

HỌC VIỆN CÔNG NGHỆ Bưu Chính Viễn Thông
KHOA CÔNG NGHỆ THÔNG TIN 1

BÀI GIẢNG
HỆ THỐNG PHÂN TÁN
(Dùng cho đề cương INT 1 4.05)

PTT

Người soạn biên: ThS. Nguyễn Xuân Anh
Đơn vị công tác: Bộ môn Hệ thống thông tin
Khoa CNTT 1

MỤC LỤC

CHƯƠNG 1:	TỔNG QUAN VỀ CÁC HỆ THỐNG PHÂN TÁN.....	6
1.1	Hệ thống phân tán là gì.....	6
1.2	Phân loại các hệ thống phân tán	7
1.2.1	Các hệ thống điện toán phân tán.....	7
1.2.1.1	Hệ thống điện toán cụm.....	7
1.2.1.2	Hệ thống điện toán lưới	8
1.2.2	Các hệ thống thông tin phân tán	9
1.2.3	Các hệ thống lan tỏa phân tán.....	10
1.3	Các đặc trưng và mục tiêu thiết kế cơ bản của các hệ thống phân tán	11
1.3.1	Kết nối người sử dụng và tài nguyên hệ thống.....	11
1.3.2	Trong suốt đối với người sử dụng	11
1.3.3	Tính mở của hệ thống	12
1.3.4	Qui mô mở rộng hệ thống.....	12
1.4	Các kiến trúc của các hệ thống phân tán	12
1.4.1	Các kiểu hệ thống phân tán.....	12
1.4.2	Phân loại kiến trúc hệ thống phân tán	15
CHƯƠNG 2:	VÂN ĐÈ VÀ GIẢI PHÁP TRONG HỆ THỐNG PHÂN TÁN.....	20
2.1	Truyền thông	20
2.1.1	Cơ sở truyền thông	20
2.1.1.1	Giao thức mạng	20
2.1.1.2	Phân loại truyền thông.....	23
2.1.2	Gọi thủ tục xa	25
2.1.2.1	Cơ chế hoạt động của phương pháp gọi thủ tục từ xa	25
2.1.2.2	Vân đè truyền tham số	28
2.1.2.3	Gọi thủ tục từ xa bằng phương pháp không đồng bộ	29
2.1.2.4	Mô hình đối tượng phân tán	29
2.1.3	Truyền thông hướng thông điệp	32
2.1.3.1	Tính bền bỉ và tính đồng bộ trong trao đổi thông tin.....	32
2.1.3.2	Truyền tin nhanh hướng thông điệp	33
2.1.4	Truyền thông hướng luồng	34
2.1.5	Truyền thông theo nhóm.....	34
2.2	Đặt tên	34
2.2.1	Tên, định danh và địa chỉ.....	34
2.2.2	Đặt tên phẳng.....	35
2.2.2.1	Các giải pháp đơn giản	35
2.2.2.2	Cách tiếp cận dựa trên nguồn gốc	35
2.2.2.3	Bảng băm phân tán	35
2.2.2.4	Cách tiếp cận phân cấp	40
2.2.3	Đặt tên có cấu trúc	40
2.2.3.1	Không gian tên	40
2.2.3.2	Phân giải tên	41
2.2.3.3	Cài đặt không gian tên	42
2.2.3.4	Ví dụ về hệ thống tên miền.....	42

2.2.4	Đặt tên dựa trên thuộc tính	47
2.2.4.1	Dịch vụ thư mục	48
2.2.4.2	Cài đặt phân cấp LDAP	48
2.2.4.3	Cài đặt không tập trung	49
2.3	Đồng bộ	49
2.3.1	Đồng bộ đồng hồ	49
2.3.1.1	Đồng hồ vật lý	50
2.3.1.2	Hệ thống định vị toàn cầu	50
2.3.1.3	Các giải thuật đồng bộ đồng hồ	51
2.3.2	Đồng hồ logic	52
2.3.2.1	Đồng hồ logic Lamport	52
2.3.2.2	Đồng hồ vector	53
2.3.2.3	Các trạng thái toàn cục	54
2.3.3	Loại trừ tương hỗ	56
2.3.3.1	Giải thuật tập trung	56
2.3.3.2	Giải thuật không tập trung	57
2.3.3.3	Giải thuật phân tán	57
2.3.3.4	Giải thuật thẻ bài	58
2.3.3.5	So sánh các giải thuật loại trừ	58
2.3.4	Định vị toàn cầu các nút	58
2.3.5	Các giải thuật bầu chọn	59
2.3.5.1	Các giải thuật bầu chọn truyền thống	59
2.3.5.2	Bầu chọn trong môi trường không dây	60
2.3.5.3	Bầu chọn trong các hệ thống qui mô lớn	61
2.4	Tiến trình trong các hệ thống phân tán	61
2.4.1	Các luồng	62
2.4.1.1	Khái niệm luồng	62
2.4.1.2	Luồng trong các hệ thống độc lập	63
2.4.1.3	Cài đặt luồng	64
2.4.1.4	Luồng trong các hệ thống phân tán	65
2.4.2	Ảo hóa	67
2.4.2.1	Vai trò ảo hóa trong các hệ thống phân tán	67
2.4.2.2	Kiến trúc của các máy ảo	68
2.4.3	Máy khách	69
2.4.3.1	Các giao diện người dùng mạng	69
2.4.3.2	Tính trong suốt phân bố tài nguyên	69
2.4.4	Máy chủ	70
2.4.4.1	Các vấn đề thiết kế chung	70
2.4.4.2	Cụm máy chủ	71
2.4.4.3	Quản lý cụm máy chủ	72
2.4.5	Di trú mã	72
2.4.5.1	Các giải pháp di trú mã	72
2.4.5.2	Di trú và tài nguyên cục bộ	74
2.4.5.3	Di trú trong hệ thống không đồng nhất	74
2.5	Quản trị giao tác và điều khiển tương tranh	74
2.5.1	Các giao tác	75
2.5.2	Các giao tác lồng nhau	76
2.5.3	Các khóa	76

Hệ thống phân tán

2.5.4	Điều khiển tương tranh tối ưu.....	77
2.5.5	Trình tự nhãn thời gian	77
2.6	Phục hồi và chịu lỗi	78
2.6.1	Giới thiệu tính chịu lỗi.....	78
2.6.1.1	Một số khái niệm cơ bản	78
2.6.1.2	Các mô hình lỗi	78
2.6.1.3	Che giấu lỗi bằng biện pháp dư thừa	79
2.6.2	Tiến trình bền bỉ	80
2.6.2.1	Những vấn đề thiết kế.....	80
2.6.2.2	Che giấu lỗi và nhân bản	80
2.6.2.3	Thỏa thuận trong các hệ thống lỗi	81
2.6.2.4	Phát hiện lỗi	82
2.6.3	Truyền thông khách/chủ tin cậy	82
2.6.3.1	Truyền thông điểm – điểm	82
2.6.3.2	Các tình huống lỗi trong gọi thủ tục từ xa.....	82
2.6.4	Truyền thông nhóm tin cậy.....	84
2.6.4.1	Lược đồ truyền thông theo nhóm tin cậy cơ bản	84
2.6.4.2	Truyền tin nhóm tin cậy trong các hệ thống lớn.....	85
2.6.5	Cam kết phân tán	86
2.6.5.1	Cam kết hai pha	87
2.6.5.2	Cam kết ba pha	87
2.6.6	Phục hồi	88
2.6.6.1	Giới thiệu	88
2.6.6.2	Điểm kiểm tra	88
2.7	Bảo mật.....	88
2.7.1	Khái niệm chung.....	89
2.7.1.1	Tấn công thăm dò	89
2.7.1.2	Truy nhập trái phép	89
2.7.1.3	Tấn công từ chối dịch vụ	90
2.7.1.4	Phần mềm độc hại	90
2.7.2	Các kênh bảo mật	91
2.7.2.1	Xây dựng bức tường lửa	91
2.7.2.2	Xây dựng mạng riêng ảo	91
2.7.2.3	Hệ thống phát hiện và ngăn chặn đột nhập	91
2.7.2.4	Xác thực truy nhập	91
2.7.2.5	Giới thiệu một số phương pháp mã hóa	93
2.7.3	Kiểm soát truy cập	94
2.7.4	Quản lý bảo mật	95
2.7.4.1	Quản lý khóa	95
2.7.4.2	Quản trị nhóm an toàn	96
2.7.4.3	Quản lý ủy quyền	96
2.8	Tính nhất quán và vấn đề nhân bản	96
2.8.1	Khái niệm chung	97
2.8.2	Các mô hình nhất quán lấy dữ liệu làm trung tâm	97
2.8.2.1	Nhất quán liên tục	97
2.8.2.2	Nhất quán thứ tự các thao tác	98
2.8.3	Các mô hình nhất quán lấy máy khách làm trung tâm	102
2.8.3.1	Nhất quán sau cùng	102

Hệ thống phân tán

2.8.3.2	Nhất quán đọc đều	102
2.8.3.3	Nhất quán ghi đều.....	103
2.8.3.4	Nhất quán đọc kết quả ghi	103
2.8.3.5	Nhất quán ghi sau khi đọc	104
2.8.4	Quản lý các bản sao	104
2.8.4.1	Đặt vị trí máy chủ nhân bản	104
2.8.4.2	Nhân bản nội dung và vị trí lắp đặt	104
2.8.4.3	Phân bổ nội dung	105
2.8.5	Các giao thức nhất quán	106
2.8.5.1	Nhất quán liên tục	106
2.8.5.2	Các giao thức dựa trên bản chính	106
2.8.5.3	Các giao thức nhân bản ghi	108
2.8.5.4	Các giao thức gắn với cache	109
2.8.5.5	Cài đặt nhất quán lấy máy khách làm trung tâm	109
CHƯƠNG 3: CÔNG NGHỆ VÀ CÁCH TIẾP CẬN PHÁT TRIỂN HỆ THỐNG PHÂN TÁN		111
3.1	Mô hình gọi thủ tục từ xa	111
3.2	Mô hình DCOM	114
3.3	Kiến trúc CORBA	115
3.3.1	Các thành phần cơ bản của CORBA	115
3.3.2	Kiến trúc Corba và các yêu cầu phần mềm trung gian.....	135
3.3.3	Áp dụng CORBA trong xây dựng ứng dụng phân tán	135
3.4	Gọi phương thức từ xa	136
3.5	Dịch vụ web	138
3.5.1	Các thành phần trong kiến trúc dịch vụ Web	139
3.5.2	Cách thức trao đổi thông tin của dịch vụ Web	139
3.5.3	Quy trình xây dựng ứng dụng dịch vụ Web	141
3.6	Kiến trúc hướng dịch vụ	141
3.6.1	Giới thiệu về kiến trúc hướng dịch vụ	141
3.6.2	Các dịch vụ	142
3.6.3	Mô hình cặp lồng	143
3.6.4	Chu kỳ sống dịch vụ	144
3.6.5	Phân loại dịch vụ	144
3.6.6	Trục dịch vụ doanh nghiệp	145
3.6.7	Các mô hình kiến trúc dựa trên SOA	145
3.6.8	Các mẫu trao đổi thông điệp	145
TÀI LIỆU THAM KHẢO		147

CHƯƠNG 1: TỔNG QUAN VỀ CÁC HỆ THỐNG PHÂN TÁN

1.1 Hệ thống phân tán là gì

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 xuất phát từ nhu cầu sử dụng khả năng tính toán tốt hơn và hiệu quả hơn bằng cách kết hợp khả năng tính toán của từng máy tính độc lập, điều đó đã trở thành hiện thực dựa trên những tiến bộ về công nghệ mạng. Ba yếu tố quyết định tốc độ tính toán trong các máy tính bao gồm: Tốc độ của bộ vi xử lý trung tâm (CPU), bộ nhớ (RAM) và đường truyền trong bo mạch chủ (Bus). Đối với hệ thống phân tán, một yếu cầu mới quan trọng đã này sinh đó là vấn đề trao đổi thông tin giữa các máy tính, thiếu yếu tố này thì nhiệm vụ tính toán trên môi trường phân tán sẽ không thể thực hiện được. Tốc độ truyền dẫn trong các công nghệ mạng ngày càng tăng đã tạo điều kiện cho sự phát triển các ứng dụng phân tán, các máy tính có thể trao đổi thông tin và chia sẻ dữ liệu với nhau mà không phụ thuộc vào khoảng cách địa lý.

Hệ thống phân tán là hệ thống các thành phần được đặt trên các máy tính mạng, chúng trao đổi thông tin và phối hợp các hoạt động chỉ bằng cách truyền tin báo và người sử dụng cảm giác như là một hệ thống ~~đơn lẻ~~. Khái niệm phân tán được thể hiện bởi tính độc lập của từng máy tính nhưng ~~phải~~ ~~phối~~ hợp làm việc với nhau để người sử dụng không có cảm giác các thành phần rời rạc. Ví dụ, một hệ thống bán hàng bao gồm nhiều cửa hàng đặt tại những vị trí khác nhau, việc nhập thông tin hàng hóa được thực hiện tại nhiều vị trí khác nhau (các cửa hàng, nhà kho, ...), tuy nhiên các nhân viên khai thác đều có thể tìm thấy thông tin theo yêu cầu của mình như thể các thông tin đó đang được lưu trữ trên máy tính ~~của~~ người sử dụng.

Nhìn chung việc xây dựng các ứng dụng phân tán phức tạp hơn nhiều so với các ứng dụng tập trung, trong nhiều trường hợp bắt buộc phải xây dựng các ứng dụng phân tán vì những lý do sau:

- **Yêu cầu tính toán phân tán:** Ứng dụng chạy trên nhiều máy tính khác nhau nhằm tận dụng khả năng tính toán song song hoặc nhằm mục đích sử dụng khả năng tính toán của các máy tính chuyên dụng.
- **Yêu cầu về khả năng xử lý lỗi:** Yêu cầu này liên quan tới các hệ thống cần phải đảm bảo an toàn tuyệt đối ngay cả khi có sự cố xảy ra, điều này được thực hiện bằng cách tăng số lần tính toán cho cùng một nhiệm vụ nhằm mục đích kịp thời phát hiện và xử lý lỗi.
- **Chia sẻ tài nguyên:** Những người sử dụng trao đổi thông tin với nhau thông qua một ứng dụng trên mạng. Mỗi người sử dụng chạy một ứng dụng phân tán trên máy tính của mình và chia sẻ các đối tượng sử dụng. Một số ứng dụng phải chạy trên nhiều máy tính vì dữ liệu được đặt phân tán trên mạng liên quan đến quyền quản lý và quyền sở hữu dữ liệu: cho phép truy nhập dữ liệu từ xa nhưng không cho phép sao chép để lưu giữ cục bộ.

Trong các hệ thống phân tán, môi trường mạng đóng vai trò quan trọng trong việc phân phát thông tin đến các thành phần và tập hợp kết quả tính toán của các thành phần đó. Các máy tính kết nối với nhau trên mạng đảm nhiệm chức năng truyền thông cho các ứng dụng, chúng không chia sẻ bộ nhớ cho nhau do đó không thể sử dụng các

biến toàn cục để trao đổi thông tin, thông tin trao đổi giữa các máy tính chỉ được thực hiện thông qua cơ chế trao đổi tin báo. Mạng là tài nguyên chung của hệ thống do đó khi xây dựng hệ thống phân tán cần phải xem xét đến các vấn đề như: Băng thông, các điểm có thể xảy ra sự cố, bảo mật và an toàn dữ liệu, đồng bộ tiến trình. Quá trình triển khai các ứng dụng trong hệ thống phân tán thường gặp một số khó khăn sau:

- Trên mạng có nhiều loại máy tính và thiết bị mạng của nhiều nhà sản xuất khác nhau và các máy tính được cài đặt các hệ điều hành khác nhau.
- Khó tích hợp các phần mềm vì chúng được phát triển trên các ngôn ngữ khác nhau.
- Thời gian phát triển phần mềm lớn do đó thường kéo theo chi phí xây dựng hệ thống cao.

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 lớp giao vận) giải quyết các vấn đề như phát hiện và sửa lỗi, định tuyến...., những vấn đề này thường do hệ điều hành đảm nhiệm. Tuy nhiên việc áp dụng 3 lớp trên (phiên làm việc, trình diễn và lớp ứng dụng) đã được sử dụng trong các sản phẩm nền của hệ thống phân tán nhằm mục đích xử lý các thủ tục kết nối giữa các thành phần phân tán và thể hiện cấu trúc dữ liệu phức tạp của các ứng dụng, các sản phẩm như vậy gọi là phần mềm trung gian. Các ứng dụng phân tán thường được xây dựng ~~dựa~~ trên nền hệ điều hành và các thư viện hỗ trợ việc xử lý phân tán, chúng ~~thực~~ hiện các nhiệm vụ xác định vị trí của các máy tính, đồng bộ và mã hóa thông tin.

1.2 Phân loại các hệ thống phân tán

1.2.1 Các hệ thống điện toán phân tán

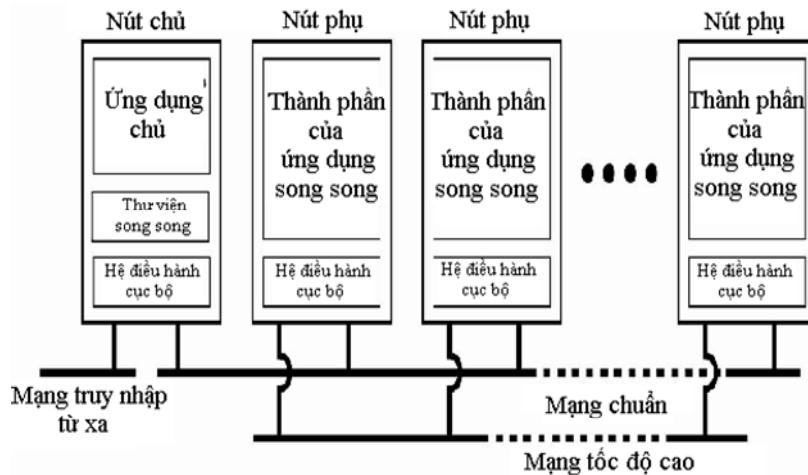
Điện toán phân tán thường được ~~sử~~ dụng trong các tác nghiệp yêu cầu hiệu năng cao, nó bao gồm hai nhóm: Điện toán cụm và điện toán lưới. Trong hệ thống điện toán cụm, các máy tính sử dụng cùng hệ điều hành và kết nối với nhau qua mạng nội bộ tốc độ cao. Điện toán lưới bao gồm nhiều hệ thống phân tán thuộc về nhiều miền quản lý khác nhau và thường không đồng nhất về phần cứng cũng như hệ điều hành.

1.2.1.1 Hệ thống điện toán cụm

Tỉ lệ giữa giá thành và hiệu suất xử lý của máy tính cá nhân ngày càng giảm là cơ hội để xây dựng hệ thống điện toán cụm. Các máy tính cá nhân được cài đặt một loại hệ điều hành và kết nối với nhau trong mạng tốc độ cao. Điện toán cụm sử dụng kỹ thuật xử lý song song trên nhiều máy tính để thực hiện tính toán.

Một ví dụ khá quen biết của điện toán cụm là hệ thống Beowulf được xây dựng dựa trên hệ điều hành Linux (hình 1.1), mỗi cụm bao gồm nhiều nút trong đó có một nút chủ (Master) đảm nhiệm chức năng xấp đặt vị trí của các thành viên khác trong chương trình song song, quản lý hàng đợi các công việc và giao tiếp với người dùng trong hệ thống. Như vậy, nút chủ chỉ việc chạy phần mềm trung gian cần thiết cho các chương trình thực hiện và quản lý cụm, trong khi đó các nút tính toán sẽ không cần gì khác ngoài hệ điều hành chuẩn. Một thành phần quan trọng của phần mềm trung gian là thư viện thực thi chương trình song song, những thư viện này cung cấp các phương tiện trao đổi thông tin dựa trên thông điệp nhưng chưa có khả năng xử lý lỗi, bảo mật...

Hệ thống phân tán



Hình 1.1 Hệ thống điện toán cụm

Một ví dụ khác của điện toán cụm là hệ thống MOSIX được xây dựng dựa trên cách tiếp cận đối xứng, nghĩa là nó cung cấp một hình ảnh đơn của hệ thống cụm. Độ trong suốt rất cao của MOSIX đạt được bằng cách di trú tiến trình, người dùng có thể khởi tạo một tiến trình trên một nút nhưng tiến trình đó có thể di trú sang nút khác để thực hiện nhằm tận dụng tối đa tiềm năng xử lý của mỗi nút trong hệ thống.

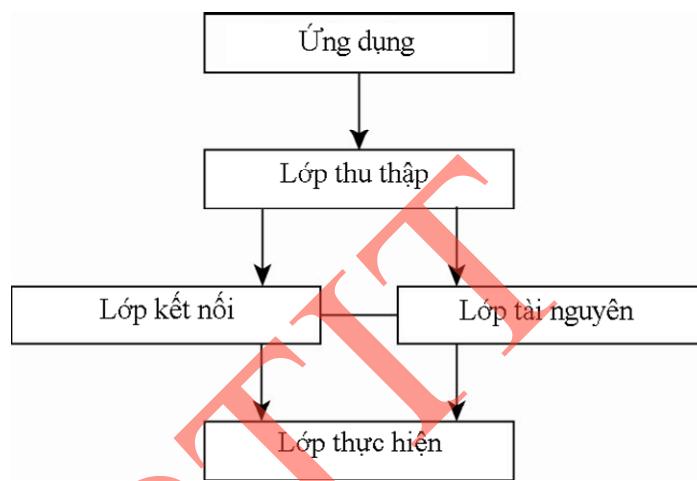
1.2.1.2 Hệ thống điện toán lưới

Hệ thống điện toán lưới 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 trong hệ thống điện toán lưới là việc lấy tài nguyên (máy tính, thiết bị ngoại vi, cơ sở dữ liệu...) từ các cơ quan khác nhau nhưng phải cho phép các nhóm người dùng thuộc các cơ quan cộng tác với nhau, như vậy sự cộng tác đó được thực hiện dựa trên cơ quan ảo, người dùng thuộc về một cơ quan ảo thì có quyền truy nhập đến các tài nguyên của cơ quan ảo đó.

Với đặc tính đó, nhiều phần mềm hệ thống điện toán lưới phát triển xung quanh việc truy nhập tài nguyên từ các vùng quản trị khác nhau cho những người dùng và ứng dụng thuộc về một cơ quan ảo, do đó tiêu điểm của hệ thống điện toán lưới thường là những vấn đề liên quan tới kiến trúc hệ thống. Hình (...) là một mẫu kiến trúc phân tầng thường được ứng dụng trong các hệ thống điện toán lưới. Kiến trúc này bao gồm 04 tầng:

- Tầng kết cấu (Fabric): Cung cấp giao diện để truy nhập tài nguyên cục bộ trại một trang riêng. Các giao diện này được làm để 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 (ví dụ khóa tài nguyên)
- Tầng kết nối (Connectivity): 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 giao thức bảo mật, tính năng bảo mật có thể cho một tài khoản và cũng có thể cho một ứng dụng (tất cả những người dùng sử dụng ứng dụng đó).

- Tầng tài nguyên (Resource): 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 tập trung (collective): 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 và thường không phải là những giao thức đã được chuẩn hóa (để đảm bảo cung cấp dịch vụ theo yêu cầu của tầng ứng dụng)
- Tầng ứng dụng (Application): Bao gồm các ứng dụng vận hành bên trong cơ quan ảo và sử dụng môi trường điện toán lưới.

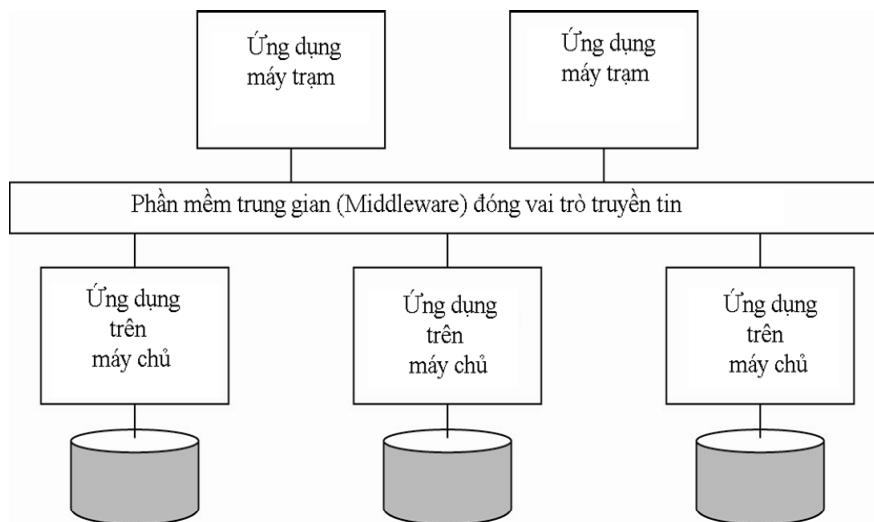


Hình 1.2 Kiến trúc phân tầng cho hệ thống điện toán lưới

Trong các hệ thống điện toán lưới, các tầng tập trung, kết nối và 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. Quan sát cho thấy, việc cung cấp các thông tin riêng lẻ trong các hệ thống điện toán khá phổ biến, điều này đã dẫn đến quan điểm về kiến trúc dịch vụ lưới mở (Open Grid Service Architecture).

1.2.2 Các hệ thống thông tin phân tán

Một trường hợp khác trong hệ thống phân tán đó là các ứng dụng mạng qui mô lớn, dữ liệu được đặt ở nhiều nơi nhưng việc xử lý ở mỗi nơi liên quan đến những nơi khác. Trong nhiều trường hợp, các máy chủ chạy tiến trình cung cấp dịch vụ xử lý cho máy khách, máy khách đơn thuần chỉ gửi yêu cầu và nhận về kết quả đã được máy chủ xử lý. Tuy nhiên một yêu cầu đặt ra là cần phải có sự phối hợp xử lý giữa các máy chủ, 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.



Hình 1.3 Phần mềm trung gian là phương tiện truyền tin trong EAI

Các phần mềm ngày càng tinh xảo hơn và chúng lần lượt tách chúng thành các thành phần (ví dụ phân biệt thành phần cơ sở dữ liệu với thành phần xử lý), như vậy việc tích hợp hệ thống phải cho phép các thành phần trao đổi thông tin trực tiếp với nhau, từ đó dẫn đến ngành công nghiệp lớp tích hợp ứng dụng doanh nghiệp (Enterprise Application Integration - EAI). Thực tế, các thao tác thực hiện trong cơ sở dữ liệu (CSDL), thường được thực hiện dưới dạng các giao tác. Phần mềm ứng dụng càng tách biệt với dữ liệu thì càng cần phải có các phương tiện để tích hợp chúng độc lập với CSDL, đặc biệt các thành phần ứng dụng phải có khả năng trao đổi thông tin trực tiếp với nhau chứ không phải chỉ là những phương tiện Yêu cầu/Trả lời như trong các hệ thống xử lý giao tác.

1.2.3 Các hệ thống lan tỏa phân tán

Các hệ thống điện toán phân tán và hệ thống thông tin phân tán đều có chung đặc điểm đó là tính ổn định của chúng, nghĩa là các nút mạng đều cố định và đường truyền kết nối mạng chất lượng cao tương đối ổn định. Ở một mức độ nào đó, tính ổn định này được thực hiện bằng nhiều kỹ thuật khác nhau (sẽ trình bày trong các chương tiếp theo của môn học này) nhằm đạt được tính trong suốt phân tán. Ví dụ với sức mạnh của các kỹ thuật che giấu lỗi và phục hồi sẽ cho chúng ta cảm giác thỉnh thoảng mới có một vài lỗi xảy ra. Tương tự như vậy chúng ta cũng có thể che giấu các khía cạnh liên quan đến vị trí các nút trên mạng, thực tế cho phép người dùng và các ứng dụng tin rằng các nút vẫn đang hoạt động.

Tuy nhiên, vấn đề đã trở nên rất phức tạp khi xuất hiện các thiết bị di động và thiết bị nhúng, chúng ta phải đương đầu với các hệ thống phân tán mà ở đó tính không ổn định là điều tất yếu. Các thiết bị trong hệ thống loại này thường là những thiết bị di động với đặc trưng là vị trí không ổn định và kết nối mạng không dây, do đó các hệ thống này được gọi là hệ thống lan tỏa phân tán (Distributed pervasive system). Hệ thống này thiếu vắng sự kiểm soát nhân công, cấu hình của các thiết bị này do chủ sở hữu thiết bị đó qui định, nếu không thì thiết bị sẽ tự động khám phá môi trường và lựa chọn cấu hình được cho là phù hợp nhất. Để sự lựa chọn được chính xác nhất, Grimm đã đưa ra các yêu cầu sau cho các hệ thống lan truyền 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 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. Với quan niệm việc kết nối mạng của các thiết bị thường gián đoạn hoặc thay đổi thì không gian lưu trữ thông tin có thể truy nhập được cũng sẽ phải thay đổi theo thời gian. Với sự hiện diện của khả năng di động thì các thiết bị phải dễ dàng thích nghi với môi trường cục bộ, chúng phải có khả năng dễ dàng phát hiện các dịch vụ và phản hồi theo các dịch vụ đó. Sự trong suốt về mặt phân bổ không chỉ ở trong hệ thống lan tỏa mà thực tế còn ở sự phân bổ về dữ liệu, xử lý và điều khiển, vì lẽ đó tốt hơn hết là phải phơi bày chứ không nên che giấu chúng.

1.3 Các đặc trưng và mục tiêu thiết kế cơ bản của các hệ thống phân tán

Người sử dụng luôn mong muốn có được các phần mềm thân thiện, tốc độ xử lý nhanh, dễ dàng cá nhân hóa các chức năng và đảm bảo an toàn thông tin. Sau đây chúng ta sẽ tóm tắt bốn mục tiêu cơ bản khi xây dựng một hệ thống phân tán.

1.3.1 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 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....

1.3.2 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 (Location): Ẩn nơi lưu trữ thông tin.
- Di chuyển (Migration): Ẩn quá trình chuyển vị trí lưu trữ dữ liệu
- Đặt lại vị trí (Relocation): Ẩn quá trình di chuyển dữ liệu mà không làm gián đoạn hoạt động của người sử dụng.
- Nhân bản (Replication): Che giấu việc tạo ra bản sao dữ liệu
- Tương tranh (Concurrency): Che giấu việc chia sẻ tài nguyên cho nhiều người sử dụng
- Lỗi (Failure): Che giấu lỗi và phục hồi tài nguyên
- Bền bỉ (Persistence): Che giấu tài nguyên phần mềm được tải vào bộ nhớ hay ở trên ổ đĩa.

1.3.3 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....

1.3.4 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 bảo đảm ít ảnh hưởng tới hiệu suất hoạt động của hệ thống. Trong cả hai trường hợp mở rộng trên, cần phải bảo đảm khả năng quản trị hệ thống.

1.4 Các kiến trúc của các hệ thống phân tán

Dựa trên cách kết nối và cách trao đổi thông tin giữa các thành viên, hệ thống phân tán có thể phân thành hai loại: Hệ thống nhiều bộ vi xử lý (các bộ vi xử lý dùng chung bộ nhớ) và 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.

1.4.1 Các kiểu hệ thống phân tán

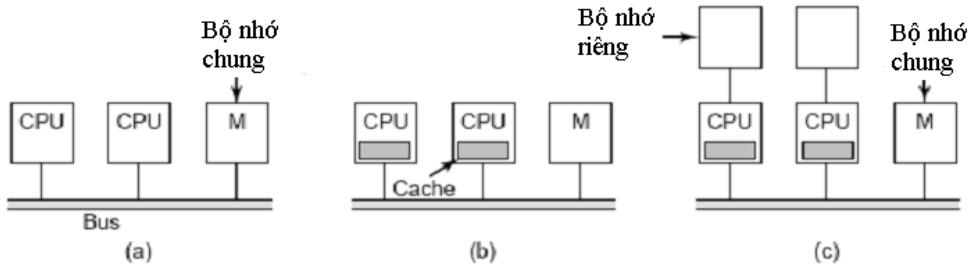
Trong hệ thống nhiều bộ vi xử lý, các bộ vi xử lý dùng chung bộ nhớ RAM, việc trao đổi thông tin giữa các bộ vi xử lý được thực hiện bằng cách đọc/ghi các ô nhớ. Ngoài các chức năng truyền thống của hệ điều hành (xử lý gọi hệ thống, quản lý bộ nhớ, quản lý tập tin, quản lý thiết bị vào ra), hệ điều hành của hệ thống nhiều bộ vi xử lý phải thực hiện các chức năng đặc biệt như: đồng bộ tiến trình, quản lý tài nguyên, lập lịch làm việc. Trước hết, chúng ta sẽ tìm hiểu sơ bộ về phần cứng của hệ thống nhiều bộ vi xử lý sau đó sẽ đề cập tới các vấn đề về hệ điều hành trong các hệ thống này. Hệ thống nhiều bộ vi xử lý đều có đặc điểm chung là các đơn vị xử lý trung tâm đều được kết nối vào kênh kết nối chung trong bo mạch chủ và truy nhập trực tiếp vào bộ nhớ dùng chung.

Hình 1.4-a minh họa hệ thống nhiều bộ vi xử lý dựa trên một kênh truyền, các đơn vị xử lý trung tâm và các mô-dun bộ nhớ dùng chung một kênh truyền để trao đổi thông tin. Trước khi truy nhập ô nhớ, đơn vị xử lý trung tâm phải kiểm tra xem kênh truyền có bận hay không. Nếu rồi, đơn vị xử lý trung tâm đặt địa chỉ ô nhớ lên kênh, phát tín hiệu điều khiển và chờ cho đến khi bộ nhớ đặt giá trị của ô nhớ đã yêu cầu lên kênh truyền.

Nếu kênh truyền bận, đơn vị xử lý trung tâm phải chờ cho đến khi kênh truyền rỗi, như vậy nảy sinh vấn đề về quản lý tương tranh. Đối với hệ thống chỉ có hai hoặc ba đơn vị xử lý trung tâm thì việc quản lý tương tranh tương đối đơn giản, vấn đề sẽ trở nên khá phức tạp đối với hệ thống có 32 hoặc 64 đơn vị xử lý trung tâm. Nói chung, hệ thống sẽ bị giới hạn bởi băng thông của kênh truyền và hầu hết các CPU sẽ lãng phí thời gian chờ đọc ô nhớ. Để giải quyết vấn đề này, người ta thêm bộ nhớ đệm

Hệ thống phân tán

vào mỗi CPU (xem hình 1.4-b), bộ nhớ đệm đó có thể được tích hợp trong CPU, bên cạnh, nằm trên bo mạch bộ vi xử lý hoặc tổ hợp các phương án trên. Bộ nhớ đệm trao đổi thông tin với bộ nhớ dùng chung theo phương pháp đọc/ghi từng khối (các khối 32 hoặc 64 byte), đơn vị xử lý trung tâm sẽ đọc/ghi các ô nhớ trong bộ nhớ đệm, như vậy sẽ giảm đáng kể lưu lượng trên kênh truyền chung.



- (a) Không có bộ nhớ đệm.
- (b) Có bộ nhớ đệm.
- (c) Có bộ nhớ đệm và bộ nhớ riêng.

Hình 1.4 Ba loại hệ thống nhiều bộ xử lý dựa trên một kênh

Mỗi khối bộ nhớ được đánh dấu bởi một trong hai trạng thái: chỉ đọc hoặc đọc/ghi. Nếu CPU muốn ghi một ô nhớ mà ô nhớ đó xuất hiện trong các bộ nhớ đệm khác, phần cứng của kênh truyền sẽ phát hiện tín hiệu ghi và chuyển tín hiệu đó đến tất cả các bộ nhớ đệm khác. Nếu các bộ nhớ đệm này đã có nội dung giống như trong bộ nhớ, chúng có thể chối bỏ bản sao đó và cho phép bộ điều khiển ghi chốt khối bộ nhớ đệm trong bộ nhớ dùng chung trước khi thay đổi ô nhớ đó. Một số bộ nhớ đệm đang thay đổi, nó phải ghi lại nội dung ô nhớ đó vào bộ nhớ dùng chung trước khi yêu cầu ghi có thể tiếp tục thực hiện hoặc chuyển trực tiếp giá trị đó đến bộ điều khiển ghi qua kênh truyền.

Hình 1.4-c minh họa một kiến trúc khác, CPU không chỉ có bộ nhớ đệm mà còn có bộ nhớ cục bộ riêng được truy nhập bằng kênh riêng. Để sử dụng cấu hình này một cách tối ưu, trình biên dịch sẽ đặt tất cả chương trình các hàng số, dữ liệu chỉ đọc và các biến số trong bộ nhớ riêng. Bộ nhớ dùng chung khi đó sẽ được dùng cho các biến dùng chung. Nói chung, kiến trúc này giảm đáng kể lưu lượng trên kênh chung nhưng nó đòi hỏi trình biên dịch phải có sự phối hợp rất chặt chẽ.

Kiến trúc UMA chỉ phù hợp với các hệ thống có ít CPU, khi số lượng CPU tăng lên sẽ phải chi phí lớn cho vấn đề xử lý chuyển mạch. Một kiến trúc khác được đề xuất là kiến trúc truy nhập bộ nhớ không đồng nhất. Giống như đa xử lý đối xứng, là công nghệ mở rộng tính khả biến của máy chủ bằng cách bổ sung thêm bộ xử lý. Cả hai công nghệ này đều cho phép người dùng khởi đầu với những máy chủ tương đối nhỏ và sau đó bổ sung bộ xử lý nếu ứng dụng phát triển thêm. Đối với hầu hết máy chủ đối xứng, việc bổ sung thêm sau khi đã có 8 bộ xử lý rất đắt tiền mà hiệu suất đạt được không cao nhưng NUMA thì cho phép mở rộng hơn thế nhiều - đến 256 bộ xử lý, liên kết với nhau trong một máy.

