối N ( $N-1)/2$  khóa.

Mô hình thứ ba là dùng khóa công khai. Mỗi một thành viên trong nhóm sẽ phải duy trì một cặp khóa công khai và khóa riêng, trong đó khóa công khai được dùng bởi tất cả thành viên trong nhóm.

#### b. Server nhàn bản an toàn

Việc nhân bản các server thường dùng trong việc chịu lỗi cho hệ phân tán nhưng đôi khi cũng được dùng để đảm bảo tính tin cậy cho hệ thống.

### 8.3 Kiểm soát truy nhập (Access Control).

#### S.3.1 Các khía cạnh tổng quát trong kiểm soát truy cập.

##### a. Ma trận kiểm soát truy cập (Access Control Matrix).

Trong ma trận điều khiển truy cập, một hảng biêu diển cho một chủ thể (subJecf), một cột biêu diển cho một đối tượng (object). Cọi ma trận kiêm soát truy nhập là M. M[s.,o]: đưa ra danh sách các phép toán mà chủ thê s có thê yêu cầu trên đối tượng o. Khi một chủ thê s gọi một phương thức m của đối tượng o thì monitor sẽ kiểm tra trong danh sách M[s.,o]. nếu m không có trong danh sách này thì lời triệu gọi bị hủy bỏ.

Thông thường hệ thống phải làm việc với rất nhiều user nên có hàng nghìn chủ thê cần quản lý. Do đó xây dựng một ma trận thực như trên là không hợp lý.  
Giải pháp đề ra là sử dụng danh sách kiêm soát truy cập.

#### b. Danh sách kiêm soát truy cập (Access Control LISE).

Mỗi một đối tượng sẽ duy trì một danh sách các truy cập hợp lệ của các chủ thê muốn truy cập nó gọi là ACL nhờ đó tránh được sự tồn tại của các enfray rồng như ở ma trận kiểm soát truy nhập.

#### Hình 67 sử dụng ACL

#### c. Miền bảo vệ (Protection Domain8).

Với việc sử dụng AACL, tuy đã khắc phục được nhược điểm của ma trận kiêm soát truy nhập nhưng vẫn có kích thước lớn nên đã đưa ra cách sử dụng miến bảo vệ. Miễn bảo vệ là một tập các cặp (đối tượng. truy cập hợp lệ), mỗi Cặp này sẽ cho ta một đối tượng và các thao tác hợp lệ trên nó. Mỗi một yêu cầu đều thuộc một miền bảo vệ nào đó. Khi một yêu cầu gửi đến, monitor sẽ tìm trong miền bảo vệ tương ứng yêu cầu này.

Để đạt hiệu quả cao hơn, người ta dùng kết hợp miền bảo vệ với việc phân nhóm các đối tượng.

### 8.3.2 Tường lửa (Firewall).

Eirewall dùng để ngăn chặn các luồng không được phép. Firewall có hai loại chính là:

Packet - filtering gateway: loại này hoạt động như một router cho phép hoặc không cho phép gói tin chuyên qua mạng dựa trên địa chỉ nguồn và địa chỉ đích  
Ở phần header của gói tin. Loại này thường dùng để ngăn chặn các gói tin từ ngoài đi vào trong mạng.

Application - level gateway: loại firewall này không chỉ kiểm tra header của gói tin gửi đến hay gửi đi mà còn kiểm tra nội dung của gói tin đó.  
Một ví dụ đặc biệt cho loại này là proxygateway.

## S.4 Quản trị an toàn - an ninh (Security management).

### S.4.1 Quản trị khóa.

#### a. Thiết lập khóa.

Việc tạo ra khóa bí mật giữa bên truyền và bên nhận được thực hiện như sau:  
Bên A và bên B đều tạo ra hai số lớn là n và g - hai số này có thể được công khai. Bên A sẽ tạo ra một số lớn khác là x, bên B tạo ra số lớn y và giữ bí mật chúng. Bên A sẽ gửi cho bên B: n, g và  $(gx \bmod n)$ . Bên B sẽ thực hiện tính  $(gy \bmod n)$ . do đó sẽ xác định được khóa bí mật x của bên A. Đồng

thời, bên B cũng gửi cho bên A ( $gy \bmod n$ ). Bên A thực hiện tính toán ( $gy \bmod n$ )  
 $x = gxy \bmod n$  nhờ đó cũng xác định được khóa bí mật y của bên B.

Hình 68 Nguyên lý của Difie - Hellman key exchange

b. Phân phát khóa.