Tương tự hệ thống UMA, NUMA cho phép khai thác sức mạnh kết hợp của nhiều bộ xử lý mà mỗi bộ xử lý truy cập một cụm bộ nhớ chung. Tuy nhiên, các bộ xử

lý được phân thành những nhóm nhỏ hay “nút”, trong đó tất cả các bộ xử lý đều liên kết với nhau. Chẳng hạn, một máy chủ 16 bộ xử lý có thể sắp xếp thành bốn nút, mỗi nút có bốn bộ xử lý và có bộ nhớ riêng. NUMA làm giảm tình trạng tắc nghẽn bus của kiến trúc đối xứng bằng cách để cho các bộ xử lý trong một nút giao tiếp với nhau và với bộ nhớ cục bộ của chúng qua những bus riêng, nhỏ hơn. Các bộ xử lý cũng có thể truy cập những vùng nhớ của từng nút khác, tuy rằng thời gian truy cập này thay đổi tùy theo khoảng cách giữa các nút. Vì thế cơ chế có tên là truy cập bộ nhớ không đồng nhất NUMA. Trong hệ thống nhiều bộ vi xử lý thuận nhất, mỗi đơn vị xử lý trung tâm truy xuất vào bộ nhớ cục bộ, vấn đề còn lại là việc trao đổi thông tin giữa các bộ vi xử lý với nhau. Trong hệ thống nhiều bộ vi xử lý không thuận nhất các máy tính được xây dựng trên nền tảng của các bộ vi xử lý khác nhau.

Phần cứng đóng vai trò quan trọng trong hệ thống phân tán, nhưng sự hoạt động của hệ thống này lại do hệ điều hành quyết định. Hệ điều hành cung cấp các tính năng quản lý tài nguyên phần cứng, cho phép nhiều người dùng và nhiều ứng dụng chia sẻ phần cứng như: CPU, bộ nhớ, thiết bị ngoại vi, mạng và tất cả các loại dữ liệu. Hệ điều hành cũng đơn giản hóa sự phức tạp và đa dạng của phần cứng bằng cách tạo ra máy ảo, như vậy các ứng dụng có thể thực hiện dễ dàng hơn. Hệ điều hành cho các máy tính gồm hai loại: Hệ điều hành phân tán (DOS) và hệ điều hành mạng (NOS). Hệ điều hành phân tán quản lý tổng thể tất cả các máy tính thuận nhất trong hệ thống phân tán, hệ điều hành mạng thường dùng cho các hệ thống không đồng nhất, mỗi máy tính tạo ra các dịch vụ cung cấp cho các máy tính khác. Từ cuối những năm 1990, một số các dịch vụ do hệ điều hành cung cấp đã được cải tiến và gọi là phần mềm trung gian (Middleware).

Bảng 1.1 Hệ điều hành phân tán, hệ điều hành mạng và phần mềm trung gian.

Hệ thống	Mô tả	Mục tiêu
DOS	Hệ điều hành liên kết chặt dùng cho các hệ thống máy tính thuận nhất	Che giấu và quản lý các tài nguyên phần cứng
NOS	Hệ điều hành liên kết lỏng, dùng cho các máy tính không thuận nhất (mạng LAN và mạng WAN)	Cung cấp các dịch vụ cục bộ cho các máy tính khác truy nhập từ xa.
Middleware	Lớp phía trên của hệ điều hành mạng, cài đặt các dịch vụ mục đích chung.	Cung cấp tính trong suốt cho hệ thống phân tán

Có hai loại hệ điều hành phân tán: Hệ điều hành phân tán cho hệ thống nhiều bộ vi xử lý và hệ điều hành phân tán cho các máy tính cùng chủng loại. Ngoài khả năng quản lý nhiều bộ vi xử lý, các tính năng khác của các hệ điều hành phân tán cũng giống như hệ điều hành dành cho các hệ thống chạy trên một bộ vi xử lý.

Hệ điều hành chạy trên một bộ vi xử lý

Mục tiêu chính của loại hệ điều hành này là cho phép người sử dụng và các phần mềm ứng dụng truy nhập dễ dàng đến các tài nguyên dùng chung như CPU, bộ nhớ chính, đĩa và các thiết bị ngoại vi. Các phần mềm ứng dụng dùng chung tài nguyên của hệ thống nhưng vẫn đảm bảo tính độc lập cho từng ứng dụng, như vậy hệ điều hành cần phải có chính sách chia sẻ các tài nguyên dùng chung đó, điều này chỉ

có thể thực hiện bằng cách thiết lập cơ chế máy ảo, cung cấp khả năng xử lý đa nhiệm cho các ứng dụng. Ví dụ, để giải quyết vấn đề tương tranh trong hệ thống, các phần mềm ứng dụng không được phép truy nhập trực tiếp đến các tài nguyên mạng, việc truy nhập phải được thực hiện thông qua các hàm nguyên thủy do hệ điều hành cung cấp. Hệ điều hành cần phải nắm toàn bộ quyền kiểm soát việc sử dụng và chia sẻ tài nguyên phần cứng, do đó hầu hết các bộ vi xử lý hỗ trợ ít nhất hai chế độ:

- Chế độ lõi: Tất cả các chỉ thị được phép thực hiện và có thể truy nhập toàn bộ bộ nhớ và các thanh ghi trong thời gian thực hiện.
- Chế độ của người sử dụng: Hạn chế việc truy nhập thanh ghi và bộ nhớ (ví dụ chỉ được phép truy nhập vào vùng nhớ do hệ điều hành qui định, không truy nhập trực tiếp vào các thanh ghi)

Hệ điều hành cho nhiều bộ vi xử lý

Là các hệ điều hành dùng để điều khiển sự hoạt động của các hệ thống máy tính có nhiều bộ vi xử lý. Các hệ điều hành cho nhiều bộ vi xử lý gồm có 2 loại:

- **Đa xử lý đối xứng:** Trong hệ thống này vi xử lý nào cũng có thể chạy một loại tiểu trình bất kỳ, các vi xử lý giao tiếp với nhau thông qua một bộ nhớ dùng chung. Hệ đối xứng cung cấp một cơ chế chịu lỗi và khả năng cân bằng tải tối ưu hơn, vì các tiểu trình của hệ điều hành có thể chạy trên bất kỳ bộ vi xử lý nào nên nguy cơ xảy ra tình trạng tắc nghẽn ở CPU giám sát đáng kể. Vấn đề đồng bộ giữa các bộ vi xử lý được đặt lên hàng đầu khi thiết kế hệ điều hành cho hệ thống đối xứng.
- **Đa xử lý bất đối xứng:** Trong hệ thống này hệ điều hành dành ra một hoặc hai vi xử lý để sử dụng riêng, các vi xử lý còn lại dùng để điều khiển các chương trình của người sử dụng. Hệ bất đối xứng đơn giản hơn nhiều so với hệ đối xứng, nhưng trong hệ này nếu có một bộ vi xử lý trong các vi xử lý dành riêng cho hệ điều hành bị hỏng thì hệ thống có thể ngừng hoạt động.

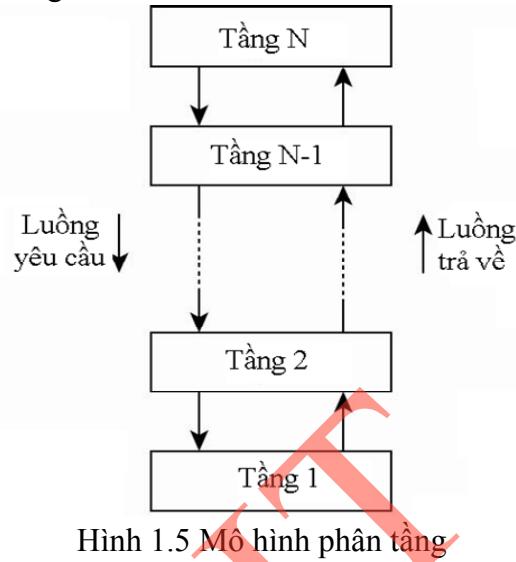
Hệ điều hành mạng là các hệ điều hành dùng để điều khiển sự hoạt động của mạng máy tính. Ngoài các chức năng cơ bản của một hệ điều hành, các hệ điều hành mạng còn phải thực hiện việc chia sẻ và bảo vệ tài nguyên của mạng.

1.4.2 Phân loại kiến trúc hệ thống phân tán

Hệ thống phân tán có thể được xây dựng theo kiến trúc phân tán dọc, phân tán ngang hoặc lai ghép hai loại trên. Trong kiến trúc phân tán dọc, 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. 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. Mô hình hệ thống phân tán là phương thức tổ chức phần mềm bên trong hệ thống, hiện nay có bốn mô hình đang được áp dụng phổ biến bao gồm: Mô hình phân tầng, mô hình dựa trên đối tượng, mô hình dựa trên sự kiện và mô hình dữ liệu tập trung.

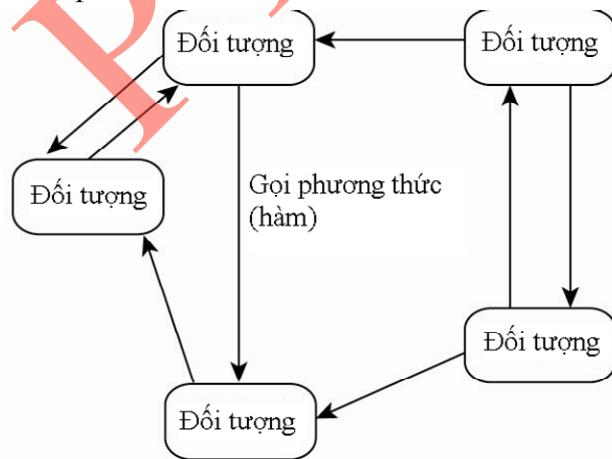
Hệ thống phân tán

Các thành phần trong mô hình phân tầng được tổ chức thành từng lớp có sự ràng buộc chặt chẽ, lớp trên 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.



Hình 1.5 Mô hình phân tầng

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 gọi bằng 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, đây là mô hình rất phù hợp với mô hình khách/chủ.



Hình 1.6 Mô hình đối tượng phân tán

Mô hình Khách/Chủ đang được áp dụng phổ biến trong các hệ thống phân tán, hiện nay mô hình Khách/Chủ đã phát triển theo hướng sử dụng các đối tượng phân tán. Mô hình Khách/Chủ được hiểu là hình thức trao đổi thông tin giữa các trình cung cấp dịch vụ (Máy chủ) và trình sử dụng dịch vụ (Máy khách). Trong mô hình này, máy khách yêu cầu các dịch vụ đã được cài đặt trên Máy chủ, Máy chủ xử lý yêu cầu

và trả về kết quả cho Máy khách. Cơ chế truyền tin sử dụng truyền tin báo giữa các tiến trình (IPC), nó cho phép cài đặt các tiến trình máy khách và máy chủ trên các máy tính khác nhau. Mô hình Khách/Chủ đóng vai trò quan trọng trong các hệ thống phân tán, nó có các đặc trưng sau:

- Máy khách và máy chủ là các mô đun chức năng với các giao diện xác định: Các chức năng thực hiện của máy khách hoặc máy chủ có thể gồm nhiều chức năng con, việc cài đặt các ứng dụng máy khách hoặc máy chủ không nhất thiết phải trên máy chủ mà có thể cài đặt tại bất cứ máy nào trên mạng.
- Quan hệ khách/chủ được thiết lập giữa hai mô đun khi máy khách đưa ra yêu cầu dịch vụ và được Máy chủ đáp lại: Khái niệm máy khách và máy chủ chỉ là tương đối, một mô đun có thể đóng vai trò máy chủ đối với mô đun này nhưng lại đóng vai trò máy khách đối với mô đun khác.
- Trao đổi thông tin giữa các mô đun được thực hiện thông qua cơ chế truyền tin báo: tất cả các yêu cầu của máy khách được tập hợp thành tin báo để chuyển đến máy chủ và ngược lại các kết quả trả về cũng được đặt trong các tin báo để chuyển đến máy khách.
- Trao đổi tin báo giữa máy khách và máy chủ thường được thực hiện theo cơ chế hỏi đáp.
- máy khách và máy chủ được cài đặt trên các máy tính khác nhau và được nối với nhau trên mạng: Về mặt lý thuyết, trên cùng một máy có thể cài đặt đồng thời mô đun Máy khách và mô đun Máy chủ, tuy nhiên trong thực tế thường các mô-đun này được cài đặt trên các máy khác nhau.

Với các đặc điểm trên, mô hình Khách/Chủ có thể cài đặt theo hai mức:

- Mức dịch vụ nền: tạo cơ sở để phát triển, hỗ trợ và quản lý các ứng dụng Khách/Chủ.
- Mức ứng dụng: là các phần mềm cung cấp các chức năng nghiệp vụ theo mô hình Khách/Chủ.

Mô hình Khách/Chủ cung cấp các phương tiện tích hợp các ứng dụng riêng với các nhu cầu xử lý nghiệp vụ chung để đảm bảo thống nhất việc xử lý thông tin trong toàn bộ hệ thống. Mô hình này có các ưu điểm sau:

Chia sẻ dữ liệu: trong mô hình Khách/Chủ, dữ liệu được lưu trên Máy chủ để sẵn sàng cung cấp cho tất cả những người sử dụng được quyền truy nhập. Việc sử dụng ngôn ngữ SQL để thống nhất các thao tác truy xuất dữ liệu đã hỗ trợ cho người sử dụng dễ dàng truy nhập dữ liệu.

Các dịch vụ tích hợp: Người sử dụng được quyền truy nhập đều có thể nhận được thông tin cần thiết từ các máy trạm và có thể xử lý các thông tin này theo nhu cầu sử dụng. Theo quyền truy nhập, Máy khách sử dụng chung các dịch vụ do Máy chủ cung cấp.

Chia sẻ tài nguyên giữa các hệ thống khác nhau: Có thể tạo các ứng dụng độc lập với hệ điều hành và thiết bị phần cứng, do đó các ứng dụng Máy khách đều có thể sử dụng các tài nguyên chung trên mạng: dữ liệu, dịch vụ....

Khả năng trao đổi và tương thích dữ liệu: Hầu hết các công cụ sử dụng để phát triển các sản phẩm theo mô hình Khách/Chủ đều dựa trên tiêu chuẩn của ngôn ngữ

SQL, do đó đảm bảo được tính tương thích và khả năng trao đổi dữ liệu giữa các tiến trình Máy khách và Máy chủ.

Không phụ thuộc thiết bị lưu trữ dữ liệu: Trên Máy chủ, dữ liệu có thể được lưu trữ trong các thiết bị khác nhau như đĩa từ hoặc băng từ, người sử dụng vẫn dùng tập các lệnh chung để truy xuất dữ liệu mà không phụ thuộc vào phương tiện lưu trữ.

Độc lập với vị trí xử lý dữ liệu: Việc truy xuất dữ liệu không phụ thuộc vào thiết bị phần cứng, hệ điều hành và vị trí lưu trữ dữ liệu.

Quản lý tập trung: Việc quản lý tập trung được thực hiện bằng cách sử dụng các công cụ giám sát và hỗ trợ từ trung tâm



Hình 1.7 Các thành phần cơ bản trong mô hình Khách/Chủ

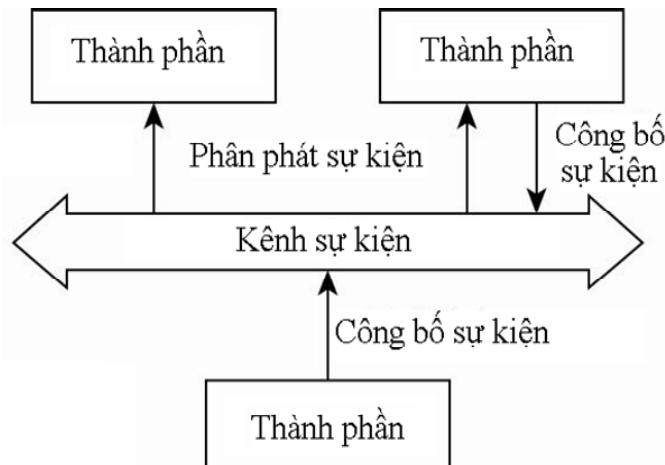
Hình 1.7 thể hiện các thành phần cơ bản trong mô hình Khách/Chủ, bao gồm các thành phần: tiến trình máy khách, máy chủ, phần mềm trung gian (MiddleWare), dịch vụ mạng, dịch vụ cục bộ (ví dụ: quản lý dữ liệu và quản lý tiến trình), hệ điều hành và thiết bị phần cứng. Tiến trình máy khách bao gồm các chức năng của lớp ứng dụng, nó thực hiện giao tiếp với người sử dụng và các chức năng ứng dụng cần thiết như hiển thị thông tin, tính toán các bảng tính... Tiến trình máy khách giao tiếp với các phần mềm trung gian qua giao diện lập trình ứng dụng (API) để gửi các yêu cầu đến máy chủ và nhận kết quả tính toán.

Tiến trình máy chủ thực hiện các chức năng lớp ứng dụng, nó cung cấp các dịch vụ cho máy khách với việc che dấu các thông tin riêng, đảm bảo cung cấp các dịch vụ xử lý lỗi và có thể thực hiện chức năng giám sát/điều phối. Phần mềm trung gian (MiddleWare) cung cấp nhiều tính năng khác nhau như thiết lập phiên làm việc giữa các tiến trình, bảo mật dữ liệu, nén/giải nén dữ liệu, xử lý lỗi. MiddleWare là môi trường trung gian kết nối tiến trình máy khách với tiến trình máy chủ, nó giao tiếp với các tiến trình qua giao diện API. Phần mềm trung gian trên máy khách thực hiện các chức năng cung cấp giao diện API, thiết lập liên kết với tiến trình trên máy chủ bằng cách gửi các lệnh thông qua giao diện mạng và phần mềm trung gian (của máy chủ).

Phần mềm trung gian trên máy chủ giám sát các yêu cầu từ phía máy khách và gọi các tiến trình máy chủ tương ứng, nó thực hiện các chức năng: Nhận các yêu cầu từ phía máy khách và chuyển các yêu cầu đó cho tiến trình máy chủ, kiểm tra bảo mật hệ thống, xử lý tranh chấp đồng thời nhận được nhiều yêu cầu từ phía máy khách,

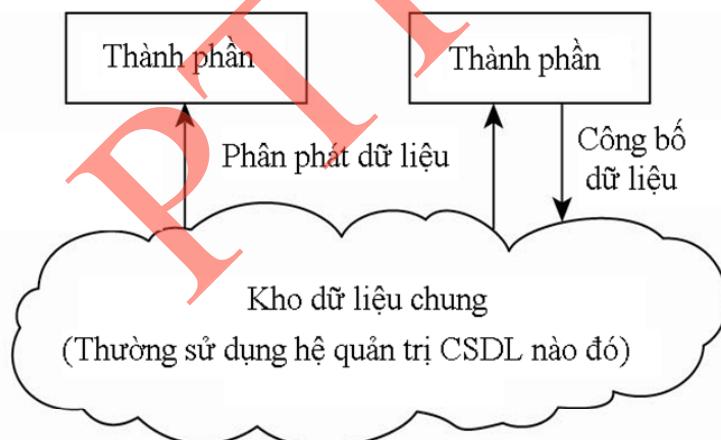
Hệ thống phân tán

nhận kết quả xử lý của tiến trình Máy chủ và chuyển đến máy khách, giám sát và xử lý lỗi.



Hình 1.8 Mô hình dựa trên sự kiện

Mô hình dựa trên sự kiện là mô hình mà các tiến trình trao đổi thông tin dựa trên việc phát tán sự kiện, các sự kiện 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.



Hình 2.9 Mô hình dữ liệu tập trung

Mô hình dữ liệu tập trung là mô hình trong đó các tiến trình trao đổi thông tin với nhau qua kho dữ liệu chung (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.

CHƯƠNG 2: VẤN ĐỀ VÀ GIẢI PHÁP TRONG HỆ THỐNG PHÂN TÁN

2.1 Truyền thông

Trao đổi thông tin giữa các tiến trình là trọng tâm của tất cả các hệ thống phân tán, do đó cần phải nghiên cứu kỹ lưỡng cách thức các tiến trình trao đổi thông tin với nhau. Thực chất trao đổi thông tin trong hệ thống phân tán là chuyển thông điệp do mạng máy tính đảm nhiệm, quá trình đó phức tạp hơn rất nhiều so với việc trao đổi thông tin trên một máy tính. Các hệ thống phân tán hiện đại bao gồm hàng triệu tiến trình trao đổi thông tin với nhau qua mạng Internet không tin cậy, nếu không thay đổi các phương thức truyền thông nguyên thủy thì sẽ rất khó phát triển các ứng dụng phân tán. Về bản chất, trao đổi thông tin giữa các tiến trình vẫn sử dụng các giao thức truyền tin truyền thống đã qui định trong từng lớp mạng. Các giao thức này được ứng dụng để xây dựng các mô hình truyền tin khác như Gọi thủ tục từ xa (RPC), gọi đối tượng từ xa (RMI), phần mềm trung gian hướng thông điệp (MOM). Mô hình truyền tin đầu tiên trong hệ thống phân tán là RPC, bản chất của nó là ẩn những thủ tục phức tạp trong việc truyền thông điệp và đó là cách lý tưởng trong các ứng dụng khách/chủ. Về sau, mô hình này được cải tiến dựa trên việc cài đặt các đối tượng phân tán.

Một số ứng dụng phân tán sử dụng phương thức truyền thông điệp, tính trong suốt của phương thức này khá thấp, do đó ~~nên chuyển~~ sang mô hình hàng đợi (tương tự như thư điện tử) hoặc sử dụng lớp phần mềm trung gian hướng thông điệp (MOM) để bảo đảm việc phân loại các thông điệp. Các dữ liệu đa phương tiện như âm thanh, hình ảnh ... đòi hỏi việc truyền tin ~~cần~~ phải đáp ứng yêu tố thời gian và tính liên tục, do đó mô hình truyền thông điệp ~~đường~~ như chưa đáp ứng được yêu cầu, khi đó phải sử dụng phương pháp truyền theo luồng (stream). Vấn đề cuối cùng cần nghiên cứu là việc sử dụng phương thức ~~truyền tin điểm~~ với ~~điểm~~-điểm hay điểm-nhóm (multicast), nghĩa là cùng một nội dung sẽ gửi một hay nhiều tiến trình khác.

2.1.1 Cơ sở truyền thông

Trước khi thảo luận về truyền thông trong các hệ thống phân tán chúng ta sẽ nhắc lại một số kiến thức cơ bản về các giao thức mạng và sau đó sẽ đề cập đến một số cách tiếp cận áp dụng chúng để giải quyết vấn đề truyền thông trong các hệ thống phân tán.

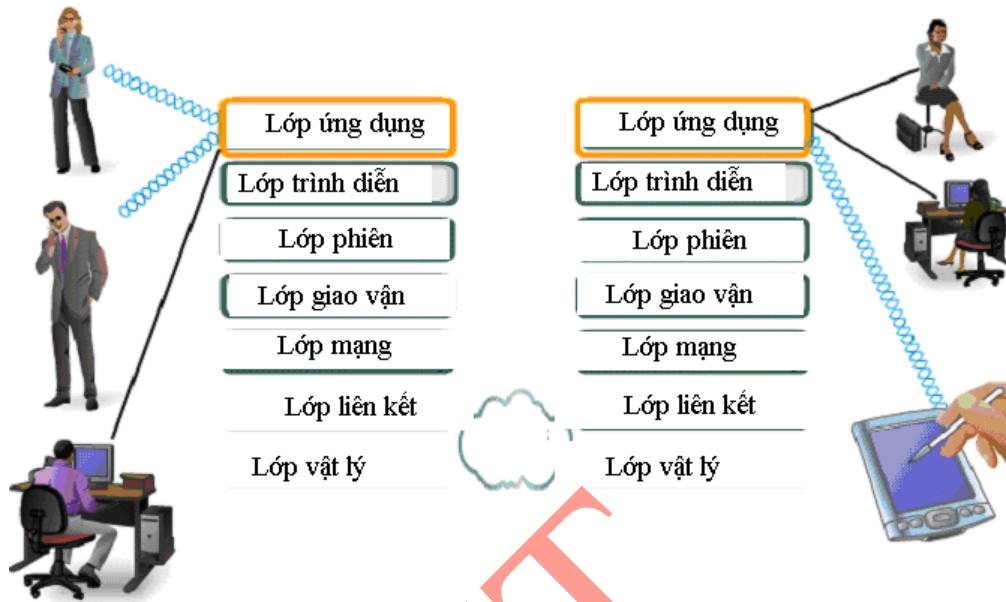
2.1.1.1 Giao thức mạng

Các 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. Khi một tiến trình A muốn trao đổi thông tin với tiến trình B, nó 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.

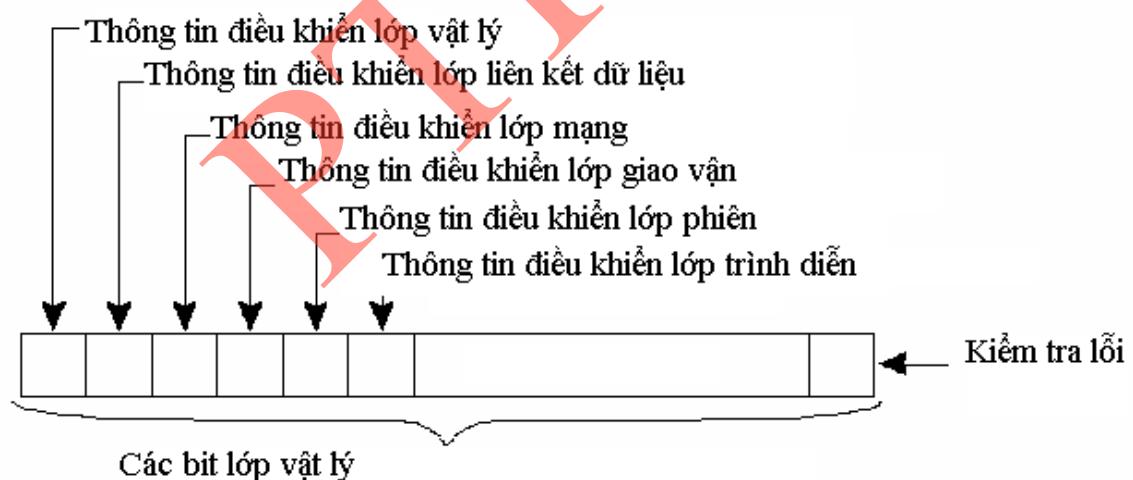
Mô hình OSI được phân thành 7 lớp (xem hình 2.1), mỗi lớp là một tập các giao thức qui định khuôn dạng dữ liệu và các thủ tục xử lý (cách gửi/nhận, cách xử lý lỗi). Có hai loại giao thức:

Hệ thống phân tán

- Giao thức có liên kết: Cần phải thiết lập liên kết trước khi truyền số liệu, sau khi truyền xong thì phải hủy bỏ liên kết.
- Giao thức không liên kết: Không cần phải thiết lập liên kết khi truyền số liệu



Hình 2.1 Mô hình liên kết hệ thống mở (OSI)



Hình 2.2: Đóng gói dữ liệu tại các tầng mô hình OSI

Mỗi lớp trong mô hình OSI thực hiện một số chức năng nhất định:

- Lớp ứng dụng: Cung cấp giao diện phục vụ cho người sử dụng và các ứng dụng khác
- Lớp trình diễn: Thực hiện mã hóa/giải mã, nén/giải nén và bảo mật dữ liệu.
- Lớp phiên: Tạo ra các phiên làm việc.
- Lớp giao vận: Tạo liên kết giữa đầu cuối với đầu cuối, điều khiển tốc độ truyền dữ liệu, xử lý lỗi truyền tin.

Hệ thống phân tán

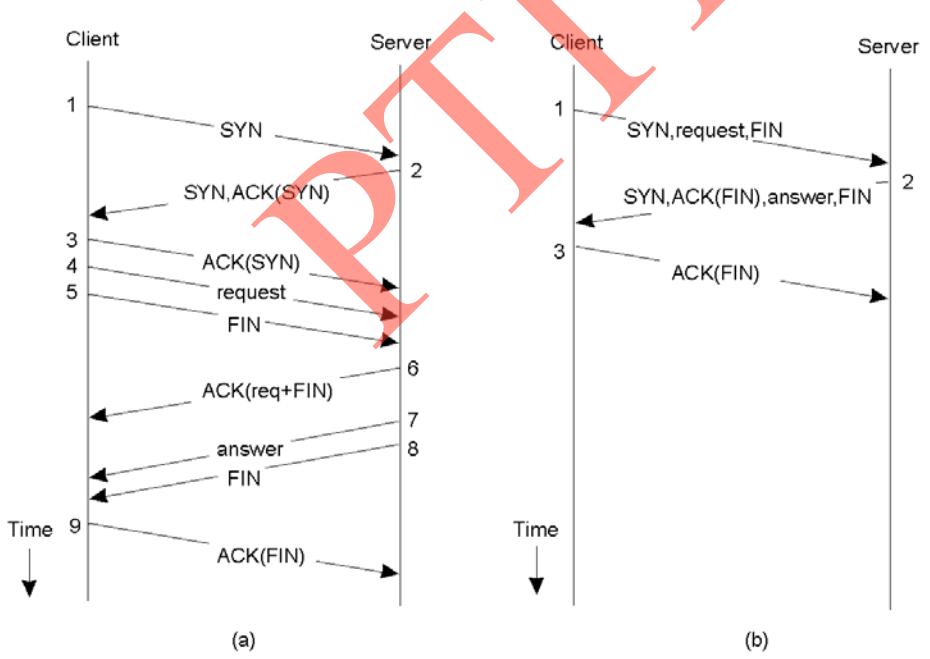
- Lớp mạng: Quản lý địa chỉ logic của các đối tượng tham gia vào mạng, tìm đường đi tốt nhất cho mỗi gói tin.
- Lớp liên kết: Thiết lập liên kết giữa hai thiết bị vật lý kề cạnh nhau
- Lớp vật lý: Biến đổi các bit dữ liệu thành các tín hiệu phù hợp với môi trường truyền dẫn và thực hiện thu phát các tín hiệu đó.

Khi một tiến trình A trên máy tính thứ nhất muốn gửi thông tin cho tiến trình B trên máy tính thứ hai, nó thêm tạo dữ liệu tại lớp ứng dụng và lần lượt chuyển đến các lớp dưới trên máy tính đó. Khi đi qua mỗi lớp, thông tin điều khiển được thêm vào dữ liệu, quá trình đó gọi là bao đóng dữ liệu (hình 2.2).

Các giao thức mức thấp

Các giao thức mức thấp hàm ý chỉ các giao thức tại lớp vật lý, liên kết dữ liệu và lớp mạng. Lớp vật lý liên quan tới việc chuyển các bit dữ liệu, qui định các tiêu chuẩn về điện, cơ và các giao diện kết nối mạng, phương pháp truyền các bit. Lớp liên kết dữ liệu tập hợp các bit thành từng nhóm (gọi là khung dữ liệu), phát hiện và sửa lỗi khi truyền các khung dữ liệu đó. Trong mạng diện rộng, việc trao đổi thông tin giữa các máy tính phải chuyển qua nhiều thiết bị định tuyến. Nhiệm vụ chính của các thiết bị này là duy trì bảng định tuyến và tìm đường đi tốt nhất cho mỗi gói tin. Hiện nay giao thức mạng đang được áp dụng phổ biến nhất là giao thức IP.

Các giao thức lớp giao vận



Hình 2.3 Qui trình truyền số liệu có liên kết

Lớp giao vận là lớp cuối cùng trong ngăn xếp giao thức cơ sở (người phát triển phần mềm sử dụng tập các giao thức này để phát triển các ứng dụng mạng). Chức năng cơ bản của lớp giao vận là quản lý việc trao đổi thông tin giữa hai thiết bị đầu cuối của người sử dụng. Kỹ thuật truyền số liệu có hai loại: truyền số liệu có liên kết và không

liên kết. Đối với truyền số liệu có liên kết, lớp giao vận thực hiện thêm các chức năng phát hiện, sửa lỗi và điều khiển tốc độ truyền dữ liệu giữa các thiết bị đầu cuối của người sử dụng. Trong phương pháp truyền dữ liệu có liên kết, trạm phát và trạm nhận cần phải thiết lập liên kết trước khi truyền số liệu, sau khi truyền xong thì phải hủy bỏ liên kết.

Các giao thức mức cao

Các giao thức mức cao nằm trên lớp giao vận, OSI khuyến nghị 3 lớp: Lớp phiên, lớp trình diễn và lớp ứng dụng. Ngoài ra, tùy theo yêu cầu phát triển hệ thống người ta có thể thêm một số các giao thức khác nhằm đơn giản hóa quá trình phát triển các sản phẩm phần mềm mạng, các giao thức đó được gộp chung vào nhóm phần mềm trung gian (Middleware). Lớp phiên quản lý cuộc hội thoại giữa các máy tính trên mạng. Tại lớp này người ta thiết kết các điểm kiểm tra nhằm hạn chế việc phải truyền lại toàn bộ dữ liệu khi xảy ra sự cố mất dữ liệu trên mạng. Lớp trình diễn thực hiện các nhiệm vụ mã hóa/giải mã dữ liệu nhằm thống nhất cách thể hiện các loại dữ liệu khác nhau của người sử dụng (dùng bảng mã ASCII), nén/giải nén dữ liệu mã hóa/giải mã bảo mật dữ liệu. Lớp ứng dụng bao gồm các giao thức phục vụ trực tiếp cho các dịch vụ của người sử dụng như: thư điện tử, truyền file, truy nhập trang thông tin điện tử....

Phần mềm trung gian

Về mặt logic, phần mềm trung gian thường nằm tại lớp ứng dụng, nó chứa nhiều giao thức đa năng để bảo tinh độc lập với các ứng dụng riêng, chúng thường được thể hiện dưới dạng các dịch vụ. Tuy theo yêu cầu sử dụng và khả năng phát triển, có thể xây dựng hệ thống phân tán theo các mức độ: truyền tập tin, xử lý khách/chủ thuần túy (Máy trạm/Máy chủ), xử lý hình ngang hàng (Peer-To-Peer). Phương pháp truyền tập tin là mức đơn giản nhất trong các hệ thống phân tán, thông tin cần trao đổi giữa các đối tượng trong hệ thống được lưu dưới dạng tập tin, các máy tính phải cùng sử dụng một giao thức truyền tập tin.

Phần mềm trung gian đơn giản hóa sự phức tạp trong việc truyền dữ liệu trong mạng, nhờ có phần mềm này mà việc gọi các thủ tục từ xa sẽ được thể hiện tương tự như gọi các thủ tục trên máy cục bộ. Đối chiếu với mô hình 7 lớp OSI, phần mềm trung gian thể hiện các tính năng của nó trong lớp trình diễn và lớp phiên. Hiện nay có nhiều kiến trúc khác nhau dùng để thể hiện phần mềm trung gian như: Gọi thủ tục từ xa (RPC), kiến trúc môi trường yêu cầu đối tượng chung (CORBA), mô hình đối tượng thành phần phân tán (DCOM) và gọi thao tác từ xa ứng dụng trong Java (RMI). Giải pháp RMI áp dụng riêng cho ngôn ngữ lập trình Java, nó sử dụng nhiều đặc tính của phương pháp gọi thủ tục từ xa RPC và kiến trúc CORBA. Tuy nhiên khi kiến trúc CORBA được đưa vào lõi của ngôn ngữ Java thì kiến trúc này sẽ dần thay thế RMI. Mô hình DCOM chỉ được ứng dụng trong các hệ thống cài đặt hệ điều hành của Microsoft và có hiều hạn chế trong việc thiết kế hệ thống. Mức xử lý ngang hàng là mức độ cao hơn trong mô hình khách/chủ, các tiến trình tương tác có thể là máy khách, máy chủ hoặc đồng thời là máy khách và máy chủ.

2.1.1.2 Phân loại truyền thông

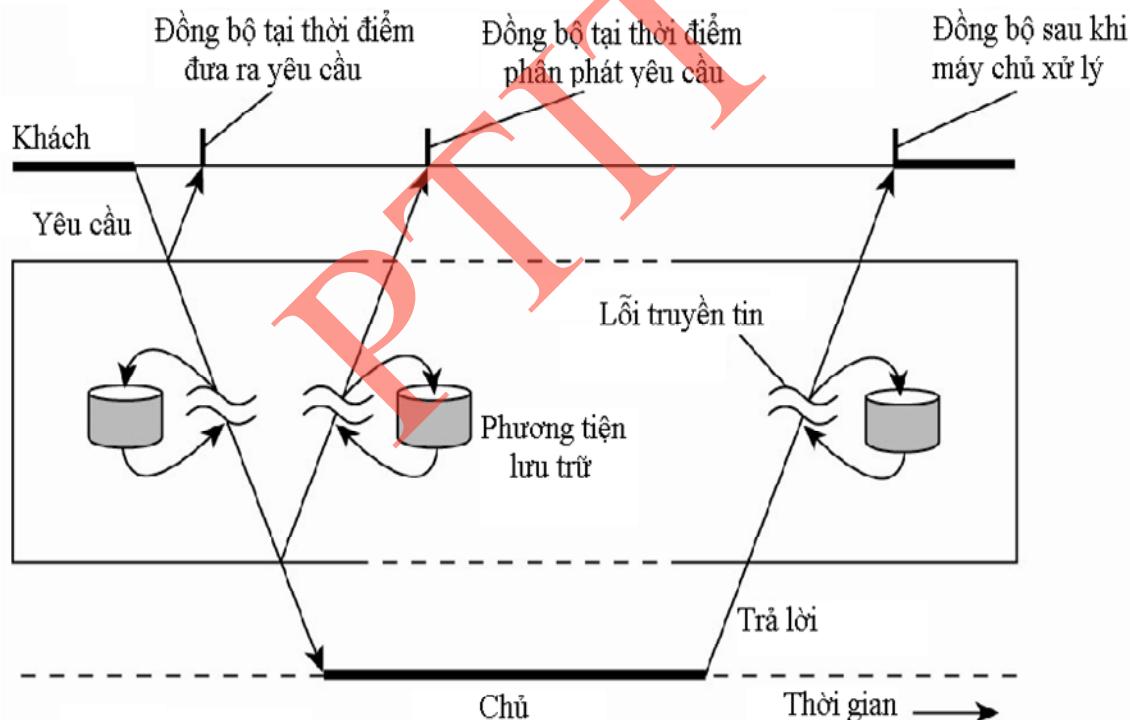
Để hiểu về các loại truyền thông mà lớp trung gian cung cấp cho các ứng dụng, chúng ta coi nó như một dịch vụ phụ trợ trong mô hình tính toán khách/chủ. Ví dụ hệ thống thư điện tử, về nguyên tắc thì lõi của hệ thống này là dịch vụ truyền thông trung gian, trên mỗi máy của người dùng cài đặt phần mềm cho phép biên soạn, gửi và nhận

Hệ thống phân tán

thư điện tử. Người dùng biên soạn thư, gửi lên hệ thống phân phát thư điện tử và chờ đợi kết quả phân phát thư đó đến người nhận. Tương tự như vậy, người nhận kết nối đến hệ thống thư điện tử, kiểm tra xem có thư của mình hay không, nếu có thì hệ thống thư điện tử sẽ chuyển các bức thư đó tới máy của người dùng.

Hệ thống thư điện tử là một ví dụ điển hình về phương pháp truyền thông bền bỉ (persistent), các thông điệp của người dùng được lưu trữ trong hệ thống cho đến khi chuyển thành công đến người nhận, bên gửi và bên nhận hoạt động hoàn toàn độc lập với nhau. Ngược lại, phương pháp truyền thông nhất thời (transient) chỉ lưu giữ thông điệp trong thời gian gửi và nhận, nghĩa là bên gửi và bên nhận phụ thuộc lẫn nhau, nếu bên nhận không hoạt động thì các thông điệp sẽ bị hủy bỏ.

Truyền thông cũng có thể được thực hiện dưới hình thức đồng bộ hoặc không đồng bộ. Trong phương thức truyền thông đồng bộ, bên gửi sẽ bị phong tỏa cho đến khi biết chắc chắn yêu cầu của mình đã được bên nhận xử lý. Phương pháp này đánh dấu ba thời điểm: Thời điểm thứ nhất bên gửi sẽ bị phong tỏa cho đến khi hệ thống trung gian tiếp nhận xong yêu cầu, thời điểm thứ hai hệ thống trung gian thông báo đã chuyển yêu cầu cho bên nhận và thời điểm thứ ba bên gửi sẽ tiếp nhận kết quả bên nhận xử lý. Ngược lại, truyền thông không đồng bộ cho phép bên gửi tiếp tục thực hiện công việc của mình sau khi đã gửi thông điệp đến hệ thống trung gian.



Hình 2.4 Nguyên lý truyền thông sử dụng thành phần trung gian

Trong thực tế người ta thường kết hợp hai loại truyền thông trên để trao đổi thông tin giữa các tiền trinh, phương pháp truyền bền bỉ và đồng bộ thường được áp dụng trong các hệ thống truyền thông diệp trong khi đó phương pháp ngắn và đồng bộ lại được áp dụng rộng rãi để thực hiện gọi thủ tục từ xa. Bên cạnh tính bền bỉ và tính đồng bộ người ta còn phân biệt tính rời rạc hay liên tục của truyền thông, những hệ thống mà mỗi thông điệp được truyền đi là những đơn vị dữ liệu độc lập sẽ được xếp

vào nhóm rác, nếu các thông điệp được truyền lần lượt và liên tục gọi là phương pháp truyền tin theo luồng.

2.1.2 Gọi thủ tục xa

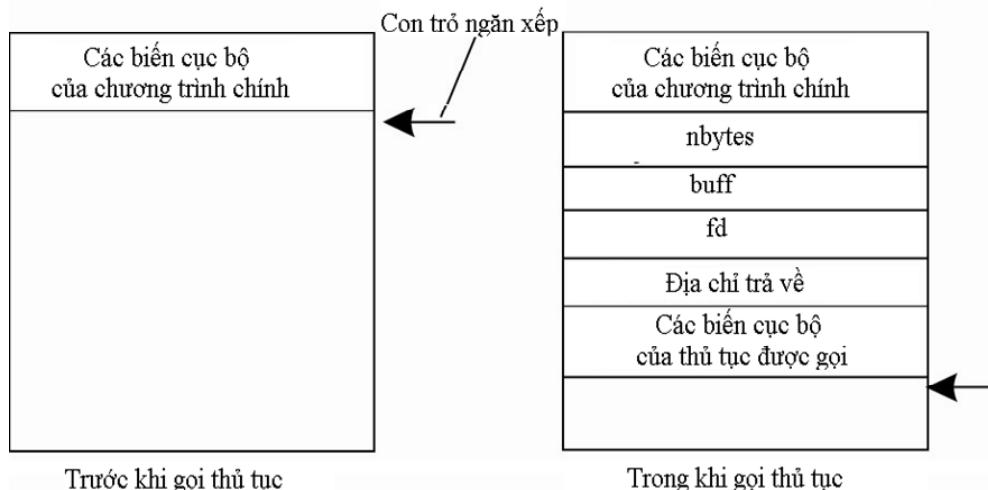
Nhiều hệ thống phân tán sử dụng phương pháp trao đổi trực tiếp các thông điệp giữa các tiến trình, quá trình này được thực hiện tường minh do đó không đáp ứng yêu cầu tính trong suốt truyền tin, các thủ tục gọi và nhận dữ liệu hoàn toàn không quan tâm đến quá trình trao đổi thông tin. Các khuyến nghị về phương pháp gọi thủ tục từ xa (RPC) đã được đưa ra từ năm 1975, nhưng mãi tới năm 1984 Birrell và Nelson mới đề xuất giải pháp hoàn toàn mới để thực hiện trao đổi thông tin bằng phương pháp này. Ý tưởng của phương pháp vẫn dựa vào qui trình gọi thủ tục, tuy nhiên điểm khác biệt nằm ở chỗ thủ tục đó được cài đặt trên một máy tính khác, người lập trình chỉ cần chuyển các giá trị vào các tham số của thủ tục. Về nguyên tắc, phương pháp này khá đơn giản cho việc cài đặt, tuy nhiên trong thực tế này sinh khá nhiều vấn đề như: 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. Mặc dù vậy, phương pháp gọi thủ tục từ xa vẫn là phương pháp được áp dụng phổ biến nhất trong các hệ thống phân tán.

2.1.2.1 Cơ chế hoạt động của phương pháp 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: Các ứng dụng khách kết nối với máy chủ và sử dụng các dịch vụ do máy chủ cung cấp. Các bước gọi thủ tục trên máy chủ được thực hiện tương tự như gọi thủ tục trên máy cục bộ, may khác chuyển các tham số đầu vào khi gọi thủ tục và 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 trạm. Để hiểu về phương pháp gọi thủ tục từ xa, trước hết chúng ta cần phải nhắc lại qui trình thực hiện khi gọi một thủ tục truyền thống trên một máy tính. Giả sử có một thủ tục đọc tập tin đơn giản sau:

```
count = read(fd, buff, nbytes);
```

trong đó *fd* là con trỏ tập tin, *buff* là vùng đệm, *nbytes* là số lượng byte cần đọc.

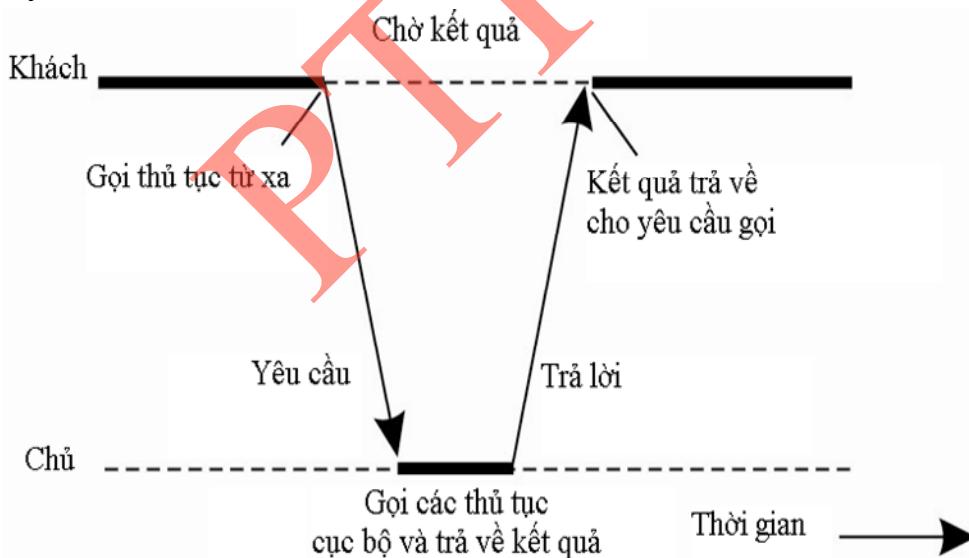


Hình 2.5 Gọi thủ tục từ xa theo phương pháp truyền thống

Khi chương trình gọi thủ tục, nó tạo ra một ngăn xếp dành cho thủ tục đó. Để thực thi lời gọi thủ tục, nó truyền các tham số của thủ tục vào ngăn xếp theo thứ tự ngược (tham số đầu tiên sẽ được chuyển cuối cùng). Sau khi 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, chương trình gọi sẽ loại bỏ tham số ra khỏi ngăn xếp, trả ngăn xếp về trạng thái như trước khi gọi thủ tục. Việc truyền tham số có thể được thực hiện bằng một trong ba phương pháp: Truyền giá trị, truyền con trỏ (pointer) và truyền tham chiếu (reference).

Con trỏ đơn giản là địa chỉ của một đối tượng trong bộ nhớ. Thông thường, các đối tượng có thể được truy xuất trong hai cách: trực tiếp bởi tên đại diện hoặc gián tiếp thông qua con trỏ. Các biến con trỏ được định nghĩa trỏ tới các đối tượng của một kiểu cụ thể sao cho khi con trỏ hủy thì vùng nhớ mà đối tượng chiếm giữ được thu hồi. Các con trỏ thường được dùng cho việc tạo ra các đối tượng động trong thời gian thực thi chương trình. Không giống như các đối tượng bình thường (tồn tại và cục bộ) được cấp phát lưu trữ trên ngăn xếp trong thời gian chạy (runtime stack), một đối tượng động được cấp phát vùng nhớ từ vùng lưu trữ khác được gọi là heap.

Tham chiếu cung cấp một tên tượng trưng khác gọi là biệt hiệu (alias) cho một đối tượng, truy xuất một đối tượng thông qua một tham chiếu giống như là truy xuất thông qua tên gốc của nó. Tham chiếu nâng cao tính hữu dụng của các con trỏ và sự tiện lợi của việc truy xuất trực tiếp các đối tượng, chúng được sử dụng để hỗ trợ các kiểu gọi thông qua tham chiếu của các tham số hàm đặc biệt khi các đối tượng lớn được truyền tới hàm.



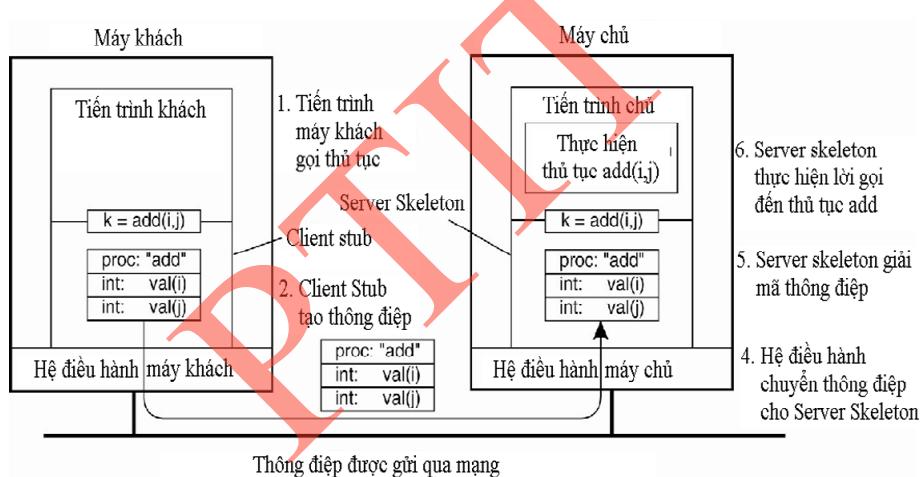
Hình 2.6 Nguyên lý gọi thủ tục từ xa

Ý tưởng phương pháp gọi thủ tục từ xa là che giấu quá trình thực hiện thủ tục trên máy tính khác (đảm bảo tính trong suốt). Điều này được thực hiện bằng cách che giấu quá trình trao đổi thông tin giữa các máy tính. Khi thực hiện gọi thủ tục từ xa, một thành phần trên máy khách (gọi là Stub) sẽ đóng gói các tham số vào thông điệp và yêu cầu thông điệp đó phải được chuyển đến máy chủ. Tại máy chủ, một thành

phản tương ứng với Stub (gọi là Skeleton) sẽ giải mã thông điệp đã nhận được và thực thi các mã lệnh như phương pháp gọi thủ tục truyền thống, sau đó trả về lại được đóng gói thành thông điệp và trả về cho Stub. Stub trên máy khách sẽ giải mã thông điệp và trả về cho chương trình gọi các giá trị hoặc các tham số theo yêu cầu. Quá trình xử lý giữa Stub và Skeleton hoàn toàn trong suốt đối với lời gọi thủ tục.

Để sử dụng tính năng gọi thủ tục từ xa, một chương trình trên máy chủ phải cung cấp các dịch vụ mô tả trong ngôn ngữ RPC. Mỗi máy chủ được gán tên và số chương trình, tất cả các dịch vụ được khai báo trong danh sách với đầy đủ các tham số thể hiện dịch vụ. Với nguyên tắc này, ngôn ngữ RPC cho phép thực hiện các kiểu dữ liệu đơn giản cũng như các dữ liệu phức tạp như: struct, enum... Qui trình thực hiện bao gồm các bước sau:

- Máy khách sử dụng tính năng gọi thủ tục cục bộ trong Stub.
- Stub trên máy khách gửi các tham số đến máy chủ bằng cách gửi yêu cầu RPC.
- Yêu cầu của Stub được Skeleton trên máy chủ phân tích.
- Thực hiện thủ tục đã được phân tích trên máy chủ.
- Máy chủ trả về kết quả thực hiện cho máy khách



Hình 2.7 Các bước thực hiện trong gọi thủ tục từ xa

Phương pháp gọi thủ tục từ xa thể hiện quan điểm tách biệt giữa giao diện và phần cài đặt, xuất phát từ việc khai báo giao diện, phần mềm trung gian tạo các mã lệnh hỗ trợ cho việc xử lý phân tán bằng cách thể hiện các ứng dụng dưới dạng ngữ nghĩa truyền thống trên máy cục bộ, tuy nhiên nó phải đảm bảo nhiều nhiệm vụ phức tạp như: truyền dữ liệu, đồng bộ tiến trình. Quá trình thực hiện gọi thủ tục từ xa được thực hiện qua mười bốn 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.

2.1.2.2 Ván đè 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 trả về 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**).

<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>5</td></tr> <tr><td>7</td><td>6</td><td>5</td><td>4</td></tr> <tr><td>L</td><td>L</td><td>I</td><td>J</td></tr> </table>	3	2	1	0	0	0	0	5	7	6	5	4	L	L	I	J	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>5</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>J</td><td>I</td><td>L</td><td>L</td></tr> </table>	0	1	2	3	5	0	0	0	4	5	6	7	J	I	L	L	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>5</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>L</td><td>L</td><td>I</td><td>J</td></tr> </table>	0	1	2	3	0	0	0	5	4	5	6	7	L	L	I	J
3	2	1	0																																															
0	0	0	5																																															
7	6	5	4																																															
L	L	I	J																																															
0	1	2	3																																															
5	0	0	0																																															
4	5	6	7																																															
J	I	L	L																																															
0	1	2	3																																															
0	0	0	5																																															
4	5	6	7																																															
L	L	I	J																																															
Thông điệp ban đầu trên máy Pentium	Thông điệp nhận được trên máy SPARC	Thông điệp sau khi đảo ngược																																																

Hình 2.8 Ván đè truyền tham số trong các hệ thống không đồng nhất

Hình 2.8 minh họa ví dụ thủ tục từ xa gồm 2 tham số (5, “JILL”), các byte (thực chất là các bit) dữ liệu lần lượt được chuyển từ máy tính của hãng Intel sang máy tính của hãng Sun, khi đến máy Sun giá trị của các tham số theo thứ tự ngược và máy Sun sẽ hiểu số 5 là 5×2^{24} , như vậy chỉ cần đảo ngược thứ tự là sẽ nhận được kết quả ban đầu. Tuy nhiên, kiểu số nguyên mới cần đảo vị trí các byte nhưng dữ liệu kiểu xâu ký tự lại không cần thiết, do đó phải bỏ xung 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. Tuy nhiên, 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.

Các biến cục bộ của thủ tục foobar(...)	
	x
	y
	5
	z[0]
	z[1]
	z[2]
	z[3]
	z[4]

foobar(char x; float y; int z[5])
 {

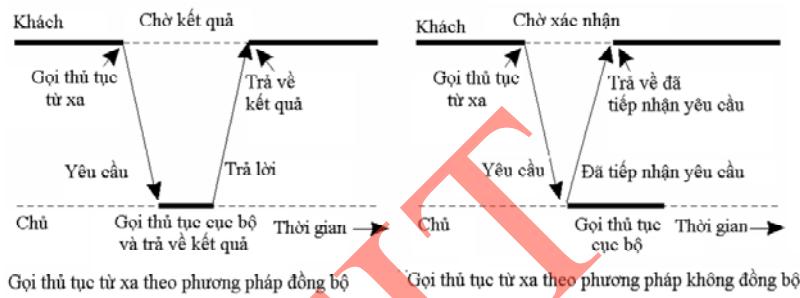
 }

Hình 2.9 Chuyển thủ tục thành thông điệp trên Stub

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.

2.1.2.3 Gọi thủ tục từ xa bằng phương pháp không đồng bộ

Giống như phương pháp gọi thủ tục thông thường, khi 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.



Hình 2.10 Gọi thủ tục từ xa bằng phương pháp đồng bộ (a) và không đồng bộ (b)



Hình 2.11 Tương tác gọi thủ tục từ xa bằng phương pháp không đồng bộ

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, như vậy sẽ rút ngắn thời gian phong tỏa hệ thống.

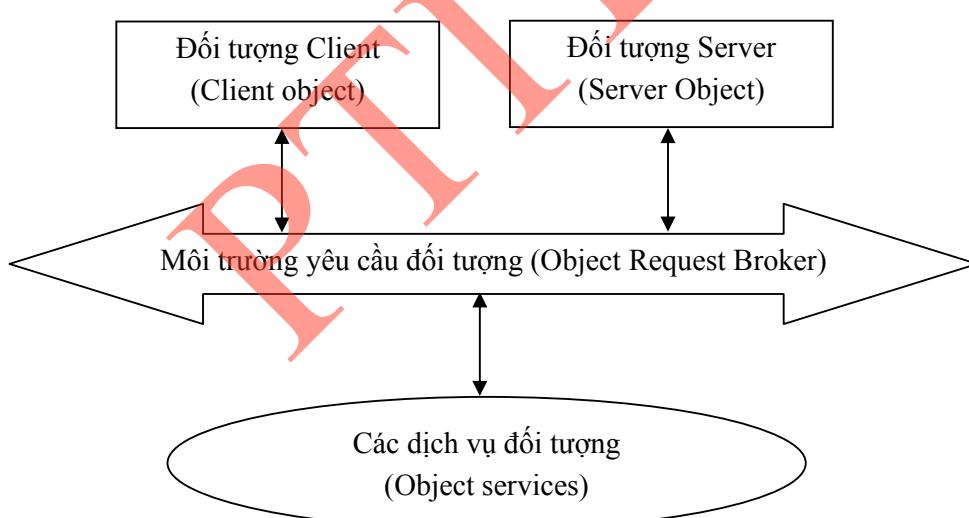
2.1.2.4 Mô hình đối tượng phân tán

Hiện nay việc xây dựng ứng dụng mạng dựa trên hai quan điểm: Hướng đối tượng và hệ thống phân tán. Mỗi ứng dụng phân tán là tập các đối tượng (giao diện người sử dụng, cơ sở dữ liệu, các mô đun của ứng dụng...), mỗi đối tượng có các thuộc tính và

các hàm riêng, tương tác giữa các đối tượng được thực hiện thông qua việc trao đổi tin báo để gọi các hàm thích hợp.

Kỹ thuật lập trình hướng đối tượng đã chứng tỏ được tính ưu việt khi phát triển các ứng dụng phần mềm chạy trên một máy tính. Đặc tính cơ bản của lập trình hướng đối tượng là việc che giấu xử lý bên trong, do đó những thay đổi bên trong đối tượng sẽ không ảnh hưởng đến các đối tượng khác sử dụng. Từ ưu điểm của kỹ thuật lập trình hướng đối tượng, trong các hệ thống phân tán người ta đã phát triển phương pháp gọi thủ tục từ xa thành phương pháp gọi đối tượng từ xa. Tiêu biểu trong kỹ thuật này có thể kể đến Java RMI, Microsoft DCOM, CORBA. Mô hình các đối tượng phân tán được thể hiện trên hình 2.12, trong đó:

- **Đối tượng:** là dữ liệu được đóng gói trong chương trình, nó có đầy đủ các tính chất của một đối tượng theo quan điểm lập trình hướng đối tượng như: tính thừa kế, tính đa hình.... Các đối tượng có thể đóng vai trò Máy trạm, Máy chủ hoặc cả hai.
- **Môi trường yêu cầu đối tượng (ORB):** là môi trường cho phép các đối tượng tìm thấy nhau theo phương pháp động và giao tiếp với nhau qua mạng, đây là xương sống của các hệ thống hướng đối tượng phân tán.
- **Các dịch vụ đối tượng:** là các dịch vụ cho phép người sử dụng thao tác và xử lý các đối tượng.



Hình 2.12 Mô hình các đối tượng phân tán

Phần mềm trung gian đóng vai trò cung cấp các công cụ hỗ trợ việc xác định các đối tượng từ xa và tạo môi trường thuận tiện để đối tượng giao tiếp với nhau.

Xu thế chuẩn hóa phần mềm trung gian

Việc thiết lập các tiêu chuẩn cho phần mềm trung gian nhằm mục đích cung cấp khả năng tương thích và tính mềm dẻo của các ứng dụng Máy trạm/Máy chủ, trong đó tập trung vào việc: Chuẩn hoá giao diện lập trình API và chuẩn hoá giao thức trao đổi thông tin. Theo hai tiêu chí trên, phần mềm trung gian được chia thành bốn loại sau:

- Phần mềm trung gian mở hoàn toàn: sử dụng giao diện API chung và giao thức trao đổi chung giữa Máy trạm và Máy chủ, nói chung các sản phẩm loại này phải đảm bảo khả năng giao tiếp giữa Máy trạm và Máy chủ mà không phụ thuộc giao thức trao đổi thông tin giữa các cổng mạng. Sản phẩm loại này bao gồm DCE RPC (gọi thủ tục từ xa trong môi trường tính toán phân tán), CORBA, OpenDoc.
- Phần mềm trung gian với giao diện API mở: Cho phép Máy trạm giao tiếp với Máy chủ sử dụng sản phẩm của các hãng khác nhau với điều kiện giao thức sử dụng trao đổi thông tin giống nhau, ví dụ sản phẩm ODBC của Microsoft.
- Phần mềm trung gian với giao thức trao đổi thông tin mở: sử dụng giao thức trao đổi thông tin chung, tuy nhiên có chuyển đổi giao diện API khi Máy trạm và Máy chủ sử dụng phần mềm trung gian của các nhà cung cấp khác nhau (ví dụ kiến trúc cơ sở dữ liệu quan hệ phân tán DRDA của IMB).
- Phần mềm trung gian riêng: các ứng dụng Máy trạm, Máy chủ chỉ giao tiếp với nhau khi sử dụng phần mềm trung gian của cùng hãng (ví dụ ActiveX/OLE).

Đối tượng phân tán

Một đặc tính quan trọng của đối tượng là tính bao đóng các thuộc tính (hàng, biến số) và các thao tác xử lý các thuộc tính đó, việc thao tác với thuộc tính trong mỗi đối tượng đều phải được thực hiện thông qua các hàm giao diện. Trong các hệ thống phân tán, đối tượng không đơn thuần nằm trên một máy tính, đa phần chúng được cài đặt trên các máy tính khác nhau.

Khi máy trạm nhúng vào một đối tượng phân tán, một thành phần giao diện của đối tượng (còn gọi là proxy) sẽ được nạp vào không gian địa chỉ của máy trạm. Proxy thực hiện chức năng tương tự như máy trạm stub trong phương pháp gọi thủ tục từ xa, nó đảm nhiệm chức năng chuyển đổi lời gọi các hàm giao diện thành dạng thông điệp để hệ điều hành chuyển đến máy chủ và ngược lại chuyển đổi thông điệp từ máy chủ thành kết quả trả về của hàm giao diện. Đối tượng thực nằm ở phía máy chủ, nó có giao diện giống như trên máy trạm và đồng thời một thành phần gọi là skeleton có nhiệm vụ chuyển đổi thông điệp từ phía máy trạm thành lời gọi hàm tương ứng với yêu cầu của máy trạm và ngược lại chuyển kết quả thực hiện của hàm thành thông điệp để hệ điều hành chuyển tới máy trạm. Các thuộc tính trong đối tượng phân tán có thể được tổ chức trên các máy tính khác nhau, tuy nhiên trong hầu hết các hệ thống phân tán, các thuộc tính của đối tượng phân tán thường được đặt trên một máy tính, đó là đối tượng từ xa.

Biên dịch đối với các đối tượng phân tán

Các đối tượng trong hệ thống phân tán xuất hiện dưới nhiều hình thức, một trong những hình thức phổ biến nhất là liên quan trực tiếp tới các đối tượng trong các ngôn ngữ lập trình hướng đối tượng như Java, C++, đó là các ngôn ngữ biên dịch, việc sử dụng các ngôn ngữ lập trình này sẽ tạo điều kiện dễ dàng cho việc phát triển các ứng dụng phân tán. Nhược điểm cơ bản của các đối tượng biên dịch là sự phụ thuộc vào ngôn ngữ lập trình, để khắc phục nhược điểm này cần phải xây dựng các đối tượng phân tán một cách tường minh, khi đó có thể xây dựng các ứng dụng phần mềm từ nhiều ngôn ngữ lập trình khác nhau.

Việc cài đặt bên trong các đối tượng phân tán được để mở cho các lập trình viên, điều quan trọng cần phải đảm bảo các tính thống nhất về mặt giao diện để các máy tính khác có thể gọi đối tượng đó. Giải pháp cho vấn đề này là xây dựng một

thành phần thích nghi đối tượng, vai trò quan trọng của nó là việc thể hiện các đối tượng phân tán trên môi trường mạng.

Nhúng Máy trạm vào đối tượng

Việc gọi các đối tượng phân tán tương tự như gọi đối tượng trên một máy tính, điểm khác biệt cơ bản là trước khi gọi hàm của đối tượng phân tán cần phải nhúng đối tượng cục bộ vào đối tượng phân tán. Có hai cách thực hiện: tường minh và không tường minh. Đối với phương pháp tường minh, cần phải khai báo biến con trỏ trên máy tính máy trạm và sau đó gọi hàm bind để nhúng đối tượng phân tán vào biến con trỏ cục bộ đó.

```
Distr_object* obj_ref; // Khai báo đối tượng tham chiếu toàn hệ thống  
obj_ref = ...; // Khởi tạo biến tham chiếu cho đối tượng phân tán  
obj_ref->do_something(); // Nhúng không tường minh, gọi hàm  
Distr_object* obj_ref; // Khai báo đối tượng tham chiếu toàn hệ thống  
Localobject* obj_ptr; // Khai báo biến con trỏ cho đối tượng cục bộ  
obj_ref = ...; // Khởi tạo biến tham chiếu cho đối tượng phân tán  
obj_ptr = bind(obj_ref); // Nhúng đối tượng bằng phương pháp tường minh  
obj_ptr->do_something(); // Gọi hàm
```

2.1.3 Truyền thông hướng thông điệp

Phương pháp gọi thủ tục từ xa hay phương pháp gọi đối tượng từ xa có ưu điểm là che giấu quá trình trao đổi thông tin trong hệ thống phân tán, góp phần nâng cao tính linh hoạt của hệ thống. Tuy nhiên điều này không phải lúc nào cũng phù hợp với các ứng dụng phân tán. Giống như quá trình gọi thủ tục trên một máy tính, máy trạm phải ở trạng thái chờ trong thời gian đối tượng phân tán trên máy chủ thực hiện các thao tác xử lý.

2.1.3.1 Tính bền bỉ và tính đồng bộ trong trao đổi thông tin

Cơ cấu tổ chức trao đổi thông điệp giữa các máy tính trong mạng được thực hiện theo qui trình sau: mỗi máy tính cung cấp giao diện trao đổi thông điệp. Khi cần chuyển thông điệp đến một máy tính khác, các ứng dụng chỉ cần chuyển các thông điệp đó đến giao diện, công việc tiếp theo sẽ do hệ điều hành đảm nhiệm. Truyền thông điệp trong hệ thống phân tán chia làm các loại sau:

- Truyền tin bền bỉ: Thông điệp được lưu tại bộ đệm, nếu không tìm thấy đích thì không bị xóa. 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 tin chuyển tiếp nhanh: Thông điệp chỉ được lưu trong thời gian thực hiện chuyển thông tin, nếu không tìm thấy đích thì thông điệp đó sẽ bị chối bỏ.
- Truyền đồng bộ: Bên gửi sẽ tạm ngừng tiến trình cho đến khi thông điệp được lưu trong vùng đệm của bên nhận.
- Truyền không đồng bộ: Bên gửi chỉ bị tạm ngừng trong thời gian chuyển thông điệp đến vùng đệm của bên gửi (chưa cần chuyển sang bên nhận).

Trong thực tế, kỹ thuật truyền tin có thể được thực hiện dựa trên tổ hợp của các đặc tính trên:

- Truyền tin không đồng bộ, bền bỉ

- Truyền tin đồng bộ, bèn bi
- Truyền tin không đồng bộ, chuyển nhanh
- Truyền tin không đồng bộ, chuyển nhanh dựa trên bên nhận
- Truyền tin không đồng bộ, chuyển nhanh dựa trên bên gửi
- Truyền tin không đồng bộ, chuyển nhanh dựa trên phản hồi

Gần đây, hầu hết các hệ thống phân tán lựa chọn phương pháp truyền tin không đồng bộ chuyển nhanh dựa trên phản hồi để áp dụng cho việc gọi thủ tục từ xa và đối tượng từ xa.

2.1.3.2 Truyền tin nhanh hướng thông điệp

Nhiều ứng dụng trên các hệ thống phân tán được xây dựng dựa trên mô hình hướng thông điệp do lớp giao vận cung cấp. Theo mô hình mạng TCP/IP, mạng máy tính được phân làm 4 lớp, để đảm bảo tính duy nhất của mỗi thông điệp trên mạng, mỗi thông điệp được gắn với cổng và địa chỉ logic của máy tính, tổ hợp cổng và địa chỉ đó gọi là socket. Để tạo điều kiện thuận lợi cho việc phát triển phần mềm trên hệ thống phân tán, mô hình mạng TCP/IP đã qui định một số hàm chuẩn (hàm nguyên thủy), quá trình truyền tin sử dụng kỹ thuật này được tóm tắt trong bảng sau:

Hàm	Ý nghĩa
Socket	Tạo Socket
Bind	Gắn địa chỉ cục bộ vào socket
Listen	Thông báo sẵn sàng nhận thông tin
Accept	Tiếp nhận yêu cầu liên kết
Connect	Thiết lập liên kết
Send	Gửi dữ liệu
Receive	Nhận dữ liệu
Close	Hủy bỏ liên kết

Giao diện chuyển thông điệp

Hạt nhân của giao diện chuyển thông điệp là các hàm nguyên thủy bao gồm:

Hàm	Ý nghĩa
MPLbsend	Thêm thông điệp vào vùng đệm gửi
MPI-send	Gửi thông điệp và chờ cho đến khi toàn bộ thông điệp đã được chuyển đến vùng đệm của bên nhận
MPLssend	Gửi thông điệp và chờ cho đến khi bên nhận bắt đầu thực hiện việc nhận thông điệp
MPLsendrecv	Chuyển thông điệp và chờ cho đến khi có xác nhận của bên nhận
MPUsend	Tham chiếu đến thông đang chờ gửi
MPLissend	Tham chiếu đến thông đang chờ gửi và chờ cho đến khi bên

	nhận bắt đầu
MPLrecv	Nhận thông điệp, phong tỏa nếu không có thông điệp
MPLirecv	Nhận thông điệp, không phong tỏa nếu không có thông điệp

2.1.4 Truyền thông hướng luồng

Những kỹ thuật truyền tin đã đề cập trên đây mới chỉ chú trọng đến việc trao đổi thông tin mà chưa nói đến thời gian thực hiện quá trình chuyển tin đó. Trong thực tế có nhiều dữ liệu cần chuyển theo thời gian thực (ví dụ âm thanh, video....). Một số dữ liệu cần phải đáp ứng yêu cầu về mặt thời gian cũng như tính liên tục của dữ liệu cần phải chuyển, khi đó cần phải kỹ thuật truyền dạng luồng. Như vậy, ở đây cần phải thực hiện theo chế độ chuyển tiếp nhan và không đồng bộ.

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 bao gồm nhiều luồng dữ liệu khác nhau như văn bản, âm thanh, hình ảnh... cần phải được đồng bộ giữa các luồng dữ liệu đó, cần phải duy trì mối tương quan giữa các luồng dữ liệu (dữ liệu có thể xuất hiện dưới dạng liên tục hoặc không liên tục). Việc duy trì đồng bộ cần phải được thực hiện tại các đầu cuối cũng như trong môi trường mạng.

2.1.5 Truyền thông theo nhóm

Truyền thông theo nhóm (multicast) 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 truyền hình, hội thảo, trình diễn báo cáo, truyền phát thông tin... Truyền thông theo nhóm có thể được cài đặt trên tầng liên kết dữ liệu hoặc tầng mạng hoặc tầng ứng dụng của mô hình OSI.

2.2 Đặt tên

Việc đặt tên đóng vai trò quan trọng trong tất cả các hệ thống máy tính, chúng dùng để xác định duy nhất thực thể khi chia sẻ tài nguyên trong hệ thống. Trong hệ thống phân tán, qui tắc đặt tên được trải rộng trên các máy tính khác nhau.

2.2.1 Tên, định danh và địa chỉ

Trong hệ thống phân tán, tên là một xâu các bit hoặc cá ký tự dùng để tham chiếu đến một thực thể, thực thể đó có thể là máy tính, ổ đĩa hoặc các tập tin. Một thực thể có thể có nhiều tên. Tên (name) là xâu các bit hoặc ký tự dùng để tham chiếu đến một thực thể trong hệ phân tán. Địa chỉ (address) là điểm truy nhập (access point) đến thực thể. Các điểm truy nhập 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) là một kiểu tên đặc biệt, mỗi thực thể chỉ được tham chiếu bởi duy nhất một định danh và mỗi định danh tham chiếu duy nhất tới một thực thể. Không gian tên (Name space) dùng để biểu diễn tất cả các tên, 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 là đồ thị mà mỗi nút là miêu tả một thực thể. Mỗi nút thư mục gắn với nhiều nút khác và được lưu trữ trong bảng thư mục bao gồm tập các cặp nhãn và định danh. Tên thân thiện (Human-friendly name) là các tên được đặt một cách dễ hiểu,

thân thuộc với con người. Để truy nhập đến một thực thể cần phải biết điểm truy nhập hay còn gọi là địa chỉ của thực thể đó. Địa chỉ là một loại tên đặc biệt:

- Một tên có thể có 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
- Định danh là một loại tên đặc biệt, nó cần phải đáp ứng ba yêu cầu sau:
- Định danh chỉ tham chiếu đến nhiều nhất 1 thực thể
- Mỗi thực thể chỉ có một định danh
- Không được tái sử dụng định danh.

Với việc sử dụng định danh sẽ tránh được những tình huống nhập nhằng trong việc truy nhập thực thể. Trong hệ thống phân tán, việc truy nhập đến một thực thể cần phải tham chiếu đến nhiều tên (đường dẫn tới thực thể).

2.2.2 *Đặt tên phẳng*

Đặt tên phẳng thuộc loại đặt tên phi cấu trúc, nó đơn thuần chỉ gồm chuỗi các bit ngẫu nhiên không chứa bất kỳ thông tin nào để định vị điểm truy nhập liên quan tới thực thể đó. Để xác định điểm truy nhập tới các thực thể của định danh, người ta sử dụng bốn giải pháp: Giải pháp đơn giản, dựa trên ~~nguồn gốc~~ (home based), hàm băm phân tán và hình thức phân cấp.

2.2.2.1 *Các giải pháp đơn giản*

Giải pháp đơn giản gồm hai phương pháp: sử dụng quảng bá (broadcast) hoặc nhóm (multicast) hoặc sử dụng con trỏ chuyên tiếp (forwarding pointer). Phương pháp quảng bá hoặc nhóm được thực hiện bằng cách gửi định danh cần tìm tới tất cả các máy, máy nào có thực thể trùng khớp với định danh cần tìm thì gửi lại một thông báo chứa địa chỉ của điểm truy nhập. Phương pháp này đòi hỏi tất cả các tiến trình đều lắng nghe và tiếp nhận yêu cầu gửi đến, nó chỉ phù hợp với qui mô nhỏ, do đó thường chỉ được áp dụng trong các mạng nội bộ. Phương pháp dùng con trỏ tiếp dựa trên nguyên tắc 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, máy khách có thể xác định ngay được địa chỉ hiện tại của thực thể này nhờ vết địa chỉ đó.

2.2.2.2 *Cách tiếp cận dựa trên nguồn gốc*

Cách tiếp cận dựa trên nguồn gốc (Home-based): cấp phát cho mỗi thực thể một vị trí gốc, phương pháp này sẽ tạo ra một vị trí gốc để lưu giữ địa chỉ hiện tại của các thực thể (thường là nơi thực thể được tạo ra). Địa chỉ của gốc được đăng ký tại dịch vụ đặt tên. Gốc đăng ký địa chỉ ngoài của các thực thể máy khách luôn đến gốc trước tiên, và sau đó tiếp tục với các vị trí bên ngoài.

2.2.2.3 *Bảng băm phân tán*

Bảng băm phân tán (DHT) được tổ chức tương tự như bảng băm thông thường dùng để cung cấp chức năng tìm kiếm trong hệ thống phân tán. Một cặp khóa và giá trị được lưu trong DHT và bất cứ nút nào tham gia vào hệ thống cũng có thể lấy được giá trị ứng với một khóa xác định. Việc duy trì bảng ánh xạ giữa khóa và các giá trị được lưu phân tán trên các nút, do đó việc thay đổi của một số nút tham gia vào hệ thống sẽ chỉ ảnh hưởng đến một số nhỏ các khóa liên quan. Điều này giúp cho DHT có thể dễ

dàng mở rộng với số lượng lớn nút tham gia, và cung cấp khả năng duy trì hệ thống khi có nút tham gia, rời khỏi mạng, hay bị lỗi.

Ưu điểm của phương pháp sử dụng hàm băm phân tán là nó không sử dụng bất kì một máy trung tâm nào để quản lý, dễ dàng thích nghi mở rộng hệ thống. DHT chưa khôn gian khóa ảo (ví dụ như chuỗi có độ dài 160 bit), chúng được phân vùng cho từng nút trong hệ thống. Một mạng phủ kết nối các nút với nhau, giúp các nút này tìm được nút đang giữ thông tin về một khóa trong khôn gian khóa.

Để lưu một tập tin với tên và dữ liệu của nó trong DHT, thuật toán SHA-1 được sử dụng để tạo mã băm của tên tập tin – là khóa k có độ dài 160 bit. Tiếp đó một thông báo put(k,data) được gửi đến các nút trong mạng DHT. Thông điệp này được chuyển tiếp qua các nút qua mạng phủ cho đến khi tới được nút giữ trách nhiệm lưu giữ khóa k được quy định bởi cách phân bổ khôn gian khóa. Nút đó sẽ thực hiện lưu giữ khóa và dữ liệu. Các nút khác có thể lấy thông tin của file bằng cách thực hiện hàm băm trên tên file để lấy được khóa k, sau đó truy vấn bất kỳ nút nào trong mạng DHT để tìm kiếm dữ liệu ứng với khóa k bằng thông điệp get(k). Thông điệp này tương tự được truyền trên mạng phủ thông qua các nút cho đến khi tới nút lưu giữ thông tin về khóa k, nút này sẽ trả lại thông tin về dữ liệu ứng với khóa.

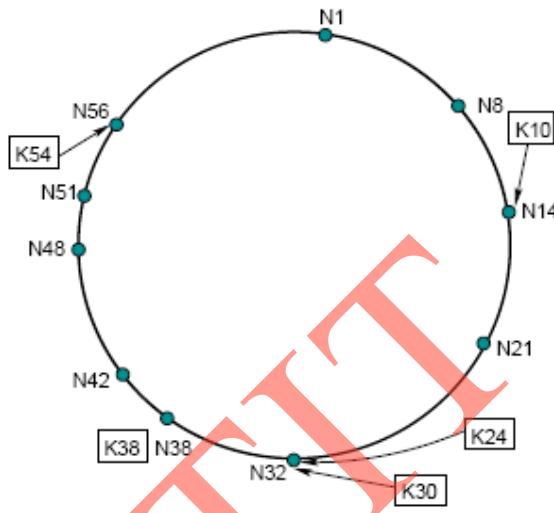
Phần lớn các hệ thống DHT sử dụng phương pháp băm nhất quán để ánh xạ khóa vào các nút. Kĩ thuật này cung cấp một hàm $\delta(k_1, k_2)$ để tính khoảng cách giữa hai khóa k_1 và k_2 . Mỗi nút được gán cho một khóa định danh ID. Một nút với ID là i_x sẽ có trách nhiệm lưu trữ với tất cả các khóa k_m nếu như i_x là định danh nút gần nhất với các khóa đó, tính toán bằng hàm $\delta(k_1, k_2)$. Khi có sự thay đổi một nút trong mạng chỉ có những khóa thuộc nút đó mới cần chuyển sang các nút lân cận, trong khi không tác động gì đến các nút khác. Điểm này hoàn toàn khác biệt với phương pháp bảng băm thông thường, khi thay đổi một phần sẽ khiến cho gần như toàn bộ khôn gian khóa phải tính toán lại. Do việc chuyển đổi khóa từ nút này sang nút khác yêu cầu lượng bảng thông trong việc chuyển đổi các dữ liệu, do đó để đáp ứng điều kiện mạng có nhiều biến động (nút ra vào với tần suất cao) thì việc có ít thay đổi lên cấu trúc mỗi khi có thay đổi là yêu cầu cấp thiết.

Mỗi nút duy trì một tập các đường dẫn đến các nút khác (các nút láng giềng) hay còn gọi là bảng định tuyến. Tập các đường dẫn này tạo lên mạng phủ. Một nút chọn các nút láng giềng dựa theo một cấu trúc nhất định gọi là topology của mạng. Tất cả các hình trạng của DHT đều chứa các đặc điểm nhất định như: với khóa k bất kỳ, mỗi nút hoặc sẽ có là nút lưu trữ khóa k hoặc có đường dẫn tới nút có định danh gần với khóa k hơn, theo nghĩa về khoảng cách giữa các khóa đã nêu ở trên. Ngoài việc đảm bảo định tuyến một cách chính xác, một hình trạng cần phải đảm bảo hai yếu tố quan trọng là đảm bảo số lượng tối đa các nút phải đi qua để trả lời một truy vấn phải nhỏ để đảm bảo đáp ứng nhanh truy vấn và số lượng láng giềng của một nút (bậc của nút) phải nhỏ để đảm bảo không làm gây khó khăn trong việc duy trì hệ thống.

Chord là một trong những giao thức phổ biến nhất được sử dụng trong bảng băm phân tán. Mạng Chord hỗ trợ khả năng gán tương ứng một khóa cho trước với một nút mạng. Tùy thuộc vào ứng dụng sử dụng Chord, nút đó có thể đảm nhiệm nhiệm vụ lưu trữ dữ liệu được gán với khóa đó. Chord sử dụng phương pháp consistent hashing, giàn tiếp thực hiện việc cân bằng tải giữa các nút do mỗi nút được gán với một số lượng key tương đương nhau. Việc tham gia và rời khỏi mạng sẽ chỉ khiến cho một số nhỏ các key chuyển từ nút này sang nút khác. Đặc điểm khiến Chord

trở nên thông dụng chính là khả năng mở rộng mạng. Trong khi với các thuật toán trước đó một nút cần phải duy trì thông tin về nhiều nút khác trong mạng thì Chord chỉ cần một số lượng cố định. Chính điều này giúp cho Chord có thể hoạt động hiệu quả trong mạng có số lượng các nút lớn.

Các nút trong mạng Chord tạo lên một mạng logic dạng vòng tròn có các vị trí nút từ 0 đến $2^m - 1$. Khóa k được gán cho nút đầu tiên có định danh bằng hoặc lớn hơn định danh của k. Nút đó được gọi là nút successor của khóa k hay successor(k). Trong vòng định danh của Chord successor của một khóa chính là nút gần nhất theo chiều kim đồng hồ tính từ khóa k.



Hình 2.13: Mô hình vòng Chord với khóa có chiều dài 6 bit

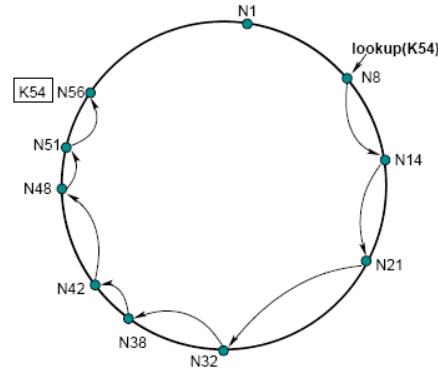
Hình 2.13 minh họa mô hình vòng Chord với $m=6$. Vòng Chord có chứa 10 nút và 5 khóa. Successor của định danh 10 là nút 14 do đó key 10 sẽ được đặt ở nút 14. Tương tự khóa 24 và 30 sẽ được đặt ở nút 32, khóa 38 tại nút 38 và khóa 54 tại nút 56. Kĩ thuật băm nhất quán được thiết kế để việc các nút tham gia hay rời khỏi mạng sẽ tạo ra ít ảnh hưởng nhất. Để duy trì bảng mapping khi một nút n tham gia vào mạng, một số khóa trước đây được đặt tại successor của n sẽ chuyển sang cho nút n. Trong ví dụ trên, nếu có một nút với định danh 26 tham gia vào mạng, nó sẽ nhận được khóa 24 chuyển từ nút 32. Successor của một nút là nút tiếp sau nút đó trên vòng Chord, predecessor là nút liền trước trên vòng Chord.

Tìm kiếm đơn giản là thuật toán tìm kiếm đơn giản nhất trong Chord. 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. Hình 2.14 là ví dụ khi nút 8 thực hiện truy vấn cho khóa có định danh 54. Nút 8 gọi hàm find_successor cho khóa 54, kết quả trả về là nút 56 – successor của khóa 54 này. Truy vấn được chuyển lần lượt qua tất cả các nút trên vòng nằm giữa nút 8 và 56. Thuật toán tìm kiếm ở trên sử dụng một số lượng thông báo tương ứng tuyến tính với số nút có trong mạng.

Hệ thống phân tán

```
// ask node n to find the successor of id
n.find_successor(id)
if (id ∈ (n, successor])
    return successor;
else
    // forward the query around the circle
    return successor.find_successor(id);
```

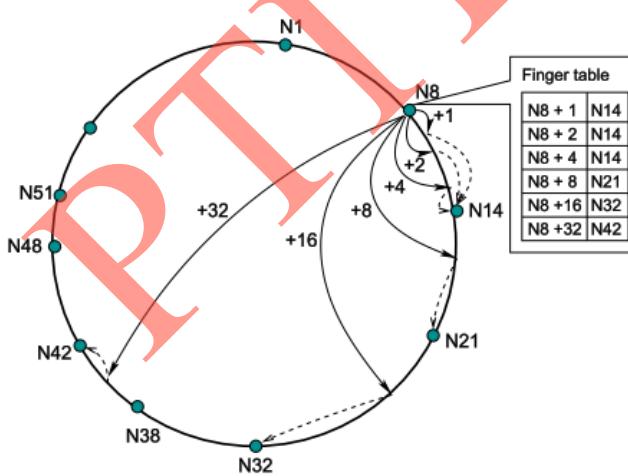
(a)



(b)

Hình 2.14: Quá trình tìm kiếm đơn giản trên Chord

Để tăng tốc độ quá trình tìm kiếm Chord sử dụng thêm một số thông tin định tuyến. Tương tự như trên, ví dụ định danh của mỗi nút và khóa có độ dài m bit. Mỗi nút n duy trì một bảng định tuyến chứa m mục, được gọi là bảng finger. Mục thứ i trong bảng của nút n chứa định danh của nút s sao cho s là nút đầu tiên trên vòng tiếp sau khóa $n+2^i-1$ ($s = \text{successor}(n+2^i-1)$, với $1 \leq i \leq m$ (lấy số dư với modun 2^m). Ta gọi s là finger thứ i của nút n . Finger đầu tiên của một nút cũng chính là successor của nút đó.



Hình 2.15: Bảng finger của nút 8

Hình 2.15 thể hiện bảng finger của nút 8. Finger đầu tiên được trỏ đến nút 14 do 14 là nút liền sau $(8+20) \bmod 26 = 9$. Tương tự finger cuối cùng của nút 8 trỏ đến nút 42 do 42 là nút liền sau $(8+25) \bmod 26 = 40$. Có thể dễ nhận thấy với các thiết lập như vậy: một nút chỉ lưu thông tin về một số giới hạn các nút có trong mạng, một nút cũng chỉ biết đến một số nút nằm gần với nó. Một nút cũng không lưu trữ đủ thông tin để có thể ngay lập tức tìm được successor của một khóa k.

```

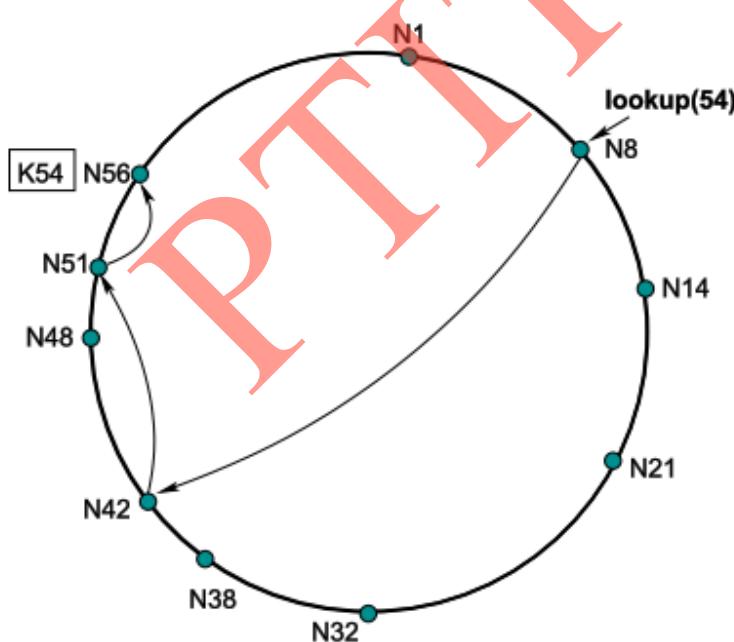
// ask node n to find the successor of id
n.find_successor(id)
  if (id ∈ (n, successor])
    return successor;
  else
    n' = closest_preceding_node(id);
    return n'.find successor(id);

// search the local table for the highest predecessor of id
n.closest_preceding_node(id)
  for i = m downto 1
    if (finger[i] ∈ (n, id))
      return finger[i];
  return n;

```

Hình 2.16 Giả mã của phương pháp tìm kiếm cài tiến

Hình 2.16 thể hiện đoạn giả mã ứng với việc thực hiện tìm kiếm successor của key id có sử dụng bảng finger. Nếu id nằm giữa n và successor của nó, find_successor sẽ trả lại successor của nó. Nếu không n tìm kiếm trong bảng finger cho nút n' – có định danh ngay sau id cần tìm kiếm và thực hiện hàm find_successor trên nút n'.



Hình 2.17 : Quá trình tìm kiếm khóa 54 trên nút 8

Như hình 2.17 ở trên nút 8 tìm kiếm successor của khóa 54. Qua bảng finger của nút 8 ta thấy nút 42 là nút gần khóa cần tìm kiếm nhất, nên nút 8 sẽ thông qua nút 42, tương tự query sẽ chuyển qua nút 51 và đến đích là nút 56. Có thể chứng minh được định lý có nội dung như sau: Với xác suất cao số nút cần thông qua để tìm kiếm successor trong một mạng N nút là $O(\log N)$.

Trên thực tế, mạng Chord cần phải giải quyết các vấn đề như việc một nút mới tham gia vào mạng, rời khỏi mạng và đột ngột rời khỏi mạng. Để tham gia vào mạng

một nút n thực hiện 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, ở vị trí nằm giữa successor s và predecessor của s thông qua quá trình ổn định mạng. Bảng finger của n được khởi tạo bằng cách sao chép bảng finger của s hoặc để s lần lượt tìm kiếm các finger cho n . Các nút cần thay đổi bảng finger khi có sự tham gia của n sẽ lần lượt thực hiện việc này thông qua quá trình ổn định mạng chạy định kỳ. Cuối cùng các khóa đang được giữ bởi s , có id nhỏ hơn hoặc bằng n sẽ được chuyển qua n .

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. Bảng finger trên các nút khác sẽ dần dần được điều chỉnh thông qua quá trình ổn định mạng định kỳ. Khi một nút đột ngột rời khỏi mạng sẽ gây ra các hậu quả như sau: Đầu tiên việc này có thể gây mất các khóa (các item liên quan đến khóa). Thứ hai: một bộ phận các nút sẽ không truy vấn được một số khóa nhất định. Chord giải quyết vấn đề này bằng cách 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.

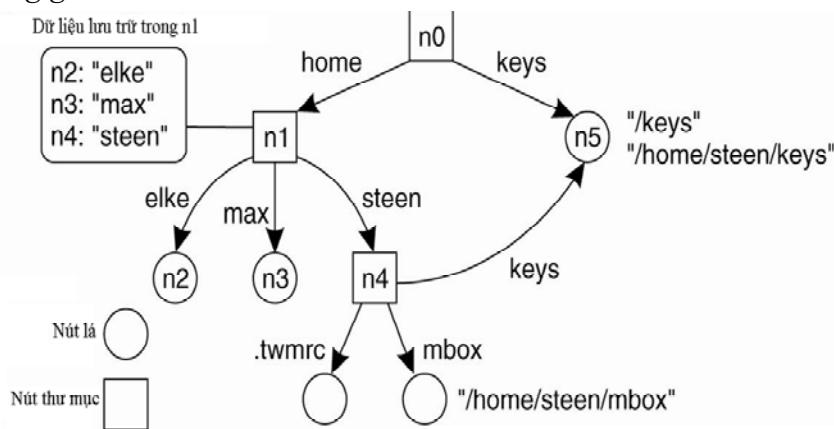
2.2.2.4 Cách tiếp cận phân cấp

Xây dựng một cây tìm kiếm phân cấp và thực hiện phân miền ở các mức khác nhau. Mỗi miền hình dung như một nút thư mục riêng biệt. Nút gốc biết tất cả các thực thể. Mỗi thực thể trong một miền tương ứng với một bản ghi trong nút thư mục, nó là địa chỉ hiện tại của thực thể hoặc một con trỏ. 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 nằm 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). Nguyên lý cơ bản bắt đầu tìm kiếm ở các nút lá cục bộ, nếu nút đó biết thực thể, tiếp theo sẽ đi xuống phía dưới theo con trỏ, ngược lại đi lên trên.

2.2.3 Đặt tên có cấu trúc

Đặt tên phẳng phù hợp các máy tính nhưng nó hoàn toàn không thân thiện với con người, con người cần những tên đơn giản và dễ đọc, cho dù đó là tên của máy tính hay tên của các tập tin.

2.2.3.1 Không gian tên



Hình 2.18 Đồ thị tên

Các tên thường được tổ chức thành không gian tên, chúng được thể hiện bằng các nhãn hoặc đồ thị có hướng với hai loại nút: nút lá và nút thư mục. Nút lá thể hiện thực thể được đặt tên và không còn nhánh nào đi ra nút đó nữa, nó chứa thông tin thể hiện thực thể ví dụ như địa chỉ, trạng thái... Mỗi nút thư mục được gán nhãn và có các nhánh đi ra, nó chứa bảng thư mục thể hiện thông tin của các nhánh đi ra từ nút đó.

2.2.3.2 Phân giải tên

Phân giải tên là chức năng cần thiết cho việc trao đổi thông tin giữa các tiến trình. Đó là quá trình chuyển đổi tên thành địa chỉ và có thể được thực hiện tập trung hoặc phân tán. Không gian tên đưa ra kĩ thuật lưu trữ và tìm kiếm các tên trên nó một cách dễ dàng. 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, bao gồm:

- Kỹ thuật gần (Closure): là kĩ thuật cho ta biết quá trình tìm kiếm tên được bắt đầu như thế nào và bắt đầu ở đâu.
- Kỹ thuật liên kết (Linking): kĩ thuật này sử dụng bí danh (alias) - tên giống với tên của thực thể. Với kĩ 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 thể đó.
- Kỹ thuật gắn kết (Mounting): là kĩ thuật được thực hiện khi tìm kiếm trên hai không gian tên. Một nút thư mục được gọi là một điểm gắn kết lưu giữ định danh 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 điểm gắn kết.

Thông thường, nếu hai không gian tên NS1, NS2 - để gắn kết một thực thể bên ngoài trong hệ phân tán, chúng ta cần tối thiểu những thông tin sau: Tên của giao thức truy nhập (được xác định để thực hiện giao thức truyền thông), tên của máy chủ (xác định địa chỉ máy chủ) và tên của điểm truy nhập (xác định định danh của nút trong không gian tên bên ngoài).

Phân giải tên tương tác (interactive name resolution): việc phân giải tên thực hiện bằng cách truyền và nhận qua lại giữa máy khách và các name máy chủ ở các mức khác nhau. Theo cách này thì các máy chủ không trao đổi trực tiếp với nhau, mỗi máy chủ chỉ phân giải nhãn tương ứng với lớp để xác định địa chỉ của máy chủ tiếp theo, kết quả trả lại cho máy khách là địa chỉ của name máy chủ tiếp theo, và việc liên kết với máy chủ tiếp theo là do máy khách đảm nhiệm.

Phân giải tên đệ quy (recursive name resolution): theo cách này thì mỗi máy chủ quản lý tên sẽ gửi kết quả đến máy chủ quản lý tên tiếp theo mà nó tìm thấy. Và cứ như vậy cho đến khi hoàn thành phân giải toàn bộ đường dẫn.

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:

- 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.
- Mô hình hai lớp: phân biệt tên và địa chỉ nhờ định danh thực thể. Gồm hai quá trình: quá trình tìm định danh thực thể 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 định danh được thực hiện bởi dịch vụ định vị (Location service).

2.2.3.3 Cài đặt không gian tên

Quản lý tên trong hệ phân tán thực hiện bằng cách phân mức quản lý:

- Mức toàn cầu (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 toàn cầu được yêu cầu cao hơn so với các lớp còn lại. Nếu máy chủ phân giải tên của lớp này bị lỗi thì việc phân giải tên không thể thực hiện.
- Mức quản trị (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 sẵn sàng của name máy chủ trong lớp administrative là rất quan trọng đối với các máy khách do name máy chủ quản lý. Vì nếu máy chủ này lỗi thì có rất nhiều các tài nguyên không thể truy cập.
- Mức đơn vị (Managerial): Chứa những nút thư mục ở mức thấp, các nút trong mức này thay đổi khá thường xuyên (Ví dụ như các máy trong một mạng nội bộ). Yêu cầu đối với tính sẵn sàng của máy chủ quản lý tên của lớp này ít khắt khe hơn so với hai lớp trên. Tuy nhiên, hiệu suất hoạt động yêu cầu đối với lớp này cao hơn do phải thường xuyên cập nhật các thay đổi.

Không gian tên là trái tim của dịch vụ đặt tên, nó cho phép người sử dụng và các tiến trình thêm, loại bỏ và tìm kiếm tên. Dịch vụ đặt tên được cài đặt trên các máy chủ. Việc tìm tên có thể được thực hiện qua một hoặc nhiều máy chủ. Đối với mạng Internet toàn cầu, việc đặt tên thường được tổ chức thành chuỗi ký tự gợi nhớ như viết tắt tên nước hoặc loại tên (doanh nghiệp, tổ chức phi chính phủ, tổ chức đào tạo...).

2.2.3.4 Ví dụ về hệ thống tên miền

Một trong những dịch vụ đặt tên phân tán lớn nhất hiện nay là hệ thống tên miền DNS (Domain Name System) của mạng Internet. Hệ thống này được phát minh vào năm 1984 cho phép thiết lập tương ứng giữa địa chỉ IP và tên miền. Hệ thống tên miền là một hệ thống đặt tên theo thứ tự cho máy tính, dịch vụ, hoặc bất kỳ nguồn lực nào tham gia vào mạng Internet. DNS liên kết nhiều thông tin đa dạng với tên miền được gán cho những thành phần tham gia và chuyển tên miền thân thiện với con người vào định danh của tên đó. Ví dụ tên miền www.ptit.edu.vn dễ nhớ đối với con người nhưng lại không áp dụng được cho việc định tuyến, trước khi tham gia định tuyến, tên miền này sẽ được chuyển thành địa chỉ 113.171.248.52.

Tên miền nên được đặt đơn giản và có tính chất gợi nhớ với mục đích và phạm vi hoạt động của tổ chức sở hữu tên miền. Mỗi tên miền được có tối đa 63 ký tự bao gồm cả dấu “.”. Tên miền được đặt bằng các ký tự (a-z A-Z 0-9) và ký tự “-”. Một tên miền đầy đủ có chiều dài không vượt quá 255 ký tự. Hệ thống tên miền sẽ phân chia trách nhiệm gán tên miền và lập bản đồ những tên tới địa chỉ IP bằng cách định rõ những máy chủ có thẩm quyền cho mỗi loại tên miền. Những máy chủ có thẩm quyền được phân công chịu trách nhiệm đối với tên miền riêng của họ và lần lượt có thể chỉ định tên máy chủ khác cho các tên miền phụ.

Ban đầu, tất cả các tên miền và địa chỉ IP tương ứng được lưu giữ trong tập tin hosts.txt tại trung tâm thông tin mạng NIC (Network Information Center) của Mỹ. Tuy

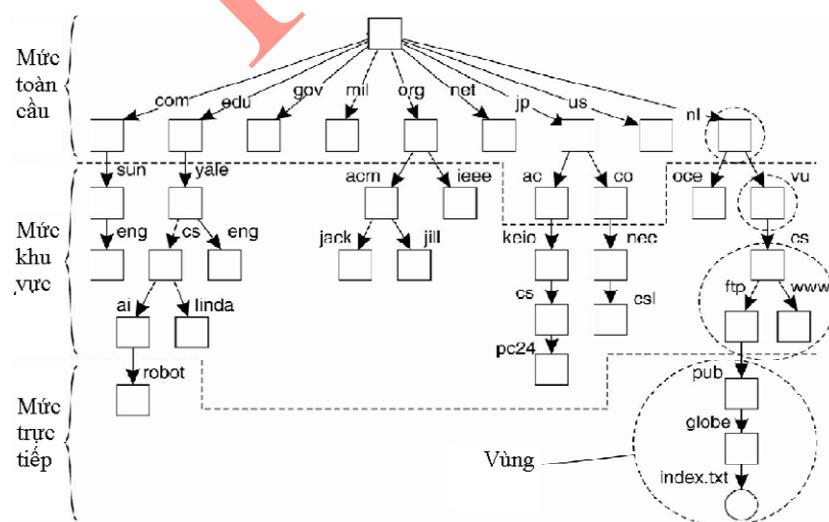
Hệ thống phân tán

nhiên khi hệ thống Internet phát triển, việc lưu giữ thông tin trong một tập không thể đáp ứng nhu cầu phân phối và cập nhật. Do đó, hệ thống tên miền DNS đã phát triển dưới dạng các cơ sở dữ liệu phân tán, mỗi cơ sở dữ liệu này sẽ quản lý một phần trong hệ thống tên miền.

Hiện nay hệ thống tên miền trên thế giới được phân bố theo cấu trúc hình cây. tên miền cấp cao nhất là tên miền gốc (ROOT) được thể hiện bằng dấu "..". Dưới tên miền gốc có hai loại tên miền là: tên miền cấp cao dùng chung- gTLDs (generic Top Level Domains) và tên miền cấp cao quốc gia – ccTLD (country code Top Level Domains) như .vn, .jp, .kr, Các tên miền iTLD và usTLD thực chất thuộc nhóm gTLD (việc phân tách ra chỉ có ý nghĩa lịch sử). Tên miền cấp cao dùng chung hiện nay do tổ chức ICANN (Internet Corporation for Assigned Names and Numbers) quản lý, tham khảo tại địa chỉ <https://www.icann.org/resources/pages/listing-2012-02-25-en>.

Tên miền cũng có thể được phân theo lĩnh vực hoạt động gồm các loại sau:

- COM, BIZ: các tổ chức, cá nhân hoạt động thương mại.
- EDU: các tổ chức, cá nhân hoạt động trong lĩnh vực giáo dục, đào tạo.
- GOV: các cơ quan, tổ chức nhà nước ở trung ương và địa phương.
- NET: các tổ chức, cá nhân hoạt động trong lĩnh vực thiết lập và cung cấp dịch vụ mạng.
- ORG: các tổ chức hoạt động trong lĩnh vực chính trị, văn hoá, xã hội.
- INT: các tổ chức quốc tế.
- AC: các tổ chức, cá nhân hoạt động trong lĩnh vực nghiên cứu.
- PRO: các tổ chức, cá nhân hoạt động trong lĩnh vực chuyên ngành cao.
- INFO: các tổ chức, cá nhân hoạt động trong lĩnh vực cung cấp thông tin.
- HEALTH: các tổ chức, cá nhân hoạt động trong lĩnh vực dược, y tế.
- NAME: tên riêng của cá nhân tham gia hoạt động Internet.



Hình 2.19 Tổ chức hệ thống tên miền trên mạng Internet

Hệ thống tên miền được sắp xếp theo cấu trúc phân cấp. Mức trên cùng được gọi là ROOT và ký hiệu là “.”, Tổ chức quản lý hệ thống tên miền trên thế giới là The Internet Corporation for Assigned Names and Numbers (ICANN). Tổ chức này quản lý mức cao nhất của hệ thống tên miền (mức ROOT) do đó nó có quyền cấp phát các tên miền dưới mức cao nhất này. Giả sử có tên miền www.ptit.edu.vn sẽ được nhận dạng từ phải qua trái. Mục đầu tiên của tên miền này là vn nghĩa là thuộc về Việt Nam, tiếp theo đó edu nghĩa là đơn vị giáo dục, ptit là tên đơn vị và www nghĩa là trang web. Theo cấu trúc và cách phân chia trong không gian tên miền đã trình bày ở trên, người dùng khi gặp một tên miền hoàn toàn có thể biết tổ chức quản lý tên miền này thuộc lĩnh vực gì, hay tên miền này do quốc gia nào quản lý... Tên miền tận cùng bằng .VN do đó tên miền này thuộc domain .VN, điều này có nghĩa nó do tổ chức quản lý tên miền của nước Việt Nam – Trung tâm Internet Việt Nam (VNNIC) quản lý, tiếp sau là tên miền cấp 2 edu, do đó tên miền này được phân cho tổ chức hoạt động trong lĩnh vực giáo dục.

Máy chủ tên miền (name server) là máy chủ chứa cơ sở dữ liệu dùng cho việc chuyển đổi giữa tên miền và địa chỉ IP. Như cách phân cấp của hệ thống tên miền, tương ứng với mỗi cấp và mỗi loại tên miền có máy chủ tên miền phục vụ tên miền ở cấp đó và loại tên miền đó. Máy chủ tên miền ở mức ROOT sẽ chứa cơ sở dữ liệu quản lý tên miền ở mức top-level-domain. Ở mức quốc gia sẽ có máy chủ tên miền quản lý domain ở mức quốc gia.

Hệ thống DNS định nghĩa hai kiểu máy chủ tên miền là máy chủ tên miền chính (primary name server) và máy chủ tên miền phụ (secondary name server). Máy chủ tên miền chính là máy chủ tên miền lấy dữ liệu cho các zone của nó từ các file có sẵn trên máy. Máy chủ tên miền phụ là máy chủ tên miền lấy dữ liệu cho các khu vực của nó từ một máy chủ tên miền chính khác. Khi máy chủ phụ khởi động nó sẽ kết nối đến máy chủ chính để lấy dữ liệu từ máy này về cho các khu vực mà nó quản lý. Quá trình lấy dữ liệu từ máy chính về máy phụ được gọi là chuyển vùng.

Máy chủ tên miền ở mức cao nhất (ROOT name server) là máy chủ tên miền chứa các thông tin để tìm kiếm các máy chủ tên miền lưu trữ (authority) cho các tên miền thuộc mức cao nhất (top-level-domain). Máy chủ ROOT có thể đưa ra các truy vấn để tìm kiếm tối thiểu là các thông tin về địa chỉ của các máy chủ tên miền ủy quyền thuộc lớp top-level-domain chứa tên miền muốn tìm. Sau đó, các máy chủ tên miền ở mức top-level-domain có thể cung cấp các thông tin về địa chỉ của máy chủ authority cho tên miền ở mức second-level-domain chứa tên miền muốn tìm. Quá trình tìm kiếm tiếp tục cho đến khi chỉ ra được máy chủ tên miền authority cho tên miền muốn tìm.

Theo cơ chế hoạt động trên máy chủ tên miền đóng vai trò quan trọng trong việc tìm kiếm một tên miền bất kỳ trên không gian tên miền. Quá trình tìm kiếm tên miền luôn được bắt đầu bằng các truy vấn gửi cho máy chủ ROOT, nếu như các máy chủ tên miền ở mức ROOT không hoạt động, quá trình tìm kiếm tên miền sẽ không được thực hiện. Để tránh điều này không xảy ra, trên mạng Internet hiện tại có 13 hệ thống máy chủ tên miền ở mức ROOT, các máy chủ tên miền này nói chung và ngay trong cùng một hệ thống cũng được đặt tại nhiều vị trí khác nhau trên mạng Internet.

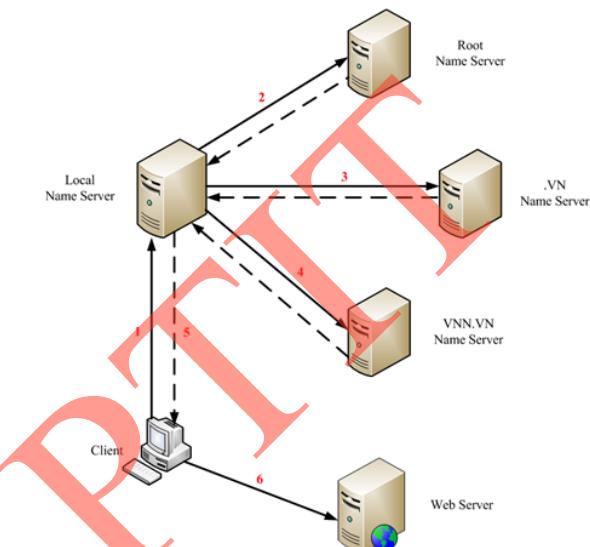
Giả sử người sử dụng muốn truy cập vào trang web có địa chỉ là www.vnn.vn. Trước hết chương trình trên máy người sử dụng gửi yêu cầu tìm kiếm địa chỉ IP ứng

Hệ thống phân tán

với tên miền www.vnn.vn tới máy chủ quản lý tên miền (name server) cục bộ thuộc mạng của nó.

Máy chủ tên miền cục bộ này kiểm tra trong cơ sở dữ liệu của nó có chứa cơ sở dữ liệu chuyển đổi từ tên miền sang địa chỉ IP của tên miền mà người sử dụng yêu cầu không. Trong trường hợp máy chủ tên miền cục bộ có cơ sở dữ liệu này, nó sẽ gửi trả lại địa chỉ IP của máy có tên miền nói trên.

Trong trường hợp máy chủ tên miền cục bộ không có cơ sở dữ liệu về tên miền này nó sẽ hỏi lên các máy chủ tên miền ở mức cao nhất (máy chủ tên miền làm việc ở mức ROOT). Máy chủ tên miền ở mức ROOT này sẽ chỉ cho máy chủ tên miền cục bộ địa chỉ của máy chủ tên miền quản lý các tên miền có đuôi .VN. Máy chủ tên miền cục bộ gửi yêu cầu đến máy chủ quản lý tên miền có đuôi (.VN) tìm tên miền www.vnn.vn. Máy chủ tên miền quản lý các tên miền.VN sẽ gửi lại địa chỉ của máy chủ quản lý tên miền vnn.vn.



Hình 2.20 Các bước thực hiện trong phân giải tên miền

Máy chủ tên miền cục bộ sẽ hỏi máy chủ quản lý tên miền vnn.vn này địa chỉ IP của tên miền www.vnn.vn. Do máy chủ quản lý tên miền vnn.vn có cơ sở dữ liệu về tên miền www.vnn.vn nên địa chỉ IP của tên miền này sẽ được gửi trả lại cho máy chủ tên miền cục bộ. Máy chủ tên miền cục bộ chuyển thông tin tìm được đến máy của người sử dụng. Người sử dụng dùng địa chỉ IP này để kết nối đến server chứa trang web có địa chỉ www.vnn.vn. Khi khai báo tên miền, các bản ghi dữ liệu tên miền gồm những loại sau:

- Bản ghi kiểu A: được dùng để khai báo ánh xạ giữa tên của một máy tính trên mạng và địa chỉ IP tương ứng của nó. Nói cách khác, bản ghi kiểu A chỉ ra tên và địa chỉ IP của một máy tính trên mạng. Bản ghi kiểu A có cú pháp như sau:

domain IN A <địa chỉ IP của máy>

Ví dụ: home.vnn.vn IN A 203.162.0.12

Theo ví dụ trên, tên miền home.vnn.vn được khai với bản ghi kiểu A trả về địa chỉ 203.162.0.12 sẽ là tên của máy tính này. Một tên miền có thể được khai nhiều

bản ghi kiểu A khác nhau để trỏ đến các địa chỉ IP khác nhau. Như vậy có thể có nhiều máy tính có cùng tên trên mạng. Ngược lại một máy tính có một địa chỉ IP có thể có nhiều tên miền trỏ đến, tuy nhiên chỉ có duy nhất một tên miền được xác định là tên của máy, đó chính là tên miền được khai với bản ghi kiểu A trỏ đến địa chỉ của máy.

- Bản ghi AAAA: được dùng để chuyển đổi từ tên miền sang địa chỉ IPv6. Giống như bản ghi kiểu A trong địa chỉ IPv4, một tên miền có thể được khai nhiều bản ghi kiểu AAAA khác nhau để trỏ đến các địa chỉ IPv6 khác nhau. Bản ghi kiểu AAAA có cú pháp như sau:

domain IN AAAA <địa chỉ IPv6 của máy>

Ví dụ:

ipv6.vnnic.net IN AAAA 2001:dc8:5::115

Khai báo trên cho thấy tên miền ipv6.vnnic.net là tên được đặt cho máy tính có địa chỉ 2001:dc8:5::115 trên mạng Internet.

- Bản ghi CNAME: Bản ghi CNAME cho phép một máy tính có thể có nhiều tên, nói cách khác bản ghi CNAME cho phép nhiều tên miền cùng trỏ đến một địa chỉ IP cho trước. Để có thể khai báo bản ghi CNAME, bắt buộc phải có bản ghi kiểu A để khai báo tên của máy. Tên miền được khai báo trong bản ghi kiểu A trỏ đến địa chỉ IP của máy được gọi là tên miền chính (canonical domain). Các tên miền khác muốn trỏ đến máy tính này phải được khai báo là bí danh của tên máy (alias domain). Cú pháp của bản ghi CNAME:

alias-domain IN CNAME canonical domain.

Ví dụ:

www.vnn.vn IN CNAME home.vnn.vn.

Tên miền www.vnn.vn sẽ là tên bí danh của tên miền home.vnn.vn , hai tên miền www.vnn.vn và home.vnn.vn sẽ cùng trỏ đến địa chỉ IP 203.162.0.12.

- Bản ghi MX: dùng để khai báo trạm chuyển tiếp thư điện tử của một tên miền. Ví dụ, để các thư điện tử có cấu trúc user@vnn.vn được gửi đến trạm chuyển tiếp thư điện tử có tên mail.vnn.vn , trên cơ sở dữ liệu của DNS cần khai báo bản ghi MX như sau:

vnn.VN IN MX 10 mail.vnn.vn

Các thông số được khai báo trong bản ghi MX nêu trên gồm có:

vnn.vn : là tên miền được khai báo để sử dụng như địa chỉ thư điện tử.

mail.vnn.vn: là tên của trạm chuyển tiếp thư điện tử, nó thực tế là tên của máy tính dùng làm trạm chuyển tiếp thư điện tử.

10 : Là giá trị ưu tiên, giá trị ưu tiên có thể là một số nguyên bất kỳ từ 1 đến 255, nếu giá trị ưu tiên này càng nhỏ thì trạm chuyển tiếp thư điện tử được khai báo sau đó sẽ là trạm chuyển tiếp thư điện tử được chuyển đến đầu tiên. Ví dụ nếu khai báo:

vnn.vn IN MX 10 mail.vnn.vn

vnn.vn IN MX 20 backupmail.vnn.vn

thì tất cả các thư điện tử có cấu trúc địa chỉ user@vnn.vn trước hết sẽ được gửi đến trạm chuyển tiếp thư điện tử mail.vnn.vn. Chỉ trong trường hợp máy chủ

mail.vnn.vn không thể nhận thư thì các thư này mới được chuyển đến trạm chuyển tiếp thư điện tử backupmail.vnn.vn.

- Bản ghi NS: dùng để khai báo máy chủ tên miền cho một tên miền. Nó cho biết các thông tin về tên miền này được khai báo trên máy chủ nào. Với mỗi một tên miền phải có tối thiểu hai máy chủ tên miền quản lý, do đó yêu cầu có tối thiểu hai bản ghi NS cho mỗi tên miền. Cú pháp của bản ghi NS:

<Tên miền> IN NS <Tên của máy chủ tên miền>

Ví dụ:

vnnic.net.VN. IN NS dns.vnnic.net.vn.

Với khai báo trên, tên miền vnnic.net.vn sẽ do máy chủ tên miền có tên dns.vnnic.net.vn quản lý. Điều này có nghĩa, các bản ghi như A, CNAME, MX ... của tên miền vnnic.net.vn và các tên miền cấp dưới của nó sẽ được khai báo trên máy chủ dns.vnnic.net.vn.

- Bản ghi PTR: Hệ thống DNS không những thực hiện việc chuyển đổi từ tên miền sang địa chỉ IP mà còn thực hiện chuyển đổi địa chỉ IP sang tên miền. Bản ghi PTR cho phép thực hiện chuyển đổi địa chỉ IP sang tên miền. Ví dụ:

12.0.162.203.in-addr.arpa IN PTR home.vnn.vn

Bản ghi PTR trên cho phép tìm tên miền home.vnn.vn khi biết địa chỉ IP mà tên miền trả về.

- Bản ghi NAPTR: là bản ghi đặc biệt trong hệ thống tên miền, chứa đựng thông tin về các nguồn, những dịch vụ và ứng dụng nào sẽ được kết hợp với một số điện thoại xác định. Những dịch vụ này được xác định và lựa chọn bởi khách hàng. Cú pháp của bản ghi NAPTR:

<Tên miền> IN NAPTR <Thứ tự> <Mức ưu tiên> <Còn> <Dịch vụ> <Biểu thức> <Thay thế>

Ví dụ:

5.1.1.6.2.2.8.4.4.8.e164.arpa. IN NAPTR 10 10 “u”“mailto+E2U”“!^.*\$!mailto:xxx@vnnic.net!”

Với khai báo trên khi thực hiện gọi số +84-xx-xxxxxxx thì ta sẽ thu được mailto:xxx@vnnic.net. Với bản ghi này có thể triển khai dịch vụ dựa trên hệ thống máy chủ DNS như ENUM và có thể mang lại sự hội tụ giữa viễn thông và Internet trong tương lai.

2.2.4 Đặt tên dựa trên thuộc tính

Đặt tên phẳng hay đặt tên có cấu trúc đều bảo đảm tham chiếu đến một thực thể duy nhất một cách độc lập với vị trí của thực thể đó. Trong thực tế, hai tiêu chí thân thiện với người dùng và độc lập vị trí là chưa đủ, người dùng muốn tìm kiếm các thực thể dựa trên các thuộc tính của chúng, mô tả càng nhiều thuộc tính càng tốt. Nếu tên được phân ra thành nhiều thuộc tính thì có thể lợi dụng thứ tự các thuộc tính. Chẳng hạn, người sử dụng với thuộc tính tên là <A>, thuộc tính tổ chức là <FO> và thuộc tính quốc gia là <VN> thì có thể tạo ra thành một thuộc tính tổng hợp là <A.FO.VN> như thuộc tính tên. Tổng quát hơn, khi không cho thứ tự các thuộc tính thì có thể định vị đối tượng nhờ tập hợp tất cả các thuộc tính. Ví dụ U=A, C=FO, O=VN khi đó U, C, và O đại diện cho các thuộc tính là tên, cơ quan và quốc gia. Những tên này được xác định dựa vào việc ghép nối kế tiếp các thuộc tính (giải pháp này dựa vào tên cấu trúc phân cấp) hoặc dựa vào chỉ tập hợp các thuộc tính (giải pháp dựa vào thuộc tính

với cấu trúc tùy ý). Việc phân chia thuộc tính cho giải pháp tên của đối tượng có thể căn cứ vào tính chất vật lý, theo tổ chức hoặc theo chức năng.

2.2.4.1 Dịch vụ thư mục

Dịch vụ thư mục tương tự với một từ điển, nó cho phép tìm kiếm của một tên và thông tin được liên kết với tên đó. Với tư cách là một từ trong từ điển có thể có nhiều định nghĩa, trong một thư mục, một tên có thể được kết hợp với thông tin nhiều, khác nhau, mẫu. Tương tự, như là một từ có thể có các phần khác nhau và các định nghĩa khác nhau, một tên trong một thư mục có thể có nhiều loại dữ liệu khác nhau. Thư mục rất có thể được thu hẹp trong phạm vi, hỗ trợ chỉ một tập hợp các loại giao điểm nhỏ và loại dữ liệu hoặc chúng có thể rất rộng, hỗ trợ một loại tùy ý hoặc mở rộng. Trong một thư mục được sử dụng bởi một hệ điều hành mạng, các giao điểm đại diện cho các nguồn tài nguyên được quản lý bởi các hệ điều hành (OS), bao gồm cả người sử dụng, máy tính, máy in và các tài nguyên chia sẻ khác.

Một thư mục đơn giản, dịch vụ đặt tên gọi là một dịch vụ bản đồ tên của mạng lưới các tài nguyên tới các địa chỉ mạng tương ứng. Với tên gọi loại hình dịch vụ thư mục, một người dùng không cần phải ghi nhớ địa chỉ lý tính của một mạng tài nguyên, cung cấp một tên sẽ tìm tài nguyên. Mỗi tài nguyên trên mạng được xem là một đối tượng trên thư mục máy chủ. Thông tin về một tài nguyên đặc biệt này được lưu giữ như là thuộc tính của đối tượng đó. Thông tin trong các đối tượng có thể được thực hiện để bảo đảm rằng chỉ có người dùng với quyền truy cập sẵn có là có thể truy cập nó. Phức tạp hơn các thư mục được thiết kế với tên như là thuê bao, dịch vụ, thiết bị, sở thích, nội dung...

Một dịch vụ thư mục là một thông tin chia sẻ về cơ sở hạ tầng cho định vị, quản lý, điều hành, tổ chức và phân biến sản phẩm và các tài nguyên mạng, mà có thể bao gồm các khối tin, thư mục, tập tin, máy in, người dùng, các nhóm, các thiết bị, số điện thoại và các đối tượng khác. Một dịch vụ thư mục là một thành phần quan trọng của một hệ điều hành mạng. Trong những trường hợp phức tạp hơn một dịch vụ thư mục là một kho thông tin trung tâm cho một nền tảng phân phối dịch vụ. Ví dụ, khi tìm kiếm cụm từ "máy tính" bằng cách sử dụng một dịch vụ thư mục có thể đưa ra một danh sách có sẵn các máy vi tính và thông tin cho việc truy nhập.

2.2.4.2 Cài đặt phân cấp LDAP

Hai tổ chức quốc tế ITU và ISO tiến tới một bộ các tiêu chuẩn – X.500, các dịch vụ thư mục, bước đầu để hỗ trợ các yêu cầu của các nhà cung cấp dịch vụ nhắn tin và tra cứu tên mạng. Giao thức thư mục nhẹ, LDAP, dựa trên các dịch vụ thư mục thông tin của X.500 nhưng sử dụng TCP / IP stack và một chuỗi mã hóa chương trình của giao thức X.500 DAP, cho nó liên quan nhiều hơn trên Internet. Đã có rất nhiều hình thức triển khai dịch vụ thư mục từ các nhà cung cấp khác nhau. Các hệ thống phát triển trước X.500 bao gồm:

- DNS: Các hệ thống tên miền, các dịch vụ thư mục đầu tiên trên Internet, mà vẫn còn được sử dụng ở khắp mọi nơi ngày nay.
- Hesiod: Các Hesiod (tên dịch vụ) dựa trên các DNS và sử dụng tại MIT của dự án Athena

- NIS: Các giao thức Dịch vụ thông tin mạng, ban đầu được đặt tên là Nhũng Trang Vàng(YP), đã được Sun Micro triển khai thực hiện một dịch vụ thư mục cho môi trường mạng Linux. Nó phục vụ một vai trò tương tự như Hesiod.

Trong số các LDAP/X.500 triển khai có thể kể đến các sản phẩm eDirectory (Novell), Red Hat Directory Server, Active Directory (Microsoft), Open Directory (Apple's Mac OS X), Apache Directory Server, Oracle Internet Directory (OID)...

2.2.4.3 Cài đặt không tập trung

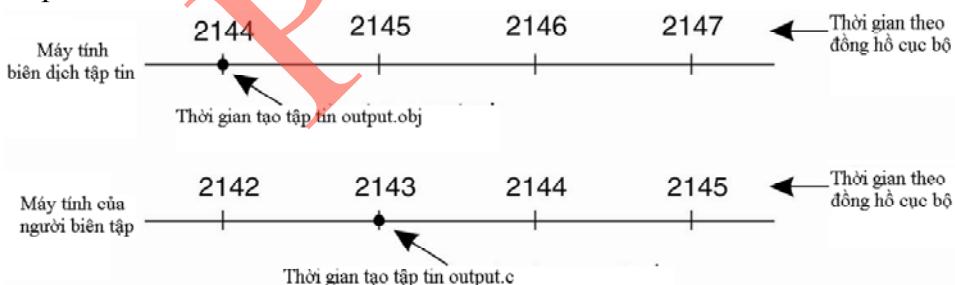
Với sự tiến bộ của các hệ thống ngang hàng, các nhà nghiên cứu đã tìm kiếm những giải pháp phi tập trung hóa hệ thống đặt tên dựa trên thuộc tính. Vấn đề mấu chốt ở đây là phải tổ chức sao cho cặp thuộc tính và giá trị được ánh xạ một cách tốt nhất sao cho quá trình tìm kiếm thực hiện nhanh nhất. Một số giải pháp đã được đề xuất như: sử dụng bảng băm phân tán, xây dựng mạng phủ ngữ nghĩa (semantic overlay network).

2.3 Đồng bộ

Đồng bộ là vấn đề quan trọng và khá phức tạp trong hệ thống phân tán, nhất là đối với các ứng dụng yêu cầu xử lý theo thời gian thực. Phần này sẽ giới thiệu các giải pháp đồng bộ về thời gian giữa các thành phần trong hệ thống phân tán, sau đó sẽ đề cập tới một số giải thuật giải quyết vấn đề tương tranh trong các hệ thống phân tán.

2.3.1 Đồng bộ đồng hồ

Trong các hệ thống tập trung, vấn đề thời gian của hệ thống tương đối đơn giản, một tiến trình khi cần biết thời gian hiện tại chỉ cần gửi yêu cầu đến hệ thống. Nếu hai tiến trình yêu cầu thời gian hệ thống thì giá trị thời gian của tiến trình gọi trước bao giờ cũng nhỏ hơn hoặc bằng giá trị thời gian của tiến trình gọi sau. Trong các hệ thống phân tán, nỗ lực đồng thuận về thời gian không phải đơn giản, chúng ta sẽ xem xét một trường hợp sau.



Hình 2.21 Hai máy tính có thời gian khác nhau

Giả sử hệ thống gồm 02 máy tính và cùng biên soạn tập tin output.c sau đó sẽ biên dịch sang tập tin output.obj. Để hiểu về nhu cầu đồng bộ thời gian giữa các thành phần trong hệ thống phân tán, chúng ta hãy xem xét trường hợp truy nhập tập tin trên hình 2., máy tính thứ nhất có thời gian nhanh hơn máy tính thứ hai. Tại thời điểm máy tính thứ nhất biên dịch, đồng hồ là 2144, trong khi đó đồng hồ trên máy tính thứ 2 là 2142, như vậy tập tin output.obj sẽ nhận thời gian được biên dịch là 2144. Sau một đơn vị thời gian, lập trình viên trên máy tính thứ hai thực hiện biên dịch, chương trình dịch sẽ kiểm tra thời gian cuối cùng biên dịch tập tin output.obj là 2044 cao hơn thời gian hiện hành 2143 máy tính thứ hai, do đó nó không thực hiện biên dịch mặc dù tập

tin output.c đã được chỉnh sửa. Như vậy đã xảy ra vấn đề nhầm lẫn về mặt thời gian khi lựa chọn tập tin được ghi gần nhất.

2.3.1.1 Đồng hồ vật lý

Hầu hết các máy tính đều có một vi mạch để duy trì thời gian, nói đúng hơn đó là bộ định thời máy tính (computer timer), thông thường đó là một bộ dao động tinh thể thạch anh. Liên quan đến tinh thể này là hai thanh ghi, một thanh ghi dùng làm bộ đếm và một thanh ghi dùng để lưu trữ, sau mỗi dao động của tinh thể bộ đếm sẽ giảm đi 1 đơn vị, khi đạt giá trị 0 sẽ sinh ra một ngắt thời gian sau đó được nạp lại giá trị lấy từ thanh ghi lưu trữ.

Mặc dù tần số của bộ dao động tinh thể thạch anh khá ổn định, nó không thể đảm bảo các tinh thể trong các máy tính khác nhau đều chạy chính xác cùng tần số. Thực tế khi một hệ thống có nhiều máy tính thì tất cả các tinh thể sẽ chạy với tần số khác nhau chút ít, dần gây ra sự mất đồng bộ và giá trị đọc ra sẽ khác nhau. Sự khác nhau về giá trị thời gian được gọi là sự sai lệch của đồng hồ và kết quả sẽ dẫn đến những sai lệch trong những tính toán có liên quan tới thời gian thực.

Trong một số hệ thống thời gian thực, thời gian đồng hồ là rất quan trọng, mỗi thành viên của hệ thống đều có một đồng hồ để xác định thời gian. Việc dùng nhiều đồng hồ vật lý như thế sẽ phát sinh vấn đề đồng bộ với thời gian thực và đồng bộ thời gian giữa các thành viên với nhau. Việc đồng bộ ~~giữa~~ các đồng hồ vật lý cần phải dựa vào một thời gian chuẩn có giá trị toàn cầu UTC (universal coordinated time). Nếu các máy tính có các bộ thu thời gian thì việc đồng bộ hóa sẽ được thực hiện theo thời gian UTC. Ngược lại, nếu các máy tính không có bộ thu thời gian thì phải sử dụng các giải thuật đồng bộ hóa đồng hồ vật lý. Ba giải thuật phổ biến đó là: giải thuật Cristian, giải thuật Berkeley và giải thuật trung bình.

Tất cả các giải thuật đều sử dụng nguyên lý cơ bản, mỗi máy xem như có một bộ đếm thời gian, nó tạo ra một ngắt H lần trong một giây. Gọi giá trị của đồng hồ này là C. Khi thời gian UTC là t, thì giá trị của đồng hồ trên máy p sẽ là Cp(t). Trong một thế giới lí tưởng chúng ta có $Cp(t) = t$ cho tất cả p và t. Hay nói cách khác, lí tưởng là $C(p).t = 1$. Bộ định thời thực không ngắt chính xác H lần trong một giây. Về lý thuyết, bộ định thời với $H = 60$ cần phát ra 216000 nhịp trong một giờ. Thực tế những sai số tương đối đạt được với các vi mạch đếm thời gian hiện đại đạt khoảng 10^{-5} , nghĩa là một máy nào đó có thể lấy giá trị từ 215998 đến 216002 nhịp trong một giờ. Một cách chính xác hơn, tồn tại một hằng số ρ thoả mãn $1 - \rho \leq dC.dt \leq 1 + \rho$. Hằng số ρ sẽ được nhà sản xuất công bố và được gọi là tỉ lệ dung sai tối đa (Maximum Drift Rate).

2.3.1.2 Hệ thống định vị toàn cầu

Hệ thống định vị toàn cầu GPS (NAVSTAR GPS - Navigation Satellite Timing and Ranging Global Positioning System) là một hệ thống các vệ tinh có khả năng xác định vị trí trên toàn cầu với độ chính xác khá cao do Bộ quốc phòng Mỹ xây dựng từ những năm 1970. Ban đầu, GPS được xây dựng để phục vụ cho các mục đích quân sự, tuy nhiên sau này cho phép sử dụng cả trong lĩnh vực dân sự. Hiện nay, hệ thống này được truy nhập bởi cả hai lĩnh vực quân sự và dân sự.

GPS bao gồm một mạng lưới 29 vệ tinh, để GPS có khả năng hoạt động tốt, số lượng vệ tinh trong mạng lưới GPS phải luôn luôn nhiều hơn 24. Để đảm bảo vùng phủ sóng liên tục trên toàn thế giới, các vệ tinh GPS được sắp xếp sao cho 4 vệ tinh sẽ nằm cùng nhau trên 1 trong 6 mặt phẳng quỹ đạo. Với cách sắp xếp này sẽ có 4 đến 10

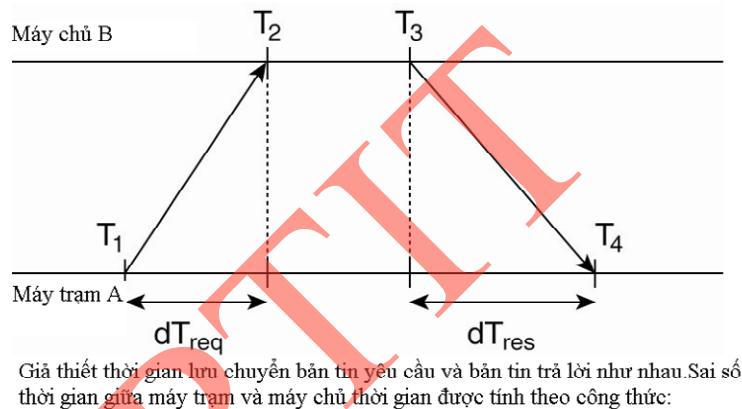
vệ tinh được nhìn thấy tại bất kỳ điểm nào trên trái đất với góc ngang là 100 nhưng thực tế chỉ cần 4 vệ tinh là có thể cung cấp đầy đủ các thông tin về vị trí. Các quỹ đạo vệ tinh GPS là những đường vòng, có dạng elip với độ lệch tâm cực đại là 0.01, nghiêng khoảng 55° so với đường xích đạo. Độ cao của các vệ tinh so với bề mặt trái đất là khoảng 20.200 km, chu kỳ quỹ đạo các vệ tinh GPS khoảng 12 giờ (11 giờ 58 phút).

2.3.1.3 Các giải thuật đồng bộ đồng hồ

Để đồng bộ thời gian giữa các máy tính người ta sử dụng một số giải thuật Cristiana, Berkeley và giải thuật trung bình.

Giải thuật Cristiana:

Giao thức thời gian mạng (NTP) 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.

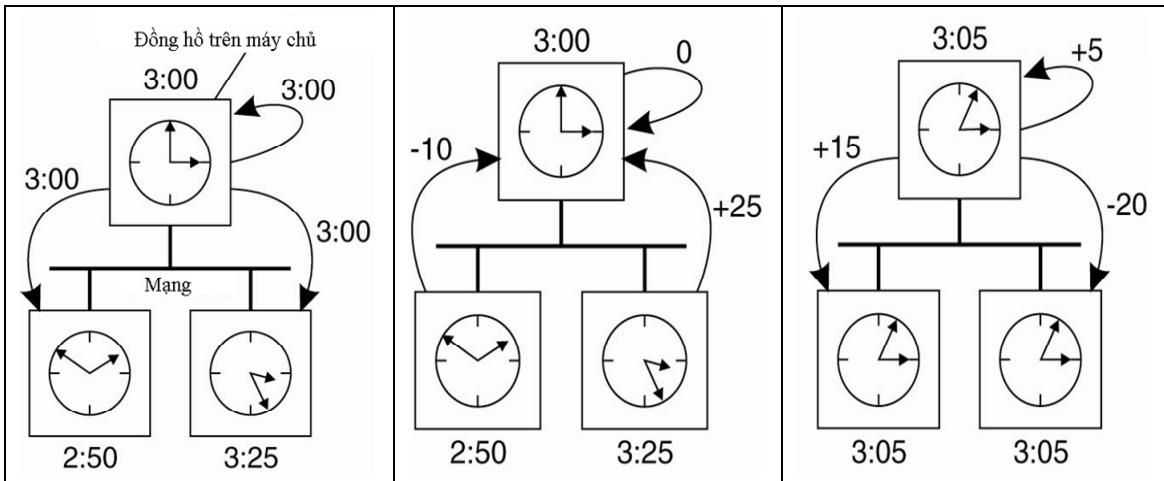


Hình 2.22 Giải thuật Cristiana

Theo chu kỳ, mỗi máy trạm gửi yêu cầu đến máy chủ để nhận được thời gian hiện hành trên máy chủ. Tại thời điểm T_0 máy trạm gửi yêu cầu và nhận được kết quả trả về từ máy chủ tại thời điểm T_1 , với giả thiết thời gian lưu chuyển thông tin yêu cầu gửi đi và kết quả trả về bằng nhau, như vậy máy trạm có thể cập nhật thời gian đồng bộ chính là kết quả trả về của máy chủ cộng thêm $(T_1 - T_0)/2$.

Giải thuật Berkeley:

Giải thuật thích hợp cho các trạm không có bộ phận nhận thời gian quốc tế WWV. Máy chủ yêu cầu máy các máy trạm cung cấp thời gian của mỗi máy, sau khi nhận được kết quả từ các máy trạm, 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.



Hình 2.23 Giải thuật Berkeley

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 biền, lấy thời gian trung bình của các tin quảng bá thời gian còn lại để làm thời gian mới của máy tính.

2.3.2 Đồng hồ logic

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. Lamport là người đầu tiên đã đưa ra mô hình đồng hồ logic, ông gọi là nhãn thời gian và sau này được gọi là nhãn thời gian Lamport.

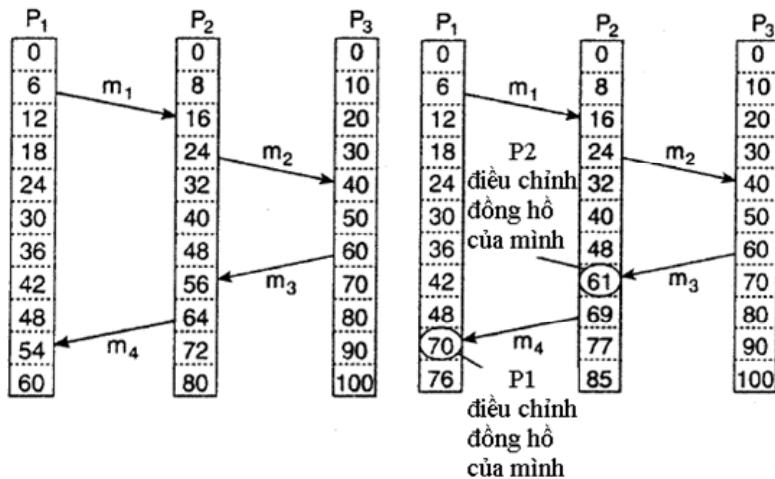
2.3.2.1 Đồng hồ logic Lamport

Để đồng bộ đồng hồ logic, Lamport dựa trên quan điểm nhân quả và đưa ra định nghĩa mối quan hệ gọi được gọi là *xảy ra - trước khi* (happens-before). 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.

Quan hệ *xảy ra - trước khi* có tính chất bắc cầu, vì thế nếu $a \rightarrow b$ và $b \rightarrow c$ thì ta sẽ có $a \rightarrow c$. Nếu hai sự kiện x và y xảy ra trong hai tiến trình khác nhau mà không xác định được mối tương quan giữa chúng thì không thể nói $x \rightarrow y$ hay $y \rightarrow x$, trường hợp này ta nói x và y là những sự kiện tranh nhau vì chúng hoàn toàn có thể xảy ra đồng thời với nhau. Ký hiệu $C(x)$ là nhãn thời gian của x, những mệnh đề sau là đúng:

- Trong cùng một tiến trình, nếu a xảy ra trước b thì $C(a) < C(b)$.
- Nếu a và b biểu diễn sự kiện gửi và nhận cùng một thông điệp thì $C(a) < C(b)$.
- Mọi sự kiện phân biệt a và b thì $C(a) \neq C(b)$.



Hình 2.24 Đồng bộ nhãn thời gian Lamport

Để đo thời gian tương ứng với một sự kiện thì ta gán một giá trị cho bộ đếm C cho sự kiện đó, thuật toán Lamport gồm các bước sau:

- Cập nhật bộ đếm C_i cho tiến trình P_i
- Trước khi thực hiện P_i gán $C_i \leftarrow C_i + 1$.
- Khi tiến trình P_i gửi thông điệp m cho tiến trình P_j , nó đặt nhãn thời gian của thông điệp $ts(m) = C_i$.
- Nhận được thông điệp m, tiến trình P_j điều chỉnh bộ đếm $C_j \leftarrow \max\{C_j, ts(m)\}$, và chuyển lên lớp ứng dụng.

Giả sử có ba tiến trình P1, P2 và P3, mỗi tiến trình đều có bộ đếm thời gian riêng nhưng nhịp đếm không giống nhau, ban đầu giá trị đồng hồ trên mỗi tiến trình đều bằng 0. Sau một khoảng thời gian nhất định, bộ đếm thời gian trên mỗi tiến trình sẽ cho những giá trị khác nhau và nếu không được điều chỉnh thì sự chênh lệch này ngày càng lớn. Ví dụ trên hình 2. khi chưa áp dụng phương pháp điều chỉnh nhãn thời gian Lamport, chênh lệch giữa tiến trình P3 và tiến trình P1 là 40 nhịp nhưng nhờ có sự điều chỉnh nhãn thời gian Lamport, giá trị này giảm xuống chỉ còn 24.

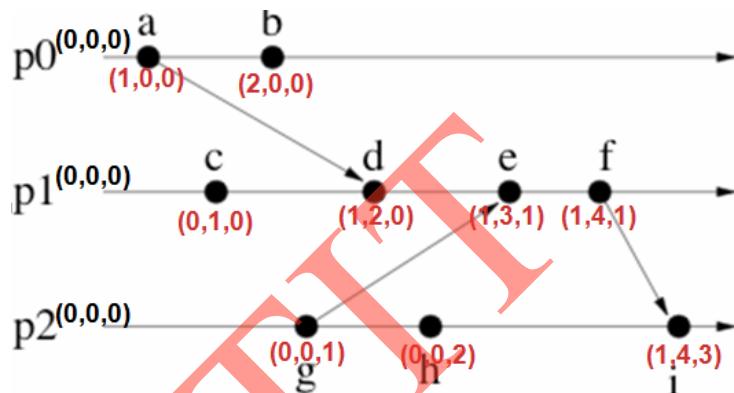
2.3.2.2 Đồng hồ vector

Một nhãn thời gian vector VT(a) được gán cho một sự kiện a có thuộc tính. Nếu sự kiện a trước sự kiện b thì ta có $VT(a) < VT(b)$. Nhãn thời gian Lamport không dễ cập nhật nguyên nhân. Hai sự kiện a và b, nếu $VT(a) < VT(b)$ thì sự kiện a là nguyên nhân của b nếu hai sự kiện a và b đều xảy ra trong một tiến trình, nếu xảy ra trên hai tiến trình thì không thể nói a hay b xảy ra trước. Trong một số trường hợp, áp dụng thuật toán nhãn thời gian vector có thể đưa ra kết luận về tính trước sau của hai sự kiện trên, do đó người ta nói đồng hồ vector là nhân hóa các sự kiện phi nhân hóa. Vector nhãn thời gian được xây dựng bằng cách để mỗi tiến trình P_i duy trì một vector V_i với hai thuộc tính sau:

- $V_i[i]$ là số sự kiện đã xảy ra cho đến thời điểm hiện tại trên tiến trình P_i .
- Nếu $V_i[j] = k$ thì P_i hiểu rằng k sự kiện đã xảy ra ở P_i .

Thuộc tính đầu tiên được duy trì bởi việc tăng $V_i[i]$ đồng thời với mỗi sự kiện mới xảy ra ở P_i . Thuộc tính thứ hai được duy trì bằng các vector nhãn thời gian cùng với các thông điệp được gửi. Mỗi khi có sự kiện mới xảy ra ở tiến trình P_i thì phải tăng $VC_i[i]$ và phải đảm bảo vector này được gửi cùng thông điệp suốt trong quá trình. Thuật toán gán nhãn thời gian vector được phát biểu như sau:

- Mỗi tiến trình p_i duy trì n-vector V_i , ban đầu tất cả các phần tử đều bằng 0.
- Mục j trong V_i là ước chừng số bước thực hiện trong tiến trình p_i
- Tại mỗi bước, giá trị $V_i[i]$ được tăng thêm 1: $V_i[i] \leftarrow V_i[i]+1$.
- Khi tiến trình P_i gửi thông điệp m cho P_j , nó đặt nhãn thời gian vector $ts(m)=V_i[i]$.
- Sau khi nhận thông điệp, tiến trình P_j cập nhật $V_j[j] \leftarrow \max\{V_j[j], ts(m)[j]\}$
- Nếu a là sự kiện trong tiến trình p_i thì gán $V(a)$ là giá trị của V_i tại cuối sự kiện a .



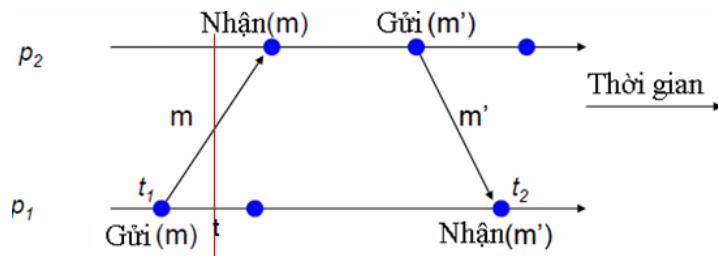
Hình 2.25 Cập nhật vector thời gian

Giả sử có ba tiến trình P_0 , P_1 và P_2 , trong mỗi tiến trình sẽ xây dựng một vector gồm ba phần tử, mỗi phần tử tương ứng với nhãn thời gian của một tiến trình, ban đầu giá trị của tất cả các phần tử đều bằng 0. Tiến trình P_0 gửi thông điệp cho tiến trình P_1 tại thời điểm giá trị vector của nó là $(1,0,0)$, nhận được thông điệp tiến trình P_1 sẽ cập nhật vector thời gian của nó thành $(1,2,0)$, lặp tương tự như vậy cho các sự kiện khác của ba tiến trình.

2.3.2.3 Các trạng thái toàn cục

Việc xác định trạng thái toàn cục của hệ thống rất có ích. Tại một thời điểm bất kỳ, 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 được đưa ra là khái niệm 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.

Hệ thống phân tán



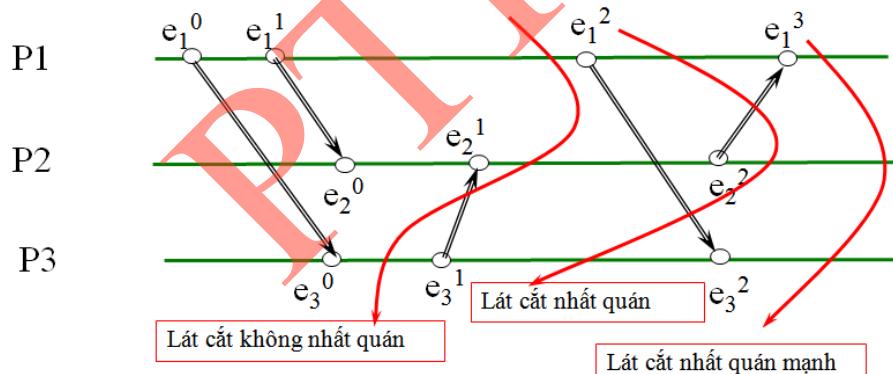
Hình 2.26 Lát cắt trong hệ thống phân tán

Nếu có đồng hồ được đồng bộ tuyệt đối, một giải pháp được đưa ra là mỗi tiến trình P_i ghi nhận trạng thái x_i tại thời điểm tương lai t nào đó sẽ gửi $x_i(t)$ đến tất cả các tiến trình khác. Vấn đề nằm ở chỗ giải pháp trên không ghi nhận 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 nới lỏng điều kiện, thay cho trạng thái toàn cục thực tế của hệ thống sẽ bắt trạng thái toàn cục nhất quán (**consistent global state**). Cho tiến trình P_i , trong đó xuất hiện các sự kiện $e_{i,0}, e_{i,1}, \dots$:

$$\text{Lịch sử tiến trình } (P_i) = h_i^0 = \langle e_{i,0}, e_{i,1}, \dots \rangle$$

$$\text{Tiền sử tiến trình } (P_i) = h_i^k = \langle e_{i,0}, e_{i,1}, \dots, e_{i,k} \rangle$$

Trạng thái tiền 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_n, \dots :

$$\text{Lịch sử toàn cục: } H = \bigcup_i (h_i)$$

$$\text{Trạng thái toàn cục: } S = \bigcup_i (S_i)$$

$$\text{Lát cắt } C \subseteq H = h_1^{c1} \cup h_2^{c2} \cup \dots \cup h_n^{cn}$$

$$\text{Ranh giới của } C = \{e_i, i = 1, 2, \dots, n\}$$

Lát cắt C là nhất quán khi và chỉ khi $\forall e \in C$ (nếu $f \rightarrow e$ thì $f \in C$). 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 toàn cục S là nhất quán khi và chỉ khi nó tương ứng với lát cắt nhất quán.

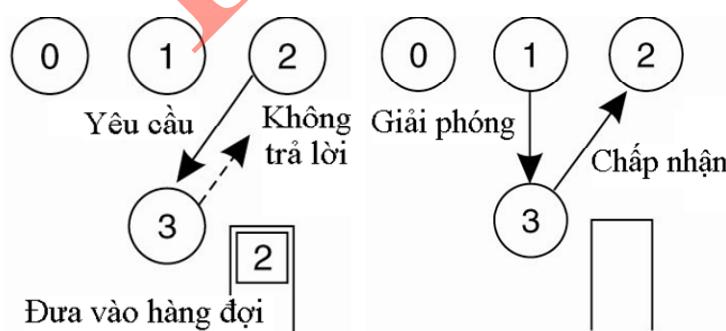
2.3.3 Loại trừ tương hỗ

Một trong những vấn đề cơ bản của các hệ thống phân tán là sự tương tranh và cộng tác giữa các tiến trình. Có thể xảy ra trường hợp cùng một lúc có nhiều tiến trình cùng truy nhập đến một tài nguyên dẫn đến sự xung đột hoặc một thao tác nào đó không được thực thi trọn vẹn. Để giải quyết vấn đề này, một số giải thuật loại trừ tương hỗ đã được đề xuất dựa trên hai phương pháp khác nhau: sử dụng thẻ bài và cấp quyền sử dụng.

Bản chất của phương pháp thẻ bài là duy trì một thông điệp duy nhất (gọi là thẻ bài) di chuyển trong một nhóm các thành viên của hệ thống. Thành viên nào nắm giữ thẻ bài thì có quyền truy nhập tài nguyên, nếu không có nhu cầu truy nhập tài nguyên thì sẽ chuyển cho thành viên kế tiếp. Phương pháp này luôn bảo đảm tránh được tương tranh vì tại một thời điểm chỉ có tối đa 01 thành viên nắm giữ thẻ bài, tuy nhiên nếu thẻ bài bị mất thì không có thành viên nào được sử dụng tài nguyên, hệ thống phải tái tạo một thẻ bài mới. Phương pháp cấp quyền sử dụng hoạt động theo hai bước, bước thứ nhất gửi yêu cầu đề nghị được cấp quyền sử dụng, sau khi được cấp quyền mới chuyển sang bước truy nhập tài nguyên.

2.3.3.1 Giải thuật tập trung

Giả sử mỗi tiến trình có một số định danh (ID) duy nhất, tiến trình được bầu chọn làm điều phối là tiến trình có số hiệu ID cao nhất. Khi một tiến trình nào đó cần vào vùng giới hạn (để truy nhập tài nguyên) nó sẽ gửi một thông điệp xin cấp quyền. Nếu 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. Khi 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 cập. Lúc này tiến trình điều phối sẽ gửi quyền truy nhập cho tiến trình đầu tiên trong hàng đợi truy nhập.



Hình 2.28 Giải thuật tập trung

Giải thuật này 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). Giải thuật này cũng có hai nhược điểm cơ bản: nếu tiến trình điều phối bị hỏng thì sẽ không có một tiến trình nào được cấp phép, hoặc 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 chỉ có một tiến trình điều phối sẽ xuất hiện hiện tượng thất cố chia.

2.3.3.2 Giải thuật không tập trung

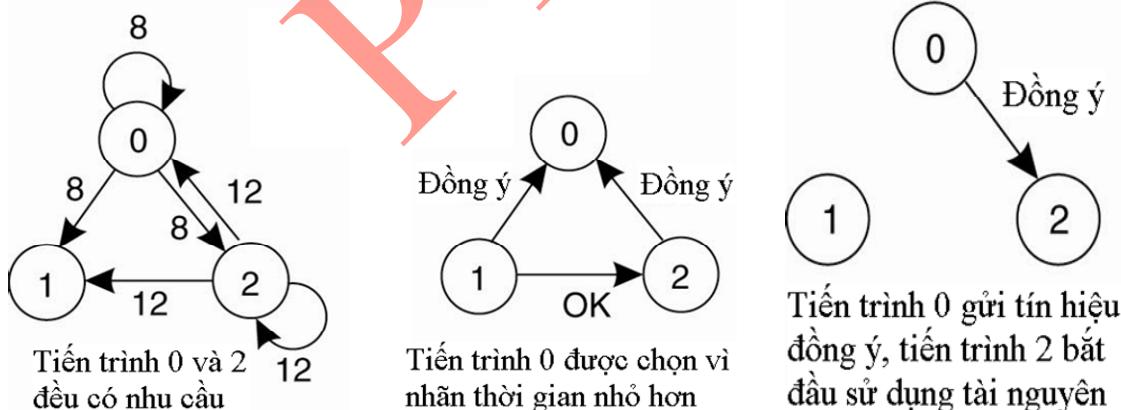
Giải thuật tập trung chỉ có một thành phần điều phối, đó là một cách tiếp cận nghèo nàn, các thành viên chỉ có duy nhất một lựa chọn. Để khắc phục nhược điểm này, mỗi tài nguyên sẽ được nhân bản N lần và có tiến trình điều phối riêng để kiểm soát truy nhập. Một tiến trình muốn truy nhập tài nguyên thì 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 $N/2$. 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:

$$P[k] = \left[\begin{matrix} m \\ k \end{matrix} \right] p^k (1-p)^{m-k}$$

2.3.3.3 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.

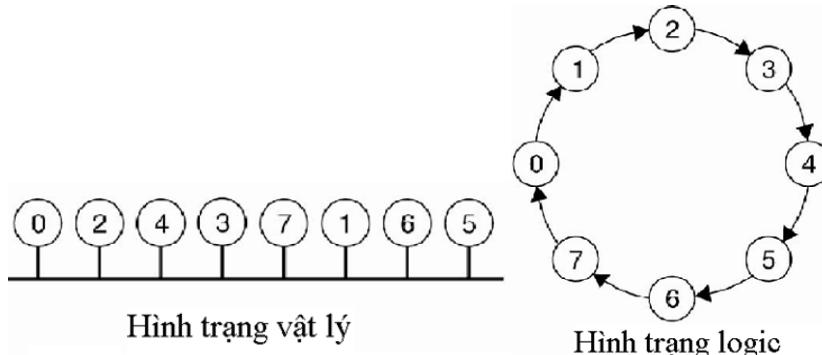


Hình 2.29 Giải thuật phân tán

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.

2.3.3.4 Giải thuật thẻ bài

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 chuyển từ tiến trình k đến tiến trình (k+1) d truyền thông điệp điểm – điểm.



Hình 2.30 Giải thuật thẻ bài

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.

2.3.3.5 So sánh các giải thuật loại trừ

Tiêu chí so sánh các giải thuật bao gồm số lượng thông điệp cần lưu chuyển, 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	$1\text{đến } \infty$	$0\text{ đén } n-1$	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 tới 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.

2.3.4 Định vị toàn cầu các nút

Khi số lượng các nút trong hệ phân tán tăng lên thì nhiệm vụ duy trì thông tin của các nút khác trở nên rất phức tạp. Trong các mạng địa lý, mỗi nút được đặt

tại một tọa độ địa lý nhất định và có thể tính toán được khoảng cách giữa các nút, nhưng khoảng cách càng xa thì càng cần nhiều thời gian để lưu chuyển thông điệp. Để giảm thiểu thời gian truy nhập người ta thường nhân bản các máy chủ dịch vụ, như vậy cần phải có các giải thuật để tìm kiếm máy chủ gần nhất đối với máy khách.

2.3.5 Các giải thuật bầu chọn

Nhiều giải thuật phân tán yêu cầu phải có một tiến trình thực hiện vai trò điều phối, khởi sướng hoặc một vai trò đặc biệt nào đó mà không cần biết đó là tiến trình nào. Như vậy, cần thiết phải có một giải thuật để tìm ra một tiến trình duy nhất đó làm tiến trình điều phối hoặc thậm chí tiến trình điều phối gấp lối 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 giải thuật Bully và giải thuật vòng.

2.3.5.1 Các giải thuật bầu chọn truyền thống

Giải thuật nổi bợt (Bully):

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 lỗi. 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 nó sẽ đảm nhiệm vai trò điều phối.

Giải thuật vòng

Giả thiết các tiến trình có một ID duy nhất và được sắp xếp trên 1 vòng tròn Logic. Mỗi một tiến trình có thể nhận biết được tiến trình bên cạnh mình. Các bước của giải thuật bao gồm: Một tiến trình bắt đầu gửi thông điệp ELEC tới các nút còn tồn tại gần nhất, quá trình gửi theo 1 hướng nhất định. Thăm dò liên tiếp trên vòng cho đến khi tìm được 1 nút còn tồn tại. Mỗi một tiến trình sẽ gắn ID của mình vào thông điệp gửi. Cuối cùng sẽ chọn ra 1 tiến trình có ID cao nhất trong số các tiến trình còn hoạt động và gửi thông điệp điều phối cho tiến trình đó.

Giải thuật loại trừ nhau

Có nhiều giải thuật được xây dựng để cài đặt cơ chế loại trừ nhau thông qua các vùng tối hạn. Có ba giải thuật phổ biến là giải thuật tập trung, giải thuật phân tán và giải thuật vòng thẻ bài.

Giải thuật tập trung

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. Khi một tiến trình nào đó cần vào vùng giới hạn nó sẽ gửi một thông điệp xin cấp quyền. Nếu không có tiến trình nào đang trong vùng giới hạn thì tiến trình điều phối sẽ gửi phản hồi cho phép. Nếu có một tiến trình khác đang ở trong vùng tối hạn rồi thì 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 tối hạn nữa. Khi 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 cập. Lúc này tiến trình điều phối sẽ gửi quyền truy cập cho tiến trình đầu tiên trong hàng đợi truy cập.

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 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 được chọn. 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 ACK đến tất cả các tiến trình khác và xóa các tiến trình trong hàng đợi đi.

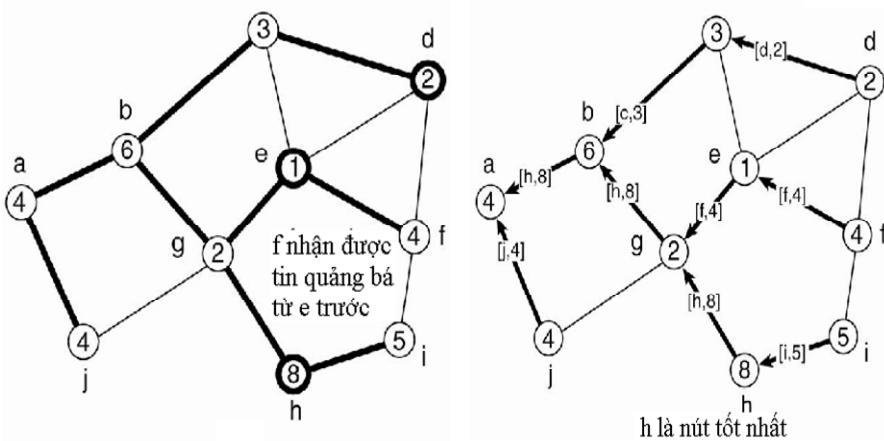
Giải thuật vòng thẻ bài

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 chuyển 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 vào 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 một tiến trình muốn vào vùng tới hạn thì nó sẽ giữ lấy thẻ bài, nếu không nó sẽ để cho thẻ bài truyền qua. Vấn đề lớn nhất trong thuật toán truyền thẻ bài là thẻ bài có thể bị mất, khi đó chúng ta phải sinh lại thẻ bài bởi vì việc dò tìm lại thẻ bài là rất khó.

2.3.5.2 Bầu chọn trong môi trường không dây

Môi trường không dây thường chịu ảnh hưởng của nhiều yếu tố về môi trường dẫn đến độ tin cậy truyền thông không cao. Các giải thuật bầu chọn thường dựa trên giả thiết việc phân phát các thông điệp tin cậy và hình trạng mạng không thay đổi, đó là những yêu cầu khó có thể đáp ứng được và như vậy chỉ có một số rất ít các giải thuật có thể áp dụng trong môi trường không dây.

Năm 2004 Vasudevan đã đề xuất giải pháp, thay vì lựa chọn ngẫu nhiên một cách chính xác như các giải thuật bầu chọn truyền thống thì chỉ cần chọn một lãnh đạo tốt nhất. Ví dụ xét một mạng không dây cơ bản, một nút nào đó trong mạng sẽ khởi tạo bằng cách gửi thông điệp ELECTION đến tất cả những nút láng giềng của mình. Khi nhận được thông điệp ELECTION, mỗi nút sẽ đánh dấu nút gửi là nút cha (nếu đó là thông điệp lần đầu) và tiếp tục gửi thông điệp ELECTION đến các nút láng giềng khác ngoại trừ nút cha và đồng thời xác nhận đã nhận được thông điệp.



Hình 2.31 Bầu chọn trong môi trường không dây

Ví dụ trên hình 2.31, giả sử nút a khởi sướng quá trình bầu chọn, nó sẽ gửi thông điệp ELECTION đến nút b và j. Hai nút này sẽ ghi nhận nút a là cha và b sẽ tiếp tục gửi thông điệp ELECTION đến nút c và g nhưng chưa gửi thông tin xác nhận cho a, j sẽ gửi đến g. Giả sử g nhận được thông điệp ELECTION từ b trước và nó ghi nhận nút b là cha, sau đó mới nhận được thông điệp ELECTION từ j, do đó nó không nhận j là cha nữa mà chỉ gửi cho j thông tin của nó (giá trị 2), j nhận được bản tin này thấy giá trị của g nhỏ hơn giá trị của nó (giá trị 4) nên gửi về cho a giá trị của nhánh j là 4. Quá trình tiếp tục như vậy cho các nút khác (đường nét đậm đánh dấu các nút cha), cuối cùng sẽ tìm được nút h với giá trị cao nhất là 8 xứng đáng làm nút lãnh đạo.

2.3.5.3 Bầu chọn trong các hệ thống qui mô lớn

Các thuật toán bầu chọn truyền thống thường chỉ áp dụng cho các hệ thống qui mô nhỏ, hơn nữa các thuật toán đó mới chỉ tập trung bầu chọn một nút. Một số tình huống đòi hỏi bầu chọn nhiều nút, ví dụ các nút đại diện của các mạng ngang hàng. Các nút đại diện cần phải đáp ứng các tiêu chí sau:

- Thông tin gửi từ các nút thường đến các nút đại diện có độ trễ nhỏ
- Các nút đại diện nên phân bố đều trên mạng bao trùm
- Nên xác định trước số lượng các nút trong mạng bao trùm
- Mỗi nút đại diện không nên phục vụ vượt quá số lượng nút bình thường đã qui định

Rất may, các tiêu chí trên hầu như dễ dàng thỏa mãn trong các mạng ngang hàng. Ví dụ, giả sử cần chọn đại diện cho hệ thống dựa trên bảng băm phân tán mà mỗi thành viên được gán m-bit định danh, chọn những định danh k-bit đại diện tính từ bên trái thì chỉ cần thực hiện phép toán AND nhị phân trong đó k bit bên trái có giá trị bằng 1 và các bit còn lại có giá trị bằng 0. Ví dụ, k=3 và m=8 khi đó chỉ cần lấy định danh và thực hiện phép tính AND với 1110 0000 sẽ cho ra nút đại diện của nhóm 8 thành viên.

2.4 Tiến trình trong các hệ thống phân tán

Khái niệm tiến trình xuất phát từ lĩnh vực hệ điều hành, một chương trình đang chạy thì được gọi là tiến trình. Một trong những nhiệm vụ chính của hệ điều hành là quản lý và lập lịch cho các tiến trình, tuy nhiên trong hệ thống phân tán sẽ phát sinh nhiều vấn đề phức tạp và quan trọng hơn rất nhiều. Tiến trình tạo nên các khối chức

năng trong hệ thống phân tán, thực tế cho thấy việc phân chia như vậy chưa đủ, cần thiết phải nhìn nhận vấn đề một cách chi tiết hơn. 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.

Trong những năm gần đây, khái niệm ảo hóa ngày càng trở nên phổ biến. Bằng kỹ thuật ảo hóa, nhiều ứng dụng và thậm chí nhiều hệ điều hành có thể chạy song hành độc lập trên một nền tảng phần cứng, điều này cho phép tận dụng tối đa tài nguyên và đồng thời cách ly được các lỗi phát sinh của mỗi tiến trình hoặc các lỗi về bảo mật. Một vấn đề quan trọng nữa là vấn đề di chuyển các tiến trình giữa các máy khác nhau trong hệ thống phân tán. Việc di chuyển tiến trình hay đặc biệt hơn là việc di trú mã có thể giúp đạt được qui mô hệ thống lớn hơn nhưng cũng có thể đạt được mục tiêu cấu hình động cho máy khách và máy chủ.

2.4.1 Các luồng

Theo quan điểm hệ điều hành, việc quản lý và lập lịch cho các tiến trình đóng vai trò quan trọng nhất. Trong hệ thống phân tán, vấn đề trao đổi thông tin giữa các tiến trình này sinh ra nhiều vấn đề khá phức tạp, ngoài việc quản lý các tiến trình cần phải đảm bảo hiệu suất hoạt động của hệ thống. Mặc dù các tiến trình hình thành nên các khối xây dựng hệ thống phân tán, nhưng thực tiễn chỉ ra rằng cần thiết phải chia nhỏ tiến trình thành các đơn vị nhỏ hơn để có thể dễ dàng hơn trong việc xây dựng các ứng dụng phân tán và đồng thời đạt được hiệu năng cao hơn. Phần này sẽ giới thiệu vai trò của các luồng trong hệ thống phân tán, chi tiết cách sử dụng để xây dựng ứng dụng phân tán cần xem thêm tài liệu *Multithreaded Programming With Pthreads* của Bill Lewis.

2.4.1.1 Khái niệm luồng

Để hiểu về vai trò của luồng trong hệ thống phân tán, trước hết cần phải hiểu tiến trình là gì và mối quan hệ của nó với các luồng. Mỗi chương trình khi chạy cần phải được cung cấp một bộ xử lý, như vậy để đáp ứng yêu cầu cùng một lúc chạy được nhiều chương trình thì hệ điều hành phải tạo ra một số bộ xử lý ảo cho mỗi chương trình. Nhằm mục đích quản lý các bộ xử lý ảo này, hệ điều hành tạo bảng tiến trình chứa thông tin giá trị các thành ghi của đơn vị xử lý trung tâm, bản đồ bộ nhớ, các tập tin đang dùng, thông tin tài khoản và các quyền thực hiện có liên quan..., do đó tiến trình có thể được hiểu như là một chương trình đang chạy trên bộ xử lý ảo. Nói cách khác, nhiều tiến trình chạy đồng thời và dùng chung đơn vị xử lý trung tâm cũng như các tài nguyên khác mà không làm ảnh hưởng tới nhau, như vậy sẽ đảm bảo tính trong suốt của hệ thống. Thường thường, hệ điều hành sẽ yêu cầu phần cứng hỗ trợ để thực hiện việc tách biệt này.

Để đạt được tính trong suốt, hệ điều hành phải trả giá khá cao trong các thao tác xử lý, nó phải tạo một không gian bộ nhớ riêng cho mỗi tiến trình. Không gian đó được mô phỏng như một hệ vi xử lý thực sự, phải có các thanh ghi và vùng nhớ để lưu trữ dữ liệu. Tại một thời điểm chỉ có duy nhất một tiến trình được phép sử dụng tài nguyên vật lý (CPU + vùng nhớ thực thi chương trình), chi phí về thời gian xử lý sẽ phát sinh trong di chuyển dữ liệu giữa bộ xử lý ảo và bộ xử lý vật lý: Hệ điều hành phải sao chép toàn bộ dữ liệu của tiến trình đang chạy vào vùng nhớ đã qui định của tiến trình đó, sau đó mới sao chép dữ liệu của tiến trình kế tiếp vào bộ xử lý vật lý.

Hệ thống phân tán

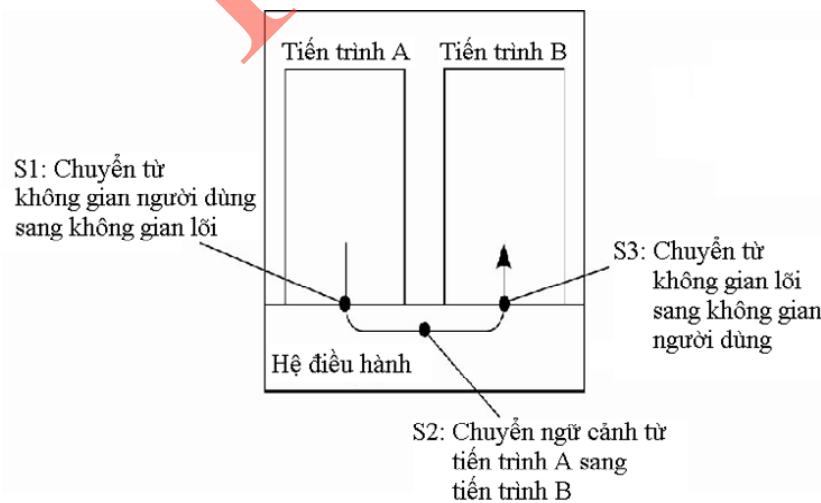
Khi các tiến trình sử dụng hết bộ nhớ chính, nó sẽ sử dụng một phần ổ đĩa để làm bộ nhớ, như vậy sẽ phát sinh rất nhiều các thao tác đọc/ghi đĩa, điều này sẽ làm suy giảm nghiêm trọng hiệu năng xử lý của hệ thống.

Giống như tiến trình, luồng thực hiện đoạn mã chương trình của mình độc lập với các luồng khác. Tuy nhiên, điểm khác biệt cơ bản so với tiến trình, luồng không có đạt độ trong suốt cao nếu thao tác đó làm suy giảm hiệu năng hoạt động, do đó hệ thống luồng thường chỉ duy trì thông tin tối thiểu nhằm chia sẻ đơn vị xử lý trung tâm. Cụ thể, vấn đề quản lý luồng được thực hiện theo cơ chế đóng/mở, tính toàn vẹn của dữ liệu do người phát triển phần mềm ứng dụng đảm nhiệm.

2.4.1.2 Luồng trong các hệ thống độc lập

Trước khi tìm hiểu vai trò của luồng trong các hệ thống phân tán, chúng ta hãy thử xem xét nó trên một hệ thống độc lập truyền thông. Ví dụ trang bảng tính Excel: Các ô trong bảng tính độc lập với nhau, nếu người sử dụng thay đổi giá trị của một ô thì sẽ làm thay đổi giá trị của các ô có liên quan với ô dữ liệu đó. Nếu chỉ sử dụng một luồng thì công việc tính toán sẽ không được thực hiện trong khi nhập dữ liệu và ngược lại khi đang tính toán thì người dùng sẽ không nhập được dữ liệu. Như vậy ở đây cần phải có hai luồng xử lý: một luồng giao tiếp với người sử dụng và một luồng khác cập nhật thông tin của trang, thậm chí cần thiết phải có luồng thứ ba thực hiện nhiệm vụ sao lưu dữ liệu vào ổ đĩa. Một ví dụ khác về ứng dụng đa luồng trong kỹ thuật xử lý song song, trong các hệ thống nhiều bộ vi xử lý, mỗi luồng được gán cho một bộ xử lý nhưng vẫn dùng chung bộ nhớ chính, kỹ thuật này thường được cài đặt trên các ứng dụng máy chủ trong mô hình khách/chủ.

Kỹ thuật xử lý đa luồng cũng được áp dụng để xây dựng những ứng dụng qui mô lớn, đó là những ứng dụng bao gồm nhiều chương trình cộng tác, mỗi chương trình sử dụng một tiến trình riêng, tương tác giữa các chương trình được thực hiện bằng cơ chế truyền thông liên tiến trình (IPC) như kỹ thuật đường ống, hàng đợi thông điệp hay vùng nhớ dùng chung. Nhược điểm cơ bản của cơ chế này là vấn đề chuyển tiến trình.

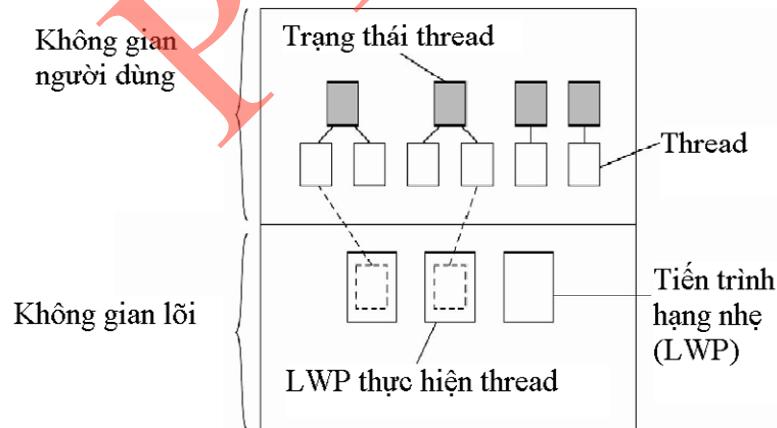


Hình 2.32 Chuyển ngữ cảnh trong truyền thông giữa các tiến trình

Hình 2.32 minh họa việc chuyển tiến trình, trước tiên tiến trình phải chuyển từ không gian người dùng sang không gian lõi, điều này sẽ làm thay đổi bản đồ bộ nhớ trong khói quản lý bộ nhớ MMU (Memory Management Unit) và xây dựng lại đệm chuyên đổi địa chỉ TLB (Translation Lookaside Buffer), bên trong lõi sẽ xảy ra quá trình chuyển ngữ cảnh xử lý và sau đó sẽ chuyển sang không gian người dùng, quá trình này cũng đòi hỏi phải thay đổi MMU và TLB. Thay cho kiến trúc tiến trình, ứng dụng có thể được phân thành nhiều phần khác nhau, mỗi phần sẽ tương ứng với một luồng. Trao đổi thông tin giữa các luồng này được thực hiện bằng việc chia sẻ dữ liệu, việc chuyển đổi từ luồng này sang luồng khác được thực hiện trong không gian người dùng, rất ít khi sử dụng đến cơ chế lập lịch trong không gian lõi. Với kỹ thuật này hiệu năng của hệ thống sẽ tăng lên đáng kể.

2.4.1.3 Cài đặt luồng

Luồng thường được cung cấp dưới dạng gói luồng, những gói như vậy bao gồm các thao tác tạo và hủy luồng giống như các thao tác trên biến đồng bộ. Có hai cách tiếp cận cơ bản: tạo thư viện luồng mức người dùng hoặc mức lõi. Với cách tiếp cận thứ nhất, luồng sẽ được thực hiện hoàn toàn trong chế độ người dùng, như vậy các thao tác tạo, hủy và chuyển luồng sẽ được thực hiện tương đối dễ dàng với chi phí khá thấp. Thao tác tạo luồng chỉ đơn giản là việc định vị không gian bộ nhớ, hủy luồng chỉ cần giải phóng vùng nhớ không còn sử dụng nữa. Chi phí thực hiện chuyển luồng cũng không lớn, về cơ bản chỉ cần vài chỉ thị lệnh chuyển dữ liệu trong các thanh ghi của đơn vị xử lý trung tâm mà không cần phải cập nhật lại bản đồ bộ nhớ và vùng đệm TBL. Cách tiếp cận này có điểm hạn chế là sử dụng phương pháp gọi phong tỏa, khi một luồng đang thực hiện thì tất cả các luồng khác trong tiến trình đó sẽ phải tạm ngừng. Cách tiếp cận thứ hai sẽ loại bỏ nhược điểm trên, khi đó tất cả các thao tác tạo, hủy hoặc chuyển luồng đều do mức lõi thực hiện, điều này lại quay trở lại mô hình tiến trình, như vậy sẽ mất đi những ưu điểm của kỹ thuật xử lý đa luồng.



Hình 2.33 Kết hợp tiến trình nhẹ và các luồng mức người dùng

Giải pháp ở đây là phải kết hợp cả hai cách tiếp cận trên, nghĩa là phải kết hợp luồng ở mức người dùng và mức lõi, những luồng như vậy gọi là tiến trình hạng nhẹ (LWP - Lightweight process). Một tiến trình chính (tiến trình hạng nặng) sẽ bao gồm một số tiến trình hạng nhẹ, thư viện luồng hoàn toàn nằm ở mức người dùng, thao tác chuyển luồng được thực hiện thông qua các biến đồng bộ mà không cần can thiệp của mức lõi.

Các tiến trình hạng nhẹ chia sẻ gói luồng và mỗi tiến trình hạng nhẹ sẽ chạy một luồng của nó ở mức người dùng. Các ứng dụng đa luồng được xây dựng bằng cách tạo ra các luồng và sau đó gán cho mỗi tiến trình hạng nhẹ, công việc này thường được thực hiện không tường minh và hoàn toàn trong suốt đối với lập trình viên. Tổ hợp hoạt động giữa gói luồng và các tiến trình hạng nhẹ được thực hiện như sau: Mỗi tiến trình hạng nhẹ sẽ tạo ra một ngăn xếp riêng chứa các luồng sẽ được thực hiện, các luồng được quản lý trong bảng luồng và được bảo vệ bằng cơ chế biến đồng bộ. Như vậy mỗi tiến trình hạng nhẹ khi gọi đến luồng nào đó chỉ cần tìm kiếm trong bảng luồng và thực hiện việc chuyển ngữ cảnh hoàn toàn trong chế độ người dùng. Trường hợp một luồng nào đó thực hiện lời gọi phong tỏa hệ thống thì nó chuyển từ chế độ người dùng sang chế độ lõi nhưng vẫn nằm trong tiến trình hạng nhẹ.

2.4.1.4 Luồng trong các hệ thống phân tán

Một đặc tính quan trọng của luồng là chúng cung cấp các phương tiện dễ dàng cho phép gọi phong tỏa hệ thống nhưng không phong tỏa toàn bộ tiến trình đang chạy, do đó kỹ thuật này thường được sử dụng trong các hệ thống phân tán để duy trì truyền thông dưới dạng các kênh logic. Ví dụ, trong mô hình khách/chủ, máy khách tạo ra nhiều luồng chạy song song độc lập với nhau, máy chủ cũng tạo ra nhiều luồng để có thể đồng thời tiếp nhận và xử lý các yêu cầu của máy khách.

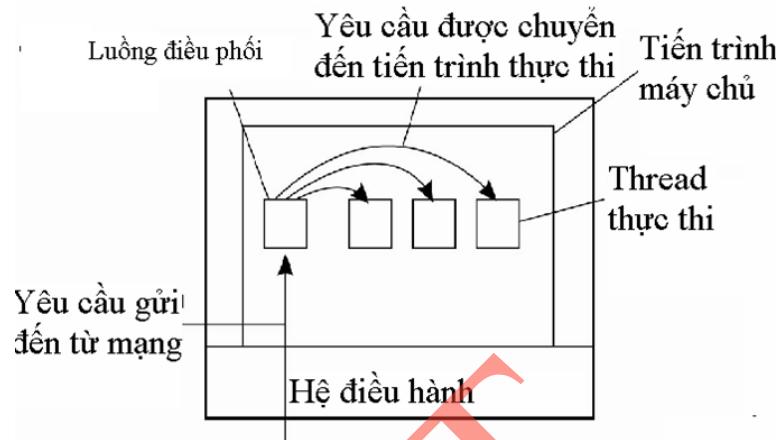
Đa luồng trên máy khách:

Trên mạng diện rộng, thông tin di chuyển từ máy tính này sang máy tính khác sẽ mất một khoảng thời gian nhất định, đôi khi cũng có thể lên tới vài giây. Để đảm bảo tính trong suốt phân bố tài nguyên thì các ứng dụng phân tán cần phải che giấu thời gian lan truyền của các thông điệp. Cách thông thường để che giấu độ trễ truyền tin là khởi tạo kênh truyền sau đó thực hiện một thao tác nào đó. Ví dụ, khi duyệt một trang tin điện tử, trình duyệt sử dụng giao thức HTML để lấy dữ liệu từ máy chủ và máy khách. Nếu hiển thị thông tin cho người dùng sau khi đã tải toàn bộ dữ liệu của trang thì thời gian chờ đợi sẽ tương đối dài, để giải quyết vấn đề này, sau khi lấy được khung dữ liệu của toàn bộ trang, máy khách sẽ tạo ra nhiều luồng riêng biệt, mỗi luồng sẽ lấy dữ liệu một phần của trang, nhận được dữ liệu của phần nào thì hiển thị ngay thông tin của phần đó. Như vậy, mỗi luồng trên máy khách sẽ thiết lập một kết nối mới gửi tới máy chủ, điều này làm tăng số lượng yêu cầu lên máy chủ và khi số lượng yêu cầu vượt quá khả năng xử lý thì các yêu cầu này sẽ được đưa vào hàng đợi chờ xử lý, như vậy hiệu năng của hệ thống sẽ không được cải thiện. Trong nhiều trường hợp, máy chủ được tổ chức thành từng cụm, mỗi yêu cầu của máy khách không phải chỉ xử lý trên một máy chủ mà có thể sẽ được chuyển đến máy khác xử lý, do đó đòi hỏi phải triển khai kỹ thuật xử lý song song trên cả máy khách lẫn máy chủ.

Đa luồng trên máy chủ:

Kỹ thuật xử lý đa luồng trên máy khách đóng vai trò rất quan trọng nhưng thực tế cho thấy việc triển khai kỹ thuật này trên máy chủ mới là nhân tố quyết định hiệu năng của hệ thống phân tán. Kỹ thuật xử lý đa luồng không những đơn giản hóa đáng kể cách viết các phần mềm mà còn làm cho việc triển khai kỹ thuật xử lý song song để đạt được hiệu năng cao nhất, dù cho đó là máy tính một hay nhiều bộ vi xử lý. Ví dụ về hệ thống cung cấp dịch vụ truyền tập tin, tập tin được lưu trữ trên ổ đĩa và quá trình đọc các khối dữ liệu trên ổ đĩa chiếm thời gian đáng kể so với các thao tác khác trong phiên làm việc. Bằng cách áp dụng kỹ thuật xử lý đa luồng, máy chủ sẽ tạo ra một

luồng chuyên để tiếp nhận các yêu cầu của máy khách (dispatcher) và nhiều luồng khác có nhiệm vụ đọc dữ liệu trên ổ đĩa (worker). Khi nhận yêu cầu đọc tập tin, thành phần dispatcher sẽ tiếp nhận yêu cầu và quyết định giao cho worker nào đó đảm nhiệm sau đó lại tiếp tục trở về trạng thái sẵn sàng tiếp nhận các yêu cầu khác từ máy khách. Nhận được yêu cầu đọc dữ liệu, một luồng worker mới sẽ được khởi tạo bằng việc phong tỏa tập tin và chờ cho đến khi hoàn thành quá trình đọc, sau đó một luồng khác (có thể vẫn là dispatcher) sẽ tiếp nhận kết quả và trả về cho máy khách.



Hình 2.34 Máy chủ đa luồng trong mô hình tiếp nhận/xử lý

Nếu chỉ sử dụng một luồng xử lý, vòng lặp của chương trình chính sẽ tiếp nhận và hoàn thành việc thực thi nhiệm vụ trước khi tiếp nhận yêu cầu kế tiếp, như vậy trong khi chờ đọc dữ liệu từ ổ đĩa máy chủ tuy ở trạng thái nghỉ nhưng vẫn không tiếp nhận thêm yêu cầu nào, **kết quả là nhiều yêu cầu của máy khách sẽ không được xử lý**. Như vậy có thể thấy việc áp dụng kỹ thuật đa luồng đã làm tăng đáng kể hiệu năng xử lý nhưng mỗi luồng vẫn được lập trình theo phương pháp thông thường. Một giải pháp khác chỉ áp dụng kỹ thuật đơn luồng nhưng vẫn có khả năng cho hiệu năng cao đó là phương pháp máy trạng thái hữu hạn. Khi có một yêu cầu gửi đến, một tiến trình duy nhất sẽ kiểm tra xem dữ liệu trong vùng nhớ cache có đáp ứng hay không, nếu đáp ứng được yêu cầu thì sẽ trả về cho máy khách, nếu không đáp ứng sẽ phải đọc từ ổ đĩa. Tuy nhiên thay vì việc phong tỏa, nó thiết lập trạng thái của yêu cầu và lưu vào bảng quản lý yêu cầu và tiếp tục xử lý các thông điệp khác chuyển đến. Nếu thông điệp mới chuyển đến là yêu cầu từ phía máy khách thì sẽ tạo một công việc mới, nếu đó là kết quả yêu cầu đọc ổ đĩa từ thao tác trước thì sẽ lấy lại thông tin của yêu cầu tương ứng và trả về cho máy khách. Với phương pháp này, máy chủ sẽ không cần phải thực hiện các thao tác phong tỏa khi gửi và nhận thông điệp và đồng thời loại bỏ được đặc điểm xử lý tuần tự của hai phương pháp trước, trạng thái tính toán được lưu tường minh khi gửi hoặc nhận mỗi thông điệp, tiến trình vận hành như máy trạng thái hữu hạn, nó lấy sự kiện và phản hồi tùy thuộc vào cái gì chứa trong tiến trình đó.

Mô hình	Đặc điểm
Đa luồng	Song song, phong tỏa hệ thống
Đơn luồng	Tuần tự, phong tỏa hệ thống
Máy trạng thái	Song song, không phong tỏa hệ thống

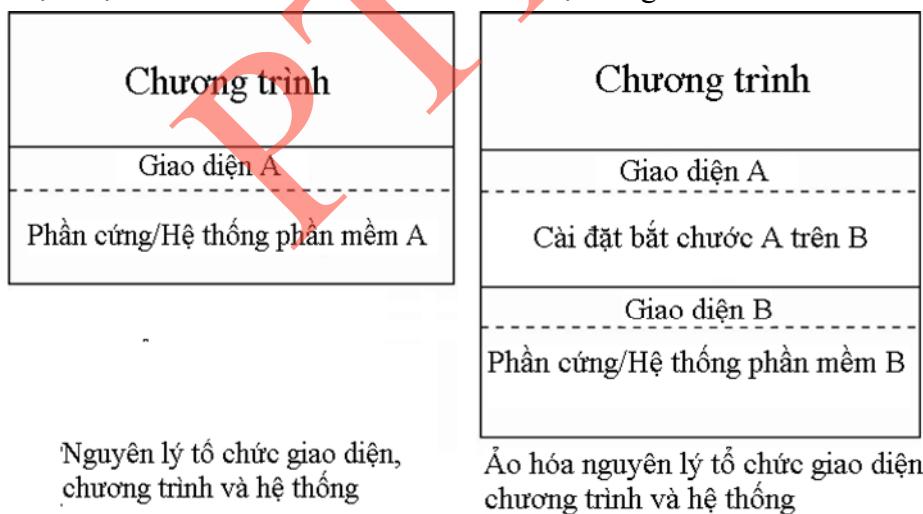
Tóm lại, trên máy chủ có thể áp dụng ba kỹ thuật xử lý đa luồng, đơn luồng hoặc máy trạng thái hữu hạn. Kỹ thuật đa luồng cho phép xử lý song song nhưng vẫn sử dụng phương pháp gọi phong tỏa hệ thống. Kỹ thuật đơn luồng không những phong tỏa hệ thống mà còn không cho phép xử lý song song, đôi khi sẽ làm giảm hiệu năng của hệ thống. Kỹ thuật máy trạng thái hữu hạn cho phép xử lý song song và không phong tỏa hệ thống, tuy nhiên việc lập trình sẽ phức tạp hơn rất nhiều.

2.4.2 Áo hóa

Đa luồng và đa tiến trình có thể được nhìn nhận như một cách thực hiện nhiều việc tại một thời điểm, kết quả là chúng ta có thể xây dựng các chương trình có cảm giác như đang chạy đồng thời. Nếu máy tính chỉ có một bộ vi xử lý thì việc chạy đồng thời nhiều chương trình chỉ là ảo giác, bởi vì tại 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 tưởng các chương trình chạy song song với nhau. Ý tưởng áo hóa xuất phát từ nhu cầu thực tế, tài nguyên vật lý chỉ có một nhưng có nhiều thành phần cùng sử dụng chung tài nguyên đó và thành phần nào cũng muốn tài nguyên đó thuộc về mình. Kỹ thuật này đã được áp dụng trong nhiều năm và ngày càng thu hút sự quan tâm của nhiều người.

2.4.2.1 Vai trò áo hóa trong các hệ thống phân tán

Các hệ thống máy tính thường được tổ chức phân tầng, tầng thấp hơn sẽ cung cấp giao diện lập trình cho các tầng mức cao hơn. Có nhiều loại giao diện, thấp nhất là giao diện với bộ xử lý trung tâm cho đến thư viện giao diện lập trình ứng dụng do các hệ thống trung gian cung cấp. Sự áo 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.



Hình 2.35 Áo hóa trong hệ thống phân tán

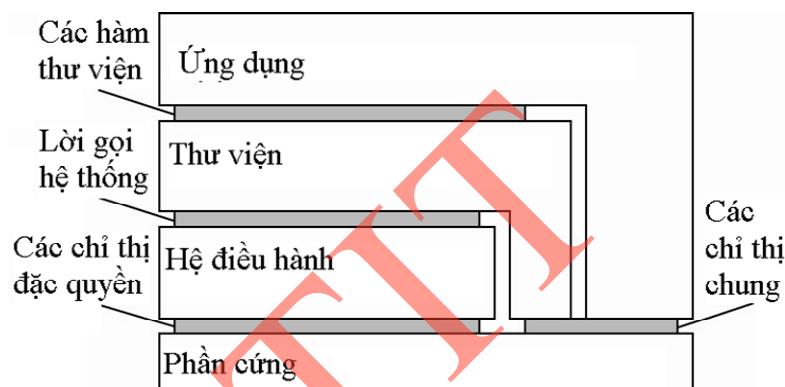
Những năm 70 của thế kỷ trước, giá thành của máy tính rất đắt mà số lượng phần mềm chưa nhiều và các hệ điều hành thường phát triển cho một loại phần mềm nào đó. Ý tưởng áo hóa được IBM triển khai nhằm tiết kiệm chi phí và cho phép các hệ thống phần mềm cũ có thể chạy trên máy chủ thế hệ mới. Khi giá thành phần cứng giảm dần, công nghệ phần cứng thay đổi nhanh hơn các ứng dụng phần mềm, vai trò áo hóa chuyển dần sang nhiệm vụ đảm bảo tính tương thích với các hệ thống cũ. Đặc

biệt, với sự bùng nổ của mạng Internet, nhiều hệ thống phần mềm lớn được xây dựng trên các nền tảng hệ thống khác nhau, việc ảo hóa nhằm tạo điều kiện cho các hệ thống dễ dàng tương tác với nhau.

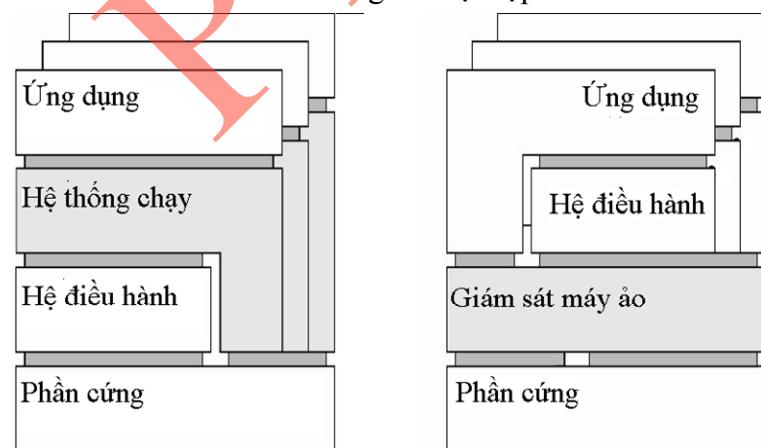
2.4.2.2 Kiến trúc của các máy ảo

Có nhiều cách triển khai ảo hóa, trước hết cần phải hiểu bốn loại giao diện sau:

- Giao diện giữa phần cứng và phần mềm chứa các chỉ thị lệnh mà bất kỳ chương trình nào cũng có thể gọi.
- Giao diện giữa phần cứng và phần mềm chứa các chỉ thị lệnh mà chỉ một số chương trình (ví dụ Hệ điều hành) mới được phép gọi.
- Giao diện lập trình hệ thống bao gồm các hàm do hệ điều hành cung cấp
- Giao diện lập trình ứng dụng bao gồm các hàm được đóng gói trong các thư viện (API)



Hình 2.36 Các giao diện lập trình



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 2.37 Hai mức ảo hóa

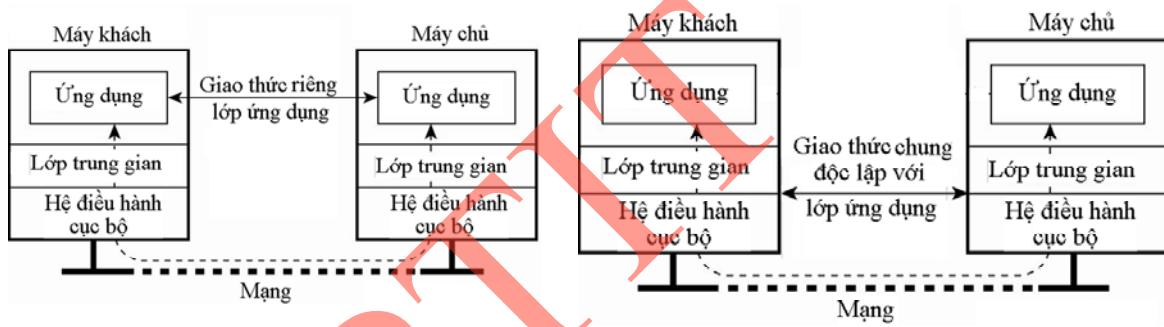
Có thể cài đặt ảo hóa bằng hai hình thức khác nhau: Máy ảo tiến trình (Process Virtual Machine) và Lớp máy ảo (Virtual Machine Monitor). Hình thức thứ nhất xây dựng hệ thống thời gian chạy cung cấp tập lệnh trùu tượng cho phần mềm ứng dụng,

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 ảo hóa này chủ yếu áp dụng cho một tiến trình. 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. 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, hình thức này ngày càng trở nên quan trọng trong việc giải quyết vấn đề tin cậy và bảo mật các hệ thống phân tán.

2.4.3 Máy khách

Nhiệm vụ chính của máy khách là cung cấp giao diện cho người dùng để tương tác với máy chủ. Hình thức thể hiện giao diện với người sử dụng đóng vai trò quan trọng trong việc phát triển các ứng dụng. Ngoài ra, tùy theo cách tổ chức xử lý, máy khách có thể thực hiện nhiệm vụ tiền xử lý các dữ liệu trước khi chuyển yêu cầu đến máy chủ.

2.4.3.1 Các giao diện người dùng mạng

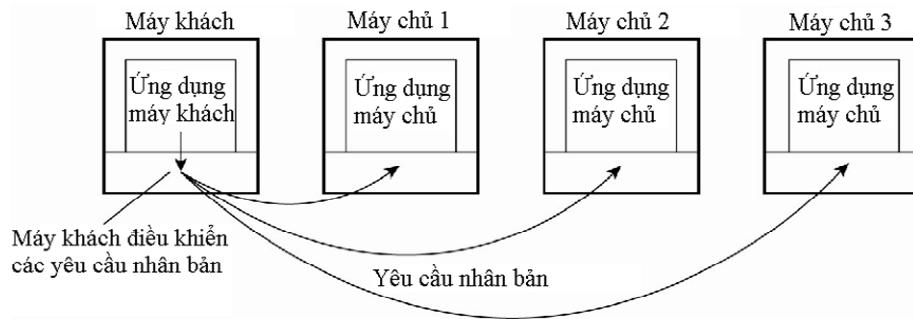


Hình 2.38 Giao thức lớp ứng dụng

Nhìn chung, tương tác giữa máy khách và máy chủ được thực hiện theo hai cách. Cách thứ nhất, với mỗi dịch vụ trên máy chủ sẽ có thành phần tương ứng trên máy khách, như vậy máy khách sử dụng giao thức riêng phục vụ cho đồng bộ dữ liệu với máy chủ. Cách thứ hai, máy khách không lưu trữ dữ liệu mà sử dụng giao thức chung để lấy dữ liệu từ máy chủ và hiển thị cho người dùng.

2.4.3.2 Tính trong suốt phân bố tài nguyên

Phần mềm máy khách không những bao gồm các chức năng giao tiếp người dùng mà còn có các thành phần khác thực hiện nhiều nhiệm vụ khác như: xử lý dữ liệu, truy nhập tài nguyên trên máy chủ..., những thành phần này càng trong suốt đối với người dùng càng tốt, trái ngược hẳn với quan điểm xây dựng các ứng dụng trên máy chủ. Nói chung, tính trong suốt truy nhập được xử lý bằng cách tạo xây dựng thành phần (Stub) có giao diện như trên máy chủ nhưng che giấu kiến trúc máy và cách trao đổi thông tin. Vấn đề trong suốt phân bố tài nguyên thường được xử lý bằng hệ thống đặt tên, tuy nhiên trong những trường hợp máy khách đã kết nối tới máy chủ nào đó thì máy chủ có thể gửi thông tin di trú đến lớp trung gian nằm trên máy khách để thực hiện kết nối đến máy chủ mới, điều này có thể tạm thời làm suy giảm hiệu năng của máy khách.



Hình 2.39 Tính trong suốt nhân bản máy chủ

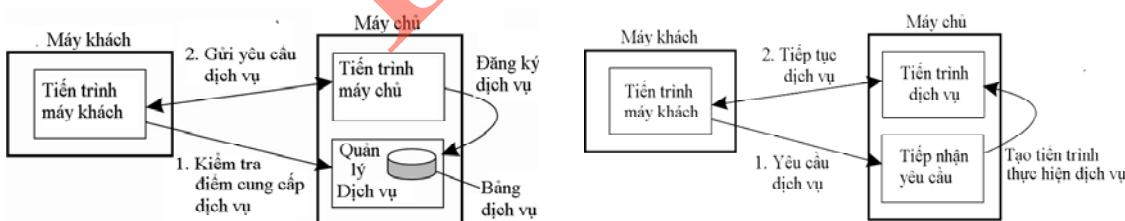
Nếu máy chủ được nhân bản, máy khách có thể gửi yêu cầu đến tất cả máy chủ nhưng sẽ chỉ nhận kết quả trả về từ 01 máy. Để đảm bảo tính trong suốt về lỗi, đối với lỗi truyền thông thì phần mềm trung gian trên máy khách có thể gửi lại một vài lần hoặc gửi yêu cầu đến máy chủ khác xử lý, trường hợp xấu nhất có thể lấy dữ liệu đã ghi nhớ (cache) của phiên liền trước. Tính trong suốt tương tranh thường được giải quyết trên các máy chủ, ví dụ sử dụng hệ thống giám sát tương tranh, máy khách ít khi phải xử lý vấn đề này.

2.4.4 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.

2.4.4.1 Các vấn đề thiết kế chung

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.



Hình 2.40 Cài đặt mô hình khách/chủ

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.

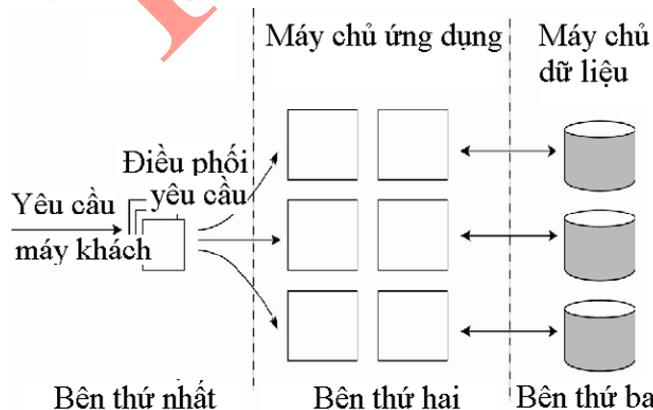
Tuy nhiên, một số dịch vụ không sử dụng đến cổng (ví dụ dịch vụ cung cấp thời gian), trong trường hợp này máy chủ phải gán điểm truy nhập dịch vụ động và gán cho mỗi hệ điều hành và đồng thời phải có một tiến trình đặc biệt (gọi là daemon) luôn theo dõi điểm truy nhập dịch vụ này. Thông thường mỗi điểm 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. Ví dụ trường hợp máy khách đột ngột hủy trong khi máy chủ đang thực thi yêu cầu, khi đó kết nối giữa máy khách và máy chủ sẽ bị hủy bỏ, máy chủ sẽ chờ một khoảng thời gian nhất định sau đó sẽ hủy bỏ liên kết. Tuy nhiên, giải pháp này sẽ không tốt trong trường hợp liên kết giữa máy khách và máy chủ bị gián đoạn do nguyên nhân khách quan (chất lượng đường truyền kém...), giải pháp tốt hơn sẽ là cung cấp báo hiệu khen ngoài (out-band) cho liên kết giữa máy khách và máy chủ hoặc thiết kế giao thức có khả năng điều khiển tương tác.

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.

2.4.4.2 Cụm máy chủ

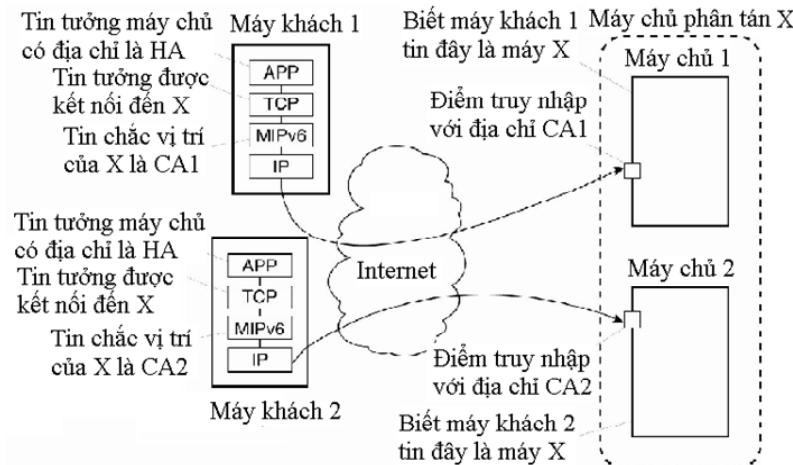
Cụm máy chủ là ~~một~~ tập hợp các máy được kết nối qua mạng, trên mỗi máy chạy một hoặc nhiều tiến trình máy chủ. Thông thường cụm máy chủ được kết nối trong mạng nội bộ để bảo đảm băng thông đủ lớn để giảm thiểu độ trễ khi trao đổi thông tin giữa các máy chủ trong cụm.



Hình 2.41 Tổ chức cụm máy chủ ba bên

Thông thường, cụm máy chủ được bố trí thành ba lớp: Lớp tiếp nhận yêu cầu, lớp xử lý nghiệp vụ và lớp lưu trữ dữ liệu. Việc phân cụm máy chủ không những làm tăng hiệu năng xử lý mà còn đảm bảo độ tin cậy cho hệ thống. Ví dụ tại lớp xử lý yêu

cần nghiệp vụ thì cần phải có những bộ vi xử lý tốc độ cao nhưng lớp lưu trữ dữ liệu thường đòi hỏi yêu cầu các bộ đọc ghi nhanh.



Hình 2.42 Tối ưu hóa định tuyến máy chủ

Điểm yếu cơ bản của mô hình cụm máy chủ nằm ở lớp tiếp nhận yêu cầu, nếu lớp này ngừng hoặc bị quá tải thì sẽ ảnh hưởng đến toàn bộ hệ thống. Một mô hình khác đưa ra là tổ chức các máy chủ phân tán. Thay vì chỉ có một thành phần tiếp nhận yêu cầu sẽ bô chí nhiều thành phần tiếp nhận yêu cầu, đây là mô hình đã áp dụng trong hệ thống dịch vụ tên miền (DNS). Mô hình này đòi hỏi phải cài đặt thêm bộ tối ưu hóa định tuyến nhằm tạo điều kiện cho các máy khách tìm được máy chủ dịch vụ một cách nhanh nhất. Ngoài ra, máy khách cũng phải được cấu hình để biết thông tin về các máy chủ tiếp nhận yêu cầu.

2.4.4.3 Quản lý cụm máy chủ

Người sử dụng luôn mong muốn công việc quản lý cụm máy chủ phải tương tự như trên một máy tính. Thực tế cho thấy các thao tác quản lý cụm máy chủ phức tạp hơn nhiều, cách đơn giản nhất là mở rộng tính năng quản lý của một máy chủ, người sử dụng phải truy nhập từ xa vào mỗi máy và thực hiện các thao tác quản lý như trên một máy. Một giải pháp khác là xây dựng phần mềm cài đặt trên một máy quản trị, mỗi máy tính trong cụm được thể hiện bằng một nút và người dùng có thể thêm hoặc bớt bất kỳ máy chủ nào. Như vậy, thay vì phải truy nhập vào từng máy thì phần mềm quản lý sẽ thu thập thông tin từ mỗi máy chủ và cung cấp giao diện quản lý cho người sử dụng. Tuy nhiên, giải pháp này chỉ phù hợp với mô hình nhỏ, vẫn đề sẽ phức tạp hơn rất nhiều đối với cụm máy chủ có quy mô lớn, vì vậy giải pháp quản lý cụm máy chủ lớn vẫn đang tiếp tục nghiên cứu.

2.4.5 Di trú mã

Từ đầu chương trình chúng ta mới chỉ tập trung vào vấn đề trao đổi dữ liệu trong hệ thống phân tán. 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.

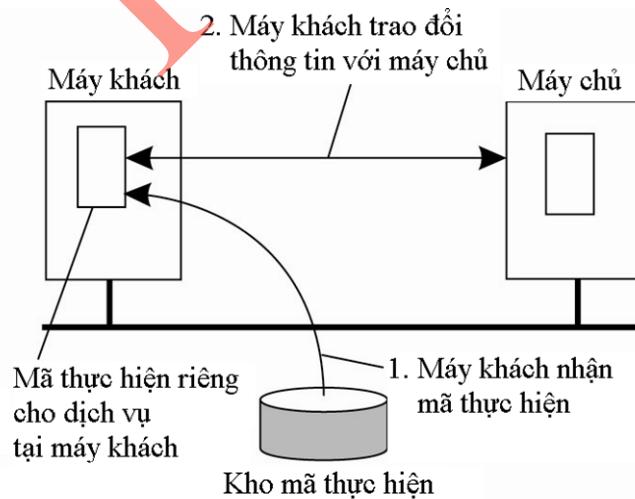
2.4.5.1 Các giải pháp di trú mã

Thông thường, di trú mã trong các hệ thống phân tán thực hiện dưới dạng di trú tiến trình, trong đó toàn bộ tiến trình được di chuyển từ máy này sang máy khác, đây

là công việc đòi hỏi chi phí cao và phức tạp. Lý do phải di chuyển tiến trình vẫn là vấn đề hiệu năng của hệ thống, các yêu cầu xử lý cần phải được chuyển tới những máy ít tải hơn, tải ở đây được hiểu là tỉ lệ sử dụng CPU, RAM và một số yếu tố khác. Các thuật toán phân tải đưa ra quyết định liên quan đến vấn đề xác định và phân phối lại nhiệm vụ dựa trên số lượng bộ xử lý đóng vai trò quan trọng trong các hệ thống tính toán với tần suất lớn. Tuy nhiên, trong các hệ thống phân tán hiện đại, vấn đề truyền dữ liệu trên được coi trọng hơn việc tối ưu hóa khả năng tính toán. Ngoài ra, vì các lý do liên quan đến vấn đề nền tảng không thống nhất hoặc mạng máy tính mà quyết định di trú mã có tính chất định tính chứ không dựa trên các mô hình toán học.

Ví dụ, nếu máy khách cần phải tải một lượng dữ liệu lớn từ máy chủ về để xử lý thì có thể di trú đoạn mã trên máy khách về máy chủ và khi đó máy khách chỉ nhận kết quả đã được tính toán trên tập dữ liệu máy khách yêu cầu. Ngược lại, đối với những trường hợp người dùng phải cung cấp thông tin để máy chủ xử lý, thao tác kiểm tra tính hợp lệ của thông tin đầu vào nên được thực hiện trên máy khách, như vậy không những giảm tải cho máy chủ mà còn giảm lượng thông tin lưu chuyển trên mạng.

Di trú mã cũng cần thiết trong các trường hợp xử lý song song, nhưng không phải trên một máy tính. Ví dụ, để thực hiện nhiệm vụ tìm kiếm thông tin trên mạng, người ta sử dụng một đoạn mã di trú (gọi là Agent), đoạn mã này sẽ di chuyển từ trang này sang trang khác, khi dừng lại ở trang nào nó sẽ tiếp tục nhận bản đoạn mã đó. Ngoài ra, kỹ thuật di trú mã còn làm cho hệ thống phân tán linh hoạt hơn, ví dụ mô hình đa phương trong các ứng dụng khách/chủ. Di trú mã giữa các máy đòi hỏi hệ thống phân tán phải được cấu hình động. Ví dụ, giả sử máy chủ cung cấp giao diện truy nhập tập tin, nó phải xây dựng một giao thức riêng. Thông thường, máy khách sẽ dựa trên giao thức này để liên kết với các ứng dụng khác. Đây là giải pháp tĩnh, nó đòi hỏi phải xây dựng giao thức trước khi phát triển phần mềm trên máy khách. Có thể sử dụng giải pháp động bằng cách đặt các đoạn mã vào một kho lưu trữ, máy khách sẽ tải một đoạn mã cần thiết từ kho lưu trữ đó, sau một số bước khởi tạo sẽ kết nối đến máy chủ cung cấp dịch vụ.



Hình 2.43 Nguyên lý cấu hình động cho máy khách

Một tiến trình bao gồm phần mã chứa tập các lệnh của chương trình đang chạy, phần tài nguyên chứa các tham chiếu đến tất cả các tài nguyên bên ngoài mà tiến trình

đang sử dụng và phần thực thi chứa các trạng thái hiện hành của tiến trình. Việc di trú mã được hiểu là di chuyển một phần hay toàn bộ tiến trình, người ta phân làm hai loại di trú yếu (Weak mobility) và di trú mạnh (Strong mobility). Di trú yếu chỉ truyền phần mã và một số các dữ liệu khởi động của tiến trình, một chương trình được truyền đi luôn được bắt đầu từ trạng thái khởi động và chỉ yêu cầu máy đích thực thi yêu cầu đó. Di trú mạnh truyền cả phần mã và phần thực thi, tiến trình đang chạy có thể tạm dừng để chuyển đến một máy khác và tiếp tục thực hiện tiến trình đó, mô hình này khó thực hiện hơn. Di trú được do gửi khởi sướng (Sender initiated migration) hoặc do bên nhận khởi sướng (Receiver initiated migration).

2.4.5.2 Di trú và tài nguyên cục bộ

Mô hình di trú mạnh đòi hỏi chú ý đặc biệt tới những tài nguyên tiến trình đang sử dụng. Ví dụ một tiến trình đang sử dụng một cổng nào đó, khi di trú sang máy khác đồng nghĩa với việc từ bỏ cổng đang sử dụng và đồng thời thiết lập cổng tương ứng trên máy mới. Có ba mức tham chiếu giữa tiến trình và tài nguyên: Mạnh, yếu và rất yếu. Ở mức thứ nhất, tiến trình tham chiếu tới tài nguyên thông qua định danh, ví dụ các tiến trình sử dụng đường dẫn URL. Mức thứ hai, tiến trình chỉ đòi hỏi giá trị của tài nguyên, ví dụ tiến trình sử dụng các thư viện lập trình. Mức thứ ba, tiến trình chỉ tham chiếu đến loại tài nguyên, ví dụ tham chiếu tới các thiết bị ngoại vi. Khi di trú mã thì chỉ có thể thay đổi tham chiếu đến tài nguyên mà không được phép thay đổi mức độ tham chiếu.

2.4.5.3 Di trú trong hệ thống không đồng nhất

Di trú mã trên các hệ thống đồng nhất ~~sẽ~~ không gặp những trở ngại lớn là do các máy tính cùng chạy trên một nền tảng phần cứng và hệ điều hành, vẫn đề ~~sẽ~~ trở nên khó khăn hơn rất nhiều nếu ~~hệ thống~~ bao gồm các máy tính sử dụng nền tảng khác nhau. Ngoài ra, vẫn đề ~~ngôn ngữ lập trình~~ cũng gây trở ngại lớn trong di trú mã, các ứng dụng thường được lập trình bằng các ngôn ngữ lập trình bậc cao như Pascal, C hay Java... Để giảm thiểu sự phụ thuộc ngôn ngữ lập trình, một giải pháp đã được đề xuất đó là việc di trú ~~không~~ chỉ thực hiện với tiến trình mà thực hiện với toàn bộ môi trường tính toán. Với đề xuất này, việc di trú phải thực hiện ba bước:

- Bước 1: Chuyển các trang bộ nhớ sang máy mới và sẽ cập nhật lại những trang bị thay đổi trong quá trình di trú.
- Bước 2: Ngừng máy ảo, di trú bộ nhớ và khởi tạo máy ảo mới.
- Bước 3: Khởi tạo các tiến trình trên máy ảo mới và sao chép các trang bộ nhớ theo yêu cầu của tiến trình đó.

2.5 Quản trị giao tác và điều khiển tương tranh

Giao tác phân tán là một đơn vị chương trình được thực hiện nhằm mục đích truy xuất dữ liệu tại một hoặc nhiều vị trí khác nhau. Điều khiển 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. Tương tranh thường dẫn đến mất mát khi cập nhật và kết quả không đồng nhất. Ví dụ Tài khoản của A, B và 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, qui trình thực hiện sau thể hiện lỗi mất mát cập nhật.

A chuyển cho B	C chuyển cho B
<pre>balance = b.getBalance(); b.setBalance(balance * 1.1); a.withdraw(balance / 10)</pre>	<pre>balance = b.getBalance(); b.setBalance(balance * 1.1); c.withdraw(balance / 10)</pre>
<pre>balance = b.getBalance(); \$200</pre> <pre>b.setBalance(balance * 1.1) \$220</pre> <pre>a.withdraw(balance / 10) \$80</pre>	<pre>balance = b.getBalance(); \$200</pre> <pre>b.setBalance(balance * 1.1) \$220</pre> <pre>c.withdraw(balance / 10) \$280</pre>
Số tiền trong tài khoản của B lẽ ra phải là \$242	

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

A chuyển \$100 cho B	Kiểm tra số dư tài khoản
<pre>a.withdraw(100)</pre>	<pre>Kiểm tra số dư tài khoản</pre>
<pre>b.deposit(100)</pre>	<pre>aBranch.branchTotal()</pre>
<pre>a.withdraw(100); \$100</pre>	<pre>total = a.getBalance() \$00</pre>
<pre>b.deposit(100) \$300</pre>	<pre>total = total + b.getBalance() \$200</pre>
	<pre>total = total + c.getBalance()</pre>
	<pre>:</pre>

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. Hai thao tác được gọi là xung đột nếu kết quả tổ hợp thực hiện của chúng phụ thuộc vào thứ tự thực hiện, do đó cần phải xác định các thao tác có thể xảy ra xung đột và tất cả các cặp thao tác xung đột của các giao tác cần phải được thực hiện theo một thứ tự để đảm bảo tính tuần tự.

2.5.1 Các giao tác

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 (thao tác) đọc hoặc ghi nhưng phải đảm bảo tính tin cậy và nhất quán. Bốn tính chất của giao tác đối với thế giới bên ngoài như sau:

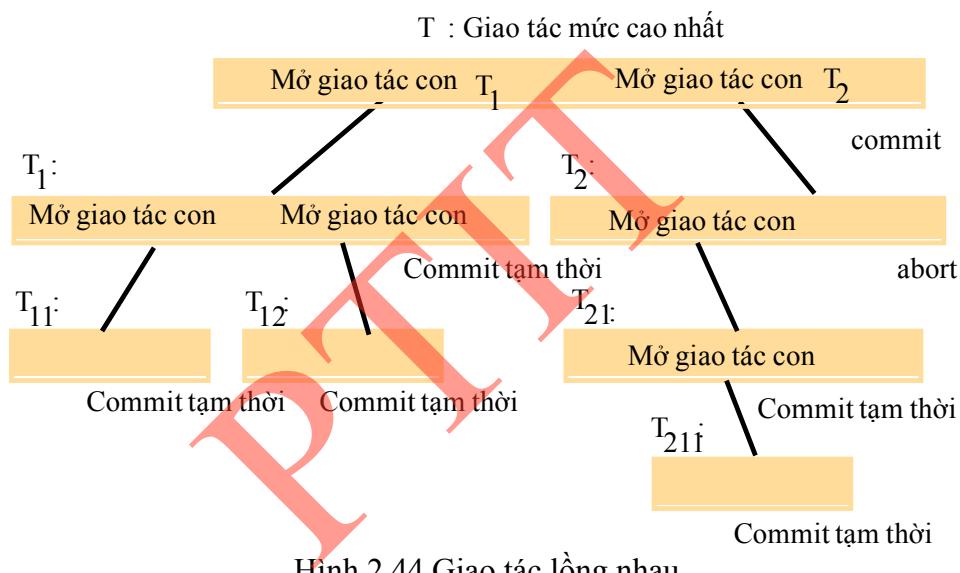
- Tính nguyên tử (Atomic): Đối với thế giới bên ngoài thì giao tác không thể chia nhỏ hơn được. Các lệnh trong giao tác đều được thực hiện hoặc không có lệnh nào được thực hiện. Tính nguyên tử đòi hỏi nếu một giao tác bị hủy giữa chừng thì các kết quả trước đó của nó phải được hủy bỏ.

- Tính 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 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 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 giao tác.

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 (aborted), nói cách khác mức độ của tính nguyên tố của giao tác phẳng còn yếu.

2.5.2 Các giao tác lồng nhau

Một giao tác lồng nhau có cấu trúc từ một số giao tác con, hay nói cách khác là trong giao tác lại bao gồm các giao tác khác. Mỗi giao tác con cũng có thể thực thi một hay nhiều giao tác con của chính nó.



Hình 2.44 Giao tác lồng nhau

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

2.5.3 Các khóa

Một phương pháp để đảm bảo tính tuần tự là yêu cầu việc truy xuất đến hạng mục dữ liệu được tiến hành theo kiểu loại trừ tương hỗ, nghĩa là trong khi một giao dịch đang truy xuất một mục dữ liệu thì không một giao tác nào khác có thể sửa đổi mục dữ liệu này. Phương pháp chung nhất được dùng để thực thi yêu cầu này là cho phép một giao tác truy xuất một mục dữ liệu khi và chỉ khi nó đang giữ khóa trên mục dữ liệu đó. Tư tưởng chính của các thuật toán này là các thao tác trên một đơn vị dữ liệu nếu có xung đột thì tại một thời điểm chỉ cho phép một giao tác thực hiện, điều này được thực hiện dựa trên cơ chế khóa.

Để truy xuất một mục dữ liệu, giao tác T_i đầu tiên phải khóa mục dữ liệu này và khi thực hiện xong thì phải giải phóng khóa. Nếu mục dữ liệu này đã bị khóa bởi một giao tác khác và khóa đó không tương thích thì bộ điều khiển tương tranh sẽ không cấp

khóa cho đến khi khóa này được giải phóng. Một giao tác cần thiết phải giữ một khóa trên một mục dữ liệu chừng nào mà nó còn truy xuất mục này. Hơn nữa, đối với một giao tác việc tháo khóa ngay sau khi truy xuất cuối cùng đến mục dữ liệu không luôn luôn là điều mong muốn vì như vậy tính khả thi tự có thể không được đảm bảo cho đến khi tất cả các khóa không tương thích do các giao tác khác giải phóng.

Sử dụng khóa có thể dẫn đến trạng thái khóa chết (*deadlock*), đó 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. Để hạn chế hiện tượng khóa chết có thể áp dụng các biện pháp sau:

- 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

2.5.4 Điều khiển tương tranh tối ưu

Có hai cách tiếp cận điều khiển tương tranh: Điều khiển bi quan và điều khiển lạc quan. Hai cách tiếp cận này trái ngược hẳn với nhau về mặt quan điểm, điều khiển tương tranh bi quan nhìn nhận hệ thống luôn tiềm ẩn tương tranh, trong khi đó điều khiển tương tranh lạc quan nhìn nhận không xảy ra tương tranh và khi nào xảy ra thì mới giải quyết. Cách sử dụng khóa là một cách tiếp cận điển hình về điều khiển tương tranh bi quan, nhược điểm cơ bản của nó là tăng tải xử lý cho hệ thống.

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

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 và không tái sử dụng số thứ tự đã gán cho giao tác, như vậy không cần kiểm tra xung đột ghi-ghi. Điều khiển tương tranh lạc quan sử dụng hai dạng phê chuẩn: Phê chuẩn ngược và phê chuẩn xuôi. Phê chuẩn ngược kiểm tra với các giao tác đã bước vào giai đoạn phê chuẩn trước nó trong khi đó phê chuẩn xuôi lại kiểm tra với các giao tác sau nhưng vẫn đang hoạt động.

2.5.5 Trình tự nhãn thời gian

Một cách tiếp cận khác trong điều khiển tương tranh là gán nhãn thời gian cho các giao tác. 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. 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 đảm bảo những tác có thời gian thực hiện nhau và thường xuyên sẽ được ưu tiên hơn các giao tác khác.

2.6 Phục hồi và chịu lỗi

Lỗi từng phần là tính chất đặc trưng của hệ thống phân tán so với các hệ thống chỉ có một máy chủ, lỗi đó có thể xảy ra khi một thành phần nào đó của hệ thống phân tán bị hỏng. Trong khi lỗi của hệ thống một máy chủ sẽ ảnh hưởng đến toàn bộ hệ thống thì lỗi trong hệ thống phân tán sẽ chỉ ảnh hưởng tới một số thành phần, ít khi ảnh hưởng đến toàn bộ hệ thống. Một trong những mục tiêu quan trọng khi thiết kế hệ thống phân tán là xây dựng một hệ thống sao cho có thể tự động phục hồi khi gặp lỗi mà ít ảnh hưởng đến sự vận hành chung của hệ thống.

2.6.1 Giới thiệu tính chịu lỗi

Trong hệ thống phân tán, lỗi có thể xảy ra trên mỗi máy tính (đặc biệt là các máy chủ và hạ tầng mạng) và là nguyên nhân gây nên gián đoạn dịch vụ hệ thống cho một số máy tính khác hoặc thậm chí cho tất cả các máy tính trong hệ thống. Một yêu cầu quan trọng khi xây dựng hệ thống phân tán là phải lường trước được các lỗi có thể xảy ra và biện pháp xử lý sao cho tối thiểu hóa các ảnh hưởng của nó đến hệ thống. Nói cách khác, khi có lỗi xảy ra thì hệ thống vẫn vận hành theo cách có thể chấp nhận được, nghĩa là hệ thống phải có khả năng chịu được lỗi. Chủ đề về tính chịu lỗi của hệ thống phân tán đã được nghiên cứu khá nhiều trong **khoa học máy tính**.

2.6.1.1 Một số khái niệm cơ bản

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

Khả năng sẵn sàng phục vụ của hệ thống thể hiện ở thời gian đáp ứng yêu cầu của mỗi dịch vụ, trong trường hợp lý tưởng nó phải được thực hiện theo thời gian thực. Độ tin cậy của hệ thống liên quan tới tính chính xác của thông tin mà không phụ thuộc vào yếu tố thời gian. Độ an toàn của hệ thống thể hiện ở khả năng bảo đảm an toàn cho dữ liệu ngay cả khi có những sự cố lớn. Khả năng bảo trì hệ thống thể hiện ở khả năng phục hồi hệ thống sau khi xảy ra lỗi, nó phải được thực hiện một cách đơn giản và trong thời gian nhanh nhất và tối thiểu sự tổn hại thông tin.

2.6.1.2 Các mô hình lỗi

Một hệ thống bị coi là lỗi nếu nó không cung cấp chính xác các dịch vụ như đã thiết kế. Nếu coi hệ thống phân tán là tập hợp máy chủ trao đổi thông tin với nhau và với các máy khách thì lỗi có thể xảy ra ở trên mỗi máy chủ đó hoặc trên hạ tầng mạng. Việc vận hành sai qui cách cũng có thể gây ra lỗi, nếu một máy chủ hoạt động phụ thuộc vào máy chủ khác thì nó cũng sẽ bị ảnh hưởng nếu lỗi xảy ra trên máy chủ khác. Xét về tần suất, lỗi được phân chia thành ba loại sau:

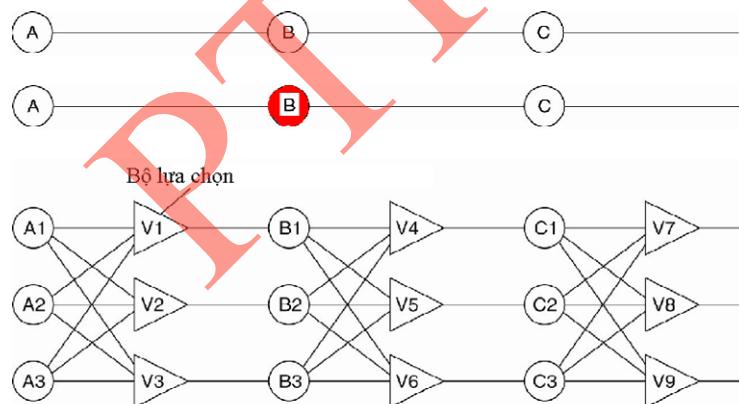
- **Lỗi nhất thời:** Là loại lỗi xảy ra một lần sau đó không xuất hiện lại.
- **Lỗi lặp:** Là loại lỗi mà chúng xuất hiện nhiều lần, có thể theo chu kỳ hoặc không. Lỗi này thường gây ra các hậu quả nghiêm trọng vì chúng rất khó xác định được.

Hệ thống phân tán

- Lỗi lâu dài: Là loại lỗi vẫn tồn tại ngay cả khi thành phần gây lỗi đó đã được sửa chữa.
Xét về mức độ, 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 trạm, nhận được yêu cầu nhưng không thể trả lời hoặc không thể trả về kết quả xử lý cho máy trạm. Trong trường hợp này cần phải kiểm tra kết nối giữa máy trạm và máy chủ, thường đó là các lỗi mạng.
 - Lỗi thời gian: Máy chủ không đáp ứng được thời gian xử lý yêu cầu của máy trạm.
 - Lỗi trả kết quả xử lý: Thông tin trả về của máy chủ không chính xác, có thể là giá trị sai.
 - Lỗi bất thường: 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.

2.6.1.3 Che giấu lỗi bằng biện pháp dư thừa

Một hệ thống chịu được lỗi thì cần phải có khả năng che giấu được các lỗi, nói cách khác người sử dụng không hề biết hoặc ít biết về sự cố đối với mạng dịch vụ. Một phương pháp phổ biến thường được áp dụng trong các hệ thống phân tán là xây dựng các thành phần dư thừa (bản sao của thành phần chính).



Hình 2.45 Xây dựng hệ thống dư thừa theo kiểu chuyển mạch khi có sự cố

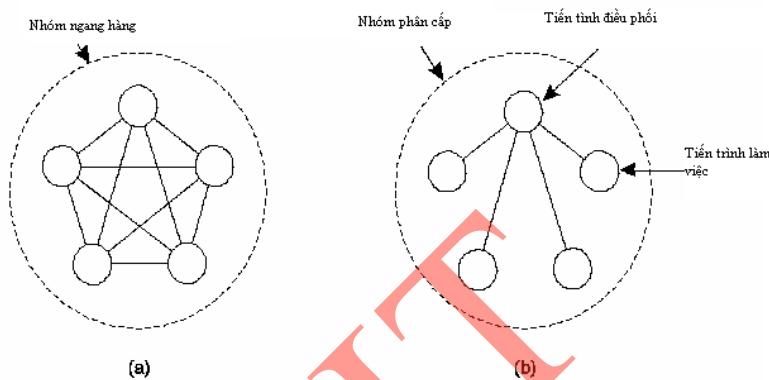
Hình 2.45 minh họa kỹ thuật dư thừa theo kiểu chuyển mạch. Bình thường hệ thống chỉ bao gồm 3 thành phần chính A, B, C. Để đảm bảo khả năng chịu lỗi, chúng ta thêm các thành phần tương ứng với A, B, C và duy trì ở trạng thái dự phòng. Khi có lỗi xảy ra, hệ thống sẽ tự động chuyển đến một trong các thành 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.

2.6.2 Tiết trình bền bỉ

Vấn đề cơ bản trong việc cung cấp tính năng chịu lỗi trong hệ thống phân tán là việc phòng chống và khắc phục lỗi. Có thể xây dựng hệ thống chịu lỗi bằng phương pháp dư thừa vật lý, tuy nhiên giải pháp này khá tốn kém, phần này sẽ đề cập tới một số giải pháp mềm để giải quyết vấn đề lỗi trong các hệ thống phân tán.

2.6.2.1 Những vấn đề thiết kế

Giải pháp chính trong việc chống lỗi là tổ chức một số tiết trình giống nhau thành các nhóm xử lý. Khi có thông điệp được gửi tới thì tất cả các thành viên trong nhóm đều nhận được thông điệp đó. Một tiết trình có thể là thành viên của một số nhóm khác nhau.



Hình 2.46 Nhóm các tiết trình trong hệ thống phân tán

Vấn đề mới này sinh đôi với loại thiết kế này là việc quản lý và phối hợp hoạt động giữa các tiết trình. Hình 2.46 minh họa hai giải pháp tổ chức nhóm tiết trình trong các hệ thống phân tán, việc tổ chức này hoàn toàn phụ thuộc vào các thuật toán xử lý bên trong mỗi tiết trình.

2.6.2.2 Che giấu lỗi và nhân bản

Việc che giấu lỗi trong hệ thống phân tán tập trung vào trường hợp có tiết trình bị lỗi. Truy nhiên trước hết cần phải xem xét các trường hợp lỗi truyền tin. Sử dụng nhóm tiết trình cũng là một trong các biện pháp che giấu lỗi, nhóm các tiết trình giống nhau sẽ che giấu một tiết trình nào đó bị lỗi. Nhân bản một tiết 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 tiết trình trong nhóm tổ chức theo mô hình phân cấp. Một tiết trình đóng vai trò tiết trình chính có nhiệm vụ điều phối tất cả các thao tác ghi. Nếu tiết trình chính của nhóm dừng hoạt động thì các tiết trình sao lưu sẽ thực hiện giải thuật bầu chọn để lựa chọn tiết trình chính mới. Nếu dựa trên ghi nhân bản thì các tiết 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 nhiêu.

2.6.2.3 Thỏa thuận trong các hệ thống lỗi

Vấn đề trắc nết 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 như các hệ thống đồng bộ đối với các hệ thống không đồng bộ, 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. Một câu hỏi khác cũng đặt ra là việc đảm bảo thứ tự thông điệp và chỉ có thể coi là tuân tự nếu thứ tự bên gửi và bên nhận giống nhau.

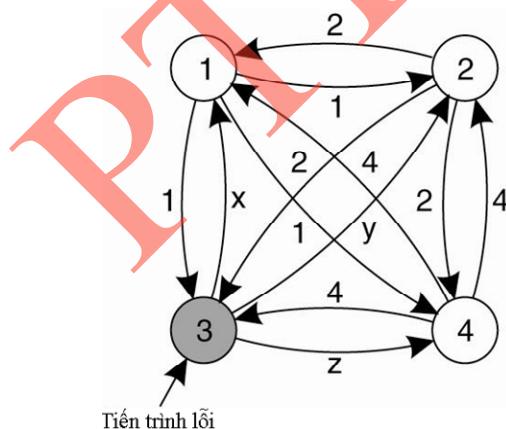
Giả sử hệ thống gồm N tiến trình, mỗi tiến trình i cung cấp giá trị V_i cho các tiến trình khác, mục tiêu cần đạt trong thỏa thuận lỗi sẽ được thực hiện bằng cách cho phép mỗi tiến trình xây dựng vector $V[N]$ với $V[i] = V_i$ nếu tiến trình i không lỗi và $V[i]$ không xác định nếu tiến trình i bị lỗi. Quá trình thỏa thuận sẽ được thực hiện theo bốn bước sau:

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.



Hình 2.47 Thỏa thuận Byzantine

Ví dụ, hệ thống gồm bốn thành viên như trên hình 2.47, giả sử tiến trình số 3 bị lỗi khi đó các thành viên khác trong hệ thống cần phải đảm bảo tiến trình 3 có thực sự bị lỗi hay không. Áp dụng các bước thực hiện trên, kết quả thực hiện ở bước 2 sẽ bao gồm $(1, 2, x, 4)$, $(1, 2, x, 4)$, $(1, 2, y, 4)$, $(1, 2, z, 4)$, tiếp tục thực hiện sẽ cho kết quả cuối cùng tại các tiến trình 1, 2 và 4 như sau: $(1, 2, \text{Unknown}, 4)$, $(1, 2, \text{Unknown}, 4)$, $(1, 2, \text{Unknown}, 4)$, vậy các tiến trình 1, 2 và 4 có thể đảm bảo chắc chắn tiến trình số 3 đã bị lỗi.

2.6.2.4 Phát hiện lỗi

Phát hiện lỗi là một trong những tính năng quan trọng hàng đầu trong khả năng chịu lỗi của hệ thống phân tán, nó đảm bảo khả năng phát hiện kịp thời những thành phần 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.

Chỉ cần tối thiểu hai cơ chế để phát hiện lỗi: Chủ động gửi thông điệp hỏi “IS ALIVE” hoặc thụ động nhận thông điệp đó từ tiền tình khác, trong thực tế thường dùng cơ chế thứ nhất (PING). Mỗi nút cần thường xuyên trao đổi thông tin với các nút có liên quan. Phải phân biệt được lỗi mạng hay lỗi trên mỗi nút mạng, bất kỳ thành viên nào phát hiện được lỗi lỗi thì phải thông báo tới các nút có liên quan.

2.6.3 Truyền thông khách/chủ tin cậy

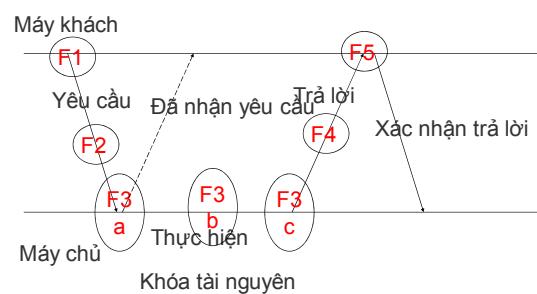
Tính chịu lỗi trong hệ thống phân tán không chỉ tập trung vào các lỗi ở các tiến trình mà còn phải chú ý đến các lỗi truyền thông. Lỗi truyền thông có thể do mất kênh truyền, thất lạc thông tin, quá thời gian, lặp thông điệp... Để khắc phục lỗi truyền thông, người ta thường sử dụng phương pháp truyền 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.

2.6.3.1 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 bỗn bề 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.

2.6.3.2 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:



Hình 2.48 Những khả năng lỗi trong phương pháp gọi thủ tục từ xa

- 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.

- 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"
- 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 cả. 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.
- 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 nhờ các số tự 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 ta 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 trạm bị lỗi ngay sau khi gửi yêu cầu tới máy chủ: Máy trạm 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 thế sẽ gây

lãng phí thời gian xử lý của CPU. Trong trường hợp này có thể giải quyết bằng bốn cách sau:

- Cách 1: trước khi gửi đi yêu cầu nào đó, stub của máy trạm 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 xảy ra lỗi, máy trạm sẽ lấy lại bản ghi đó và và việc thực hiện 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 trạm được phục hồi thì tham số đó được tăng thêm một đơn vị. Lúc này 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 sẽ kiểm tra xem mình có đang thực hiện một tính toán từ xa nào hay không. Nếu có, máy đó sẽ cố xác định xem máy trạm 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 trạm sẽ đợi thêm một khoảng thời gian T trước khi thực hiện lại. Vấn đề đặt ra là phải lựa chọn giá trị khoảng thời gian T như thế nào cho hợp lý.

2.6.4 Truyền thông nhóm tin cậy

Sau khi phân nhóm tiến trình, một tiến trình khác muốn thực hiện gửi thông điệp tới tất cả các tiến trình trong nhóm đó. Truyền thông nhóm tin cậy là phải có cơ chế để đảm bảo thông điệp đến được tất cả các thành viên trong nhóm. Khi xảy ra lỗi thì che giấu bằng phương pháp đánh số tuần tự các thông điệp cần gửi. 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 bản tin nào đó thì nó sẽ gửi về một thông điệp 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.

2.6.4.1 Lược đồ truyền thông theo nhóm tin cậy cơ bản

Tin cậy là phải có cơ chế để đảm bảo thông điệp 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 ví dụ bằng cách đánh số các thông điệp cần gửi. Các thông điệp được lưu tại vùng đệm của bên gửi và vẫn lưu ở đó cho đến khi nhận được thông điệp xác nhận báo về từ bên nhận. 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 nào đó mà nó không nhận được thông điệp xác nhận.

Để tăng hiệu quả công việc khi làm việc với một số lượng lớn các tiến trình thì đưa ra mô hình truyền thông nhóm tin cậy mở rộng. Với mô hình này sẽ không gửi trả về thông điệp xác nhận thành công mà chỉ gửi trả về cho tiến trình nhận thông điệp NACK thông báo khi có lỗi truyền. Việc này được thực hiện bằng giao thức SRM (Scalable Reliable Multicasting).

Để có thể thực hiện truyền thông nhóm tin cậy cho một số lượng lớn các tiến trình thì thực hiện tổ chức các nhóm theo cấu trúc dạng cây. Cấu trúc của cây, gốc là nhóm

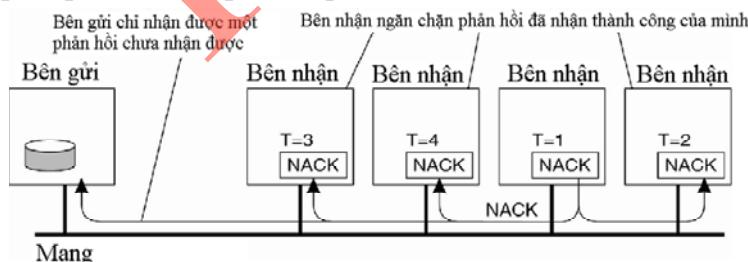
chứa tiến trình gửi và các nút là các nhóm có chứa tiến trình nhận. Việc chia thành các nhóm nhỏ hơn cho phép sử dụng các kịch bản truyền thông nhóm tin cậy cho từng nhóm. Trong mỗi nhóm nhỏ sẽ đề cử một tiến trình làm điều phối. Tiến trình điều phối có khả năng điều khiển việc truyền lại khi nhận được thông báo truyền lỗi. Tiến trình điều phối của mỗi nhóm sẽ có bộ đệm riêng. Nếu tiến trình điều phối của mỗi nhóm không nhận được thông điệp thì nó sẽ gửi yêu cầu truyền lại tới coordinator của nút cha nó.

Trong qui định của truyền số liệu tin cậy sử dụng thông điệp xác nhận thì khi coordinator nhận thành công một thông điệp m nó sẽ gửi thông điệp xác nhận tới tiến trình điều phối của nút cha nó. Nếu tiến trình điều phối của một nhóm nhận được thông điệp xác nhận thành công việc chuyển thông điệp m của tất cả các tiến trình trong nhóm gửi về thì nó sẽ xóa thông điệp m khỏi bộ đệm của nó. 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.

Khi một tiến trình muốn gửi thông điệp cho một tập các tiến trình khác theo kiểu multicast, nó sẽ không gửi thông điệp tới tất cả các tiến trình của nhóm chứa các tiến trình nhận 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 kỳ tiến trình nào nếu một tiến trình trong nhóm bị lỗi nghiêm trọng.

2.6.4.2 Truyền tin nhóm tin cậy trong các hệ thống lớn

Truyền tin nhóm tin cậy không hỗ trợ 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.

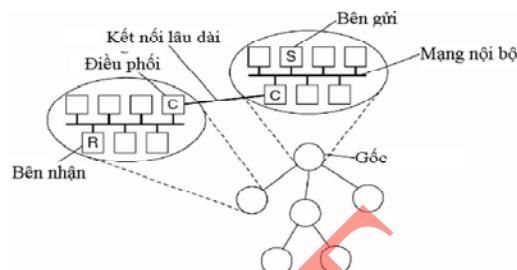


Hình 2.49 Phản hồi không phân cấp

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 hổn nứa 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 2.50 Điều khiển phản hồi phân cấp

Thay vì truyền tin 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 bầu chọn 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 nhóm 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.

2.6.5 Cam kết phân tán

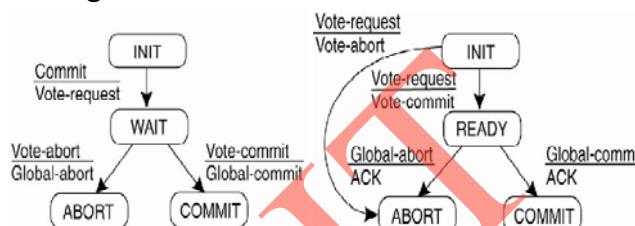
Một yêu cầu quan trọng của giao tác phân tán là phải đảm bảo tính nguyên tử, nghĩa là ảnh hưởng của giao tác phải được thực thi thành công trên tất cả các thành viên trong nhóm hoặc không có một thành viên nào được thực thi, đặc tính này có thể đạt được bằng cách cài đặt giao thức khẳng định nguyên tử (ACP – Atomic Commit Protocol). Giao thức khẳng định một pha giảm độ phức tạp của cả thông điệp lẫn nhật ký dựa trên giả thiết đúng trong các xử lý của các giao tác. Trong khi một số giả thiết đáng tin cậy thì một số giả thiết khác chỉ được sử dụng hạn chế trong một số ứng dụng. Giao thức khẳng định một pha chứa bầu chọn đồng ý không tường minh và nhật ký điều phối coi mỗi hoạt động của giao tác đều được xác nhận trong mỗi thành phần tham gia. Thao tác xác nhận trong các giao thức này không chỉ có ý nghĩa 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 người ta đưa mô hình mới đó là cam kết hai pha và cam kết ba pha.

2.6.5.1 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 và không có lỗi xảy ra. Cam kết 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 cầu 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 bầu cử 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. Tuy nhiên, 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.

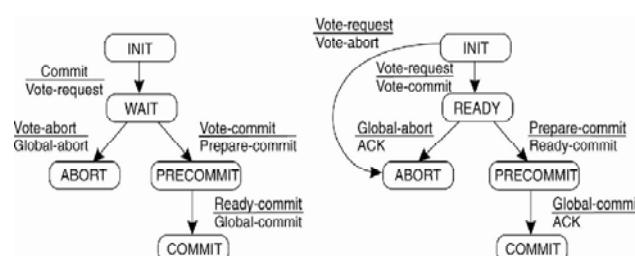


Hình 2.51 Cam kết hai pha

Các thành viên sau khi đã gửi thông báo chấp nhận cam kết sẽ đợi phản hồi từ tiến trình điều phối. Nếu nó nhận về thông báo GLOBAL_COMMIT thì giao dịch sẽ được chấp thuận, còn nếu nhận được GLOBAL_ABORT thì giao dịch 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 số 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.

2.6.5.2 Cam kết ba pha

Nhược điểm chính 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 một thông điệp nào đó được gửi đến.



Hình 2.52 Cam kết ba pha

Nếu tiến trình điều phối bị lỗi thì hoạt động của cả hệ thống sẽ bị ảnh hưởng. Để khắc phục nhược điểm của cam kết hai pha trong trường hợp tiến trình điều phối bị lỗi, người ta đưa ra mô hình cam kết ba pha. Các trạng thái trong cam kết ba pha khá giống cam kết hai pha nhưng thêm một trạng thái PRECOMMIT.

2.6.6 Phục hồi

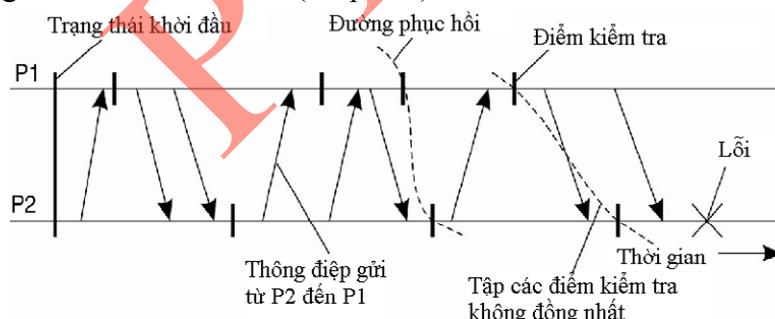
Lỗi có thể xảy ra ở bất kỳ thời điểm nào, một hệ thống đáng tin cậy phải không những có khả năng phát hiện lỗi mà còn phải biết phục hồi sau khi gặp lỗi, nói cách khác phục hồi là các biện pháp đưa hệ thống từ trạng thái bị lỗi trở về trạng thái bình thường.

2.6.6.1 Giới thiệu

Có hai cách tiếp cận cho phục hồi lỗi: phục hồi lùi (back forward) và phục hồi tiến (forward recovery). Tư tưởng của phương pháp phục hồi lùi là đưa trạng thái của hệ thống ở thời điểm hiện tại về trạng thái tại thời điểm trước khi lỗi xảy ra. Để làm được điều đó cần phải liên tục ghi lại trạng thái của hệ thống, thời điểm hệ thống thực hiện sao lưu dữ liệu và trạng thái gọi là điểm kiểm tra (checkpoint). Khi phát hiện có lỗi xảy ra, giải thuật lùi sẽ lấy lại trạng thái và dữ liệu của hệ thống tại thời điểm gần nhất trước khi xảy ra lỗi. Giải pháp phục hồi tiến không đưa hệ thống trở lại trạng thái trước khi lỗi như phục hồi lùi mà chuyển hệ thống sang trạng thái mới để có thể hoạt động bình thường. Khó khăn lớn nhất của phương pháp này là phải dự đoán chính xác khi nào lỗi có thể xảy ra. Phương pháp phục hồi tiến chỉ áp dụng cho hệ thống có đặc tính riêng biệt, đặc thù nào đó.

2.6.6.2 Điểm kiểm tra

Phục hồi lùi đòi hỏi hệ thống phải thường xuyên ghi lại trạng thái của nó vào bộ nhớ vĩnh cửu (thường là ổ đĩa), đặc biệt hệ thống phân tán có khả năng chịu lỗi phải ghi được trạng thái toàn cục của nó (snapshot).



Hình 2.53 Phục hồi lùi sử dụng trạng thái toàn cục

Các tiến trình trong hệ thống phân tán thường trao đổi thông điệp cho nhau, mỗi sự kiện gửi hoặc nhận thông điệp sẽ được ghi lại trong bộ nhớ vĩnh cửu. Nếu có lỗi xảy ra, hệ thống sẽ dò tìm trạng thái nhất quán toàn cục gần nhất để phục vụ cho công tác phục hồi.

2.7 Bảo mật

Bảo mật trong hệ thống phân tán có thể chia làm hai nhóm chính: Bảo mật trong quá trình trao đổi thông tin và bảo vệ việc lưu giữ thông tin tại các kho dữ liệu.

2.7.1 Khái niệm chung

Bảo mật là một trong những yêu cầu cấp thiết trong hệ thống thông tin, nhằm tránh hoặc hạn chế các thiệt hại do các cuộc tấn công, bảo đảm tính bí mật, toàn vẹn, khả năng sẵn sàng cao cho hệ thống.

- Tính bí mật: Bảo vệ dữ liệu của hệ thống thông tin, chỉ những người dùng hợp pháp mới được phép truy cập thông tin và thông tin đó được trao đổi theo các cơ chế đảm bảo an toàn, tránh truy cập của người dùng bất hợp pháp.
- Tính toàn vẹn: Đảm bảo dữ liệu không bị sửa đổi hoặc phá hủy bởi người dùng không hợp pháp.
- Tính sẵn sàng: Đảm bảo khả năng hoạt động liên tục của hệ thống, đáp ứng nhu cầu sử dụng tài nguyên của người dùng. Điều này cũng liên quan tới vấn đề dự phòng và khắc phục khi xảy ra sự cố.

Các điểm yếu của hệ thống phân tán bao gồm:

Điểm yếu về kỹ thuật truyền tin trên mạng và điểm yếu về chính sách bảo mật thông tin.

Điểm yếu về kỹ thuật truyền tin trên mạng nằm trong đặc tính mỏ của các giao thức truyền tin như: HTTP, FTP, ICMP, SMNP, SMTP.

Điểm yếu về chính sách bảo mật hệ thống thông tin bao gồm:

Không xây dựng chính sách khắc phục khi tấn công xảy ra.

Không có qui trình giám sát và kiểm tra các hoạt động trong hệ thống thông tin.

Hiểm họa đối với thông tin bao gồm những vấn đề liên quan tới điều kiện vật lý và các cuộc tấn công vào hệ thống. Hiểm họa vật lý có thể kể tới vấn đề nguồn điện, môi trường... Các cuộc tấn công từ bên ngoài có thể xuất hiện ở các dạng sau: Tấn công thăm dò, truy nhập trái phép, tấn công từ chối dịch vụ và tấn công bằng phần mềm chứa mã độc như: Virus, Trojan Horses, worms...

2.7.1.1 Tấn công thăm dò

Kẻ tấn công sử dụng để khám phá ra hệ thống thông tin, bao gồm hình trạng mạng, hệ điều hành, các dịch vụ sử dụng và các điểm yếu có thể lợi dụng để tấn công nhằm mục đích cuối cùng là lấy cắp dữ liệu, bắt đầu cho một cuộc tấn công nguy hiểm hoặc tấn công từ chối dịch vụ. Tấn công thăm dò thường được thực hiện theo các bước sau:

Bước 1: Ping đến địa chỉ đích xem địa chỉ IP đó có hoạt động không.

Bước 2: Quét cổng để xác định cổng hay dịch vụ nào đang hoạt động trên địa chỉ IP đó.

Bước 3: Truy vấn đến các cổng, từ đó có thể xác định kiểu và phiên bản của ứng dụng, hệ điều hành.

Bước 4: Từ các bước trên suy ra các điểm yếu có thể lợi dụng để tấn công.

2.7.1.2 Truy nhập trái phép

Để truy nhập được vào hệ thống thông tin phải có tài khoản đang nhập, việc lộ bí mật tài khoản dưới bất cứ hình thức nào cũng có thể gây nên tổn hại về mất mát thông tin. Đối tượng truy nhập trái phép trước hết phải đánh cắp thông tin tài khoản, công việc này có thể được thực hiện bằng phương pháp sau:

Thăm dò mật khẩu: nhằm mục đích lấy cắp thông tin đăng nhập của người dùng bao gồm tên đăng nhập và mật khẩu. Có thể sử dụng phương pháp dự đoán mật khẩu theo ký tự từ điển, đoán mật khẩu

Đóng vai trung gian: Sử dụng các công cụ và phần mềm bắt gói tin, phân tích sửa đổi và là người trung gian chuyển dữ liệu cho máy trạm và máy chủ.

Sử dụng tính năng tin cậy có sẵn trong một số hệ thống truyền tin: Một số thiết bị và máy tính có cài đặt sẵn tính năng này nhằm nâng cao hiệu suất hoạt động.

2.7.1.3 Tấn công từ chối dịch vụ

Kẻ tấn công vô hiệu hóa hoặc phá hỏng hệ thống thông tin, làm chậm hệ thống và các dịch vụ ứng dụng với mục đích từ chối yêu cầu sử dụng dịch vụ, tài nguyên hệ thống của người dùng hợp pháp.

2.7.1.4 Phần mềm độc hại

Phần mềm độc hại (Worms, Virus, Trojan Horses) là các chương trình chứa mã độc được chèn vào máy tính để phá hủy hệ thống, làm hỏng hệ thống, tự nhân bản và phát tán, từ chối dịch vụ hoặc truy cập tới mạng, hệ thống và các dịch vụ ứng dụng. Virus là phần mềm chứa mã độc được gắn vào một chương trình khác để thực thi các hoạt động phá hoại trên máy tính. Virus có thể làm nguy hại đến dữ liệu, phá hủy hệ điều hành và phát tán đến hệ thống khác. Viruses xâm nhập vào máy tính theo một trong 3 cách sau: đĩa mềm hoặc CD-ROM, USB, thông qua e-mail.... Các dấu hiệu khi hệ thống bị nhiễm virus:

Các chương trình khởi động chậm hơn. Điều này bởi vì virus đang phát tán tới các files khác trên hệ thống hoặc đang chiếm tài nguyên.

Xuất hiện các file lạ trên ổ cứng hoặc mất file. Một số viruses xóa các files quan trọng trong hệ thống.

Kích thước chương trình thay đổi so với phiên bản cài đặt ban đầu. Điều này xảy ra vì virus gắn chính nó vào chương trình.

Trình duyệt, chương trình xử lý văn bản, Screens, menus và cách ứng dụng khác bị thay đổi.

Hệ thống shutdown hoặc khởi động một cách thất thường.

Mất quyền truy cập đến ổ cứng hoặc các tài nguyên hệ thống khác. Virus có thể thay đổi các thiết lập trên thiết bị làm cho nó không hoạt động chính xác.

Hệ thống không khởi động hoặc đưa ra các thông báo lỗi trong suốt quá trình khởi động.

Trojan horse không tự nhân bản. Khi được thả vào máy tính, chúng thường tạo ra một "cửa sau". Hacker có thể thông qua "cửa sau" xâm nhập và nắm quyền kiểm soát máy tính.

Worm là một dạng mã độc có khả năng tự nhân bản, lây lan và không cần vật chủ, chúng có kích thước lớn hơn virus và thường nấp dưới dạng một file đính kèm e-mail. Worms không phải là viruses nhưng hay được sử dụng như một công cụ để chứa và phát tán virus tới một hệ thống đích. Khi máy tính bị nhiễm Worms nó sẽ nhanh chóng chiếm bộ nhớ và tự nhân bản trong RAM. Worms có thể sử dụng TCP/IP, e-mail, Internet services, và các dịch vụ khác để phát tán đến hệ thống đích.

2.7.2 Các kênh bảo mật

Giải pháp bảo mật trong hệ thống phân tán cần phải thực hiện bằng cả biện pháp bảo vệ mạng thông tin cũng như nơi cất giữ thông tin. Các giải pháp bảo mật mạng bao gồm:

- Xây dựng bức tường lửa
- Xây dựng hệ thống mạng riêng ảo
- Xây dựng hệ thống phát hiện và ngăn chặn đột nhập
- Phương pháp kiểm soát xác thực truy nhập

2.7.2.1 Xây dựng bức tường lửa

Tường lửa được phân ra thành hai dạng: Tường lửa bảo vệ toàn bộ mạng và tường lửa bảo vệ riêng từng máy tính. Bức tường lửa có thể được xây dựng dựa trên cấu trúc các gói dữ liệu theo mô hình phân lớp mạng, kết quả của chính sách bảo mật bằng phương pháp bức tường lửa phải được thể hiện bằng danh sách kiểm soát truy nhập.

2.7.2.2 Xây dựng 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 các kênh thuê 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 về tốc độ, không vượt quá tốc độ của đường truyền Internet.

2.7.2.3 Hệ thống phát hiện và ngăn chặn đột nhập

Phát hiện đột nhập là khả năng phát hiện, nhận dạng mã độc tấn công vào mạng hoặc một máy tính nào đó. Hệ thống phát hiện đột nhập có khả năng phát hiện các kiểu tấn công thăm dò, tấn công truy nhập, tấn công từ chối dịch vụ (DDoS), worms, virus.. và có thể được cấu hình để gửi cảnh báo khi phát hiện những kiểu lưu lượng không rõ ràng.

Ngăn chặn đột nhập là khả năng phát hiện và ngăn chặn việc thực thi các mã độc tấn công xâm nhập vào hệ thống. Khi phát hiện có xâm nhập trái phép, một thiết bị cảm ứng sẽ thực hiện các bước sau:

- Gửi cảnh báo đến hệ thống quản lý giám sát
- Hủy bỏ phiên kết nối
- Ngay lập tức vứt bỏ gói tin
- Từ chối các gói tin đến từ địa chỉ đã tấn công

2.7.2.4 Xác thực truy nhập

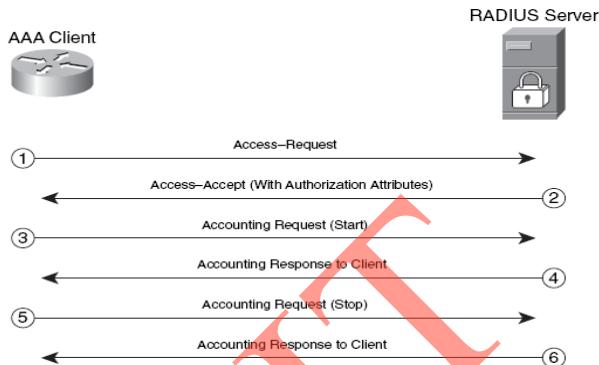
Đây là quá trình xác nhận tính đúng đắn của người dùng dựa vào các thông tin mà họ có, như mật khẩu hay chứng chỉ số khi họ đăng nhập vào hệ thống. Một số phương pháp xác thực được sử dụng bao gồm:

- Kerberos
- PAP
- CHAP
- Username/password
- Certificates
- Tokens
- Multi-Factor

Biometrics
Smart card
Radius

RADIUS là giao thức chuẩn được sử dụng phổ biến, thường hoạt động trong môi trường máy trạm/máy chủ. Các thiết bị mạng như Router, Firewall, Switches là các Máy chủ truy cập mạng (NAS), đồng thời cũng là RADIUS máy trạm có nhiệm vụ xác thực người dùng dựa vào thông tin đáp trả mà nó nhận được từ RADIUS máy chủ. Hiện nay, RADIUS sử dụng cổng UDP 1812 cho xác thực (authentication) và cổng UDP 1813 cho thanh toán (accounting). Vì RADIUS là một giao thức chuẩn chung nên được triển khai rộng rãi hơn TACACS+ (giao thức được phát triển bởi Cisco).

Quá trình xác thực sử dụng RADIUS diễn ra như sau:



Hình 2.54 Qui trình xác thực Radius

Bước 1: Khi nhận được yêu cầu xác thực từ người dùng (user), AAA máy trạm (Router) sẽ gửi yêu cầu truy nhập (access-request) tới RADIUS máy chủ.

Bước 2: Nếu user được xác nhận là đúng, RADIUS máy chủ sẽ gửi trả lại một thông báo chấp nhận là access-accept. Thông báo này có thể chứa thông tin về các thuộc tính phân quyền (authorization) nếu nó được cấu hình. Nếu user không được xác nhận là đúng RADIUS máy chủ sẽ gửi lại cho AAA máy trạm thông điệp từ chối truy nhập (access-reject).

Bước 3: Nếu accounting được cấu hình, AAA máy trạm sẽ gửi thông báo accounting request (start) tới RADIUS máy chủ.

Bước 4: Nếu RADIUS máy chủ đã cấu hình accounting, nó sẽ trả lời với thông báo chấp nhận (Acknowledgement).

Bước 5: Khi người dùng kết thúc phiên sử dụng, AAA máy trạm sẽ gửi thông báo accounting request (stop) đến RADIUS máy chủ.

Bước 6: RADIUS máy chủ sẽ trả lời với thông báo chấp nhận (Acknowledgement). Kết thúc phiên truy nhập.

Để đảm bảo an toàn cho thông tin trên hệ thống phân tán cần phải xây dựng các chính sách bảo mật. Chính sách bảo mật cần phải được cụ thể hóa đến từng đối tượng và các thành phần trong hệ thống. Chính sách bảo mật máy chủ nhằm mục đích ngăn chặn truy nhập trái phép đến các máy chủ trong hệ thống. Nội dung cơ bản của chính sách bảo mật máy chủ bao gồm chính sách bảo vệ vật lý và bảo vệ logic. Chính sách bảo vệ vật lý bao gồm:

Tránh nguy cơ ảnh hưởng của môi trường: Để tránh ảnh hưởng của môi trường đến thiết bị, Data center cần có điều hòa nhiệt độ, điều hòa độ ẩm. Các tham số về môi trường trong trung tâm dữ liệu cần được giám sát và cảnh báo từ xa. Ngoài ra, phòng đặt trung tâm dữ liệu phải không bị ảnh hưởng của nhiễu từ trường.

Đối với nguồn cung cấp điện: Để tránh những nguy cơ về điện làm ảnh hưởng đến hoạt động của hệ thống mạng, phòng trung tâm dữ liệu cần được cung cấp một nguồn điện ưu tiên đặc biệt hơn các phòng khác. Cần xây dựng hệ thống điện của công ty đủ và liên tục để cung cấp nguồn điện cho các thiết bị. Đối với nguồn điện cần sử dụng bộ lưu điện (UPS), máy phát điện, nguồn cung cấp dự phòng và nên chú ý vấn đề cách điện nhằm ngăn chặn cháy nổ do điện gây ra. Ngoài ra, cần sử dụng hệ thống cảnh báo, giám sát theo dõi các tham số nguồn điện từ xa.

Đối với việc bảo trì: Cáp sử dụng phải được đánh nhãn rõ ràng, cần bảo vệ tủ rack để tránh việc đứt kết nối, lỏng đầu cắm cáp hay cắm sai cổng, việc đi cáp các dây phải được bó gọn. Nên có một kho chứa các thiết bị dự phòng khi cần, log off giao diện quản trị khi thực hiện xong công việc, bảo vệ truy cập đến cổng console, khi thay thế hay thao tác trực tiếp với các thiết bị cần chú ý tránh để điện làm hỏng thiết bị.

Chính sách bảo vệ logic bao gồm:

Ghi chép thông tin cơ bản về máy chủ: Địa điểm lắp đặt, người chịu trách nhiệm quản lý trực tiếp. Các thông tin về cấu hình và dịch vụ

Việc thay đổi cấu hình phải tuân thủ các qui định đã được phê chuẩn.

Loại bỏ hoặc tạm ngừng các dịch vụ và ứng dụng không cần thiết.

Phải có qui trình theo dõi hoạt động của máy chủ.

Thường xuyên cập nhật và cài đặt các bản vá lỗi cho hệ điều hành và các ứng dụng trên máy chủ.

Hạn chế tối đa việc cấu hình tự động truy nhập hoặc cho phép truy nhập từ hệ thống khác.

Xây dựng qui trình lưu dự phòng dữ liệu.

Xây dựng chính sách truy nhập từ xa đến máy chủ.

Xây dựng qui định về bảo mật tài khoản truy nhập (qui định cách đặt tên, mật khẩu, thời gian quá hạn cần phải thay đổi...)

Ngoài các chính sách trên, cần phải xây dựng chính sách bảo mật các máy trạm nhằm đảm bảo an toàn cho thông tin được lưu trữ trên các máy và thông tin mà các máy trạm phải truy cập tới.

2.7.2.5 Giới thiệu một số phương pháp mã hóa

Mã hóa thông tin là các giải pháp khác nhằm bảo vệ thông tin khi lưu chuyển trên mạng cũng như truy nhập vào kho dữ liệu. Có thể sử dụng các thuật toán mã hóa thông tin như DES, RSA, MD5... Tuy nhiên việc mã hóa thông tin sẽ làm tăng công việc xử lý, điều đó làm ảnh hưởng đến hiệu suất hoạt động của hệ thống. Do đó việc mã hóa thông tin cần phải thực hiện cho từng loại thông tin nhằm đảm bảo tính sẵn sàng phục vụ của hệ thống phân tán, nói cách khác, cần phải biết cân đối hài hòa giữa mục tiêu bảo mật và khả năng đáp ứng yêu cầu dịch vụ của hệ thống.

Tư tưởng cơ bản trong việc mã hóa thông tin như sau: 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. Gọi bản tin ban đầu là P, khóa mã hóa là E_k, bản tin được mã hóa theo khóa E_k là C: C=E_k (P), khóa giải mã là D_k, bản tin được giải mã theo khóa giải mã:

$P=D_k(C)$. Có hai loại hệ thống mật mã: mật mã đối xứng (symmetric cryptosystem) và mật mã bất đối xứng (asymmetric cryptosystem).

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: $P=D_k(E_k(P))$. 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): Thực hiện trên các khối dữ liệu 64 bit. Mỗi khối 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: L_i -1 và R_i -1, cùng bằng 32 bit. Phần dữ liệu bên phải R_i -1 được lấy làm phần bên trái của dữ liệu cho vòng sau: $R_i = L_{i-1}$. Hàm f với đầu vào là R_i -1 và khóa K_i sinh ra khối 32 bit được XOR với L_i -1 để sinh ra R_i . 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 vòng, 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.

Mật mã bất đối xứng: dùng khóa công khai. Khóa mã hóa và khóa giải mã là khác nhau. Ta có: $P=D_kD(E_kD(P))$. Trong hệ thống 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

Tính $n = p \cdot q$ và $\phi = (p-1)(q-1)$

Chọn một số d liên quan đến ϕ

Tính toán e sao cho $e \cdot d = 1 \pmod{\phi}$.

Như vậy d có thể dùng để giải mã còn e dùng để mã hóa. Ta có thể công khai một trong hai số này, tùy thuộc 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 $0 \leq m_i \leq n$. Ở bên gửi, với mỗi khối mi sẽ tính một giá trị $c_i = me_i \pmod{n}$ rồi gửi đi. Bên nhận sẽ giải mã bằng cách tính: $m_i = c_i d \pmod{n}$. Như vậy, để mã hóa cần biết e và n còn để giải mã thì cần biết d và n .

2.7.3 Kiểm soát truy cập

Trong mô hình khách/chủ, máy khách sử dụng kênh bảo mật để kết nối đến máy chủ và sử dụng tài nguyên hệ thống. Để bảo đảm vấn đề an ninh, cần thiết phải xây dựng các chính sách kiểm soát các thao tác của người dùng trên máy chủ. Cách đơn giản nhất thường được áp dụng là xây dựng ma trận kiểm soát truy cập, đó là một ma trận gồm hàng biểu diễn cho một chủ thể truy nhập (người dùng) và cột biểu diễn cho tài nguyên. Giá trị chứa trong mỗi phần tử của ma trận thể hiện quyền người dùng được thao tác trên tài nguyên đó.

Thông thường hệ thống sẽ có rất nhiều người sử dụng, do đó xây dựng một ma trận thực như trên là không hợp lý, giải pháp khác là sử dụng danh sách kiểm soát truy cập ACL (Access Control List). Mỗi đối tượng sẽ duy trì một danh sách các truy nhập hợp lệ của các chủ thể muốn truy cập nó gọi là ACL nhờ đó tránh được sự tồn tại của các mục rỗng như trong ma trận kiểm soát truy nhập.

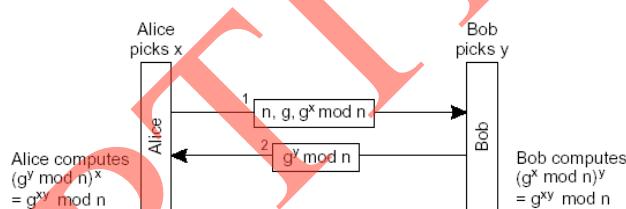
Danh sách kiểm soát truy nhập tuy đã khắc phục được nhược điểm của ma trận kiểm soát truy nhập nhưng vẫn có kích thước lớn nên đã đưa ra cách sử dụng miền bảo vệ. Miền bảo vệ là một tập các cặp (đối tượng, truy cập hợp lệ), mỗi cặp này sẽ cho ta một đối tượng và các thao tác hợp lệ trên nó. Mỗi một yêu 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.

Tường lửa (Firewall) dùng để ngăn chặn các luồng không được phép, chúng có thể lọc thông tin từ mức mạng đến lớp ứng dụng. Tường lửa lọc gói tin (