Trong hệ mã mật đôi xứng, khóa bí mật tạo ra phải được truyền đi trên kênh mật riêng .

Hình 69 Phân phát khóa theo kênh riêng.

Trong hệ mật mã dùng khóa công khai, khóa công khai phải đảm bảo cùng một cặp với một khóa bí mật. Khóa công khai được truyền đi như một bản rõ trên đường truyền và phải có hỗ trợ xác thực. Khóa bí mật được truyền đi trên một kênh riêng và cũng phải được xác thực.

Thông thường, khóa công khai thường được thay bằng một chứng chỉ khóa công khai (public - key certificate). Chứng chỉ này bao gồm một khóa công khai và một xâu định danh để xác định được khóa mật liên kết với nó.

b. Thời gian tồn tại của chứng chỉ.

Khi cần hủy bỏ một chứng chỉ ta có thể thực hiện theo nhiều phương pháp:

Cách 1: sử dụng danh sách các chứng chỉ bị hủy bỏ CRL (certification revocation list). Khi client kiểm tra một chứng chỉ thì nó cũng kiểm tra trong danh sách CRL để kiểm tra xem chứng chỉ này đã bị hủy hay không. Như thế mỗi client phải được cập nhật danh sách này thường xuyên.

Cách 2: mỗi chứng chỉ tự động hết hiệu lực sau một thời gian xác định nào đó.

Nhưng nếu muốn hủy chứng chỉ trước thời gian đó thì vẫn phải dùng đến danh sách CRL như trên.

Cách 3: giảm thời gian tồn tại có hiệu lực của một chứng chỉ xuống gần bằng 0.

Khi đó client phải thường xuyên kiểm tra chứng chỉ để xác định thời gian có hiệu lực của khóa công khai.

#### S.4.2 Quản trị nhóm an toàn.

Xét nhóm 7, khóa mật CKG được chia sẻ với tất cả các thành viên của nhóm

để mã hóa thông điệp của nhóm. Nhóm còn có thêm 1 cặp khóa công khai/riêng (KG+, KG-) để giao tiếp với các thành viên của nhóm khác.

Tiến trình P muốn tham gia vào nhóm G sẽ gửi yêu cầu tham gia JR.  
RP (Reply pad) và khóa bí mật KP,G được mã hóa sử dụng khóa công khai KŒ+ của nhóm. JR được gán bởi P và nó được gửi đi cùng với chứng chỉ chưa khóa công khai của P.

Khi một thành viên nhóm Q nhận một yêu cầu từ P, nó sẽ xác thực P, xác định

tem thời gian T để đảm bảo rằng P vẫn còn giá trị tại thời điểm gửi. Sau đó lấy ra khóa công khai của P để kiểm tra tính hợp lệ của JR.

Nếu P được chấp nhận vào nhóm, Q trả lại thông điệp GA nhận dạng P và chưa N (nonce). RP được sử dụng để mã hóa khóa giao tiếp của nhóm CKO. P sử dụng khóa KŒ- để mã hóa cùng với CKG. Sau đó thông điệp GA được gán cho Q sử dụng khóa KP,G.

### 8.4.3 Quản trị ủy quyền (Authorization management )

Sử dụng capability để xác định quyền truy cập tài nguyên của tiến trình chiếm

giữ nó. Một capability là một từ định danh 128 bit, cấu trúc bên trong được mô

tả như sau:

48 bit đầu tiên được khởi tạo khi đối tượng được tạo ra bởi server của đối tượng. 48 bit này được gọi là server port.

24 bit tiếp theo được sử dụng để xác định đối tượng tại server đã định sẵn.

§ bit tiếp theo xác định quyền truy cập của holder của capability

Trường check (48bit cuối) được dùng để tạo ra một capability thật (không thê giả mạo được).

Khi một đối tượng được khởi tạo, server của đối tượng đó chọn lấy một trường

check ngẫu nhiên và lưu trữ nó trong cả capability và trong cả table riêng của

SETVET

#### Sự ủy quyền(delegation)

Sự ủy thác quyền truy nhập là một kỹ thuật quan trọng để thực thi sự bảo vệ

trong hệ thống máy tính và đặc biệt hơn là trong hệ phân tán. Ý tưởng cơ bản

rất đơn giản: bằng VIỆC chuyên quyền truy nhập từ tiến trình này sang tiến trình

khác, nó sẽ trở nên dễ dàng hơn để phân tán công việc giữa các tiến trình mà

không làm ảnh hưởng tới việc bảo vệ tài nguyên.

Có vài cách để thực thi sự ủy quyền, một cách là sử dụng proxy.

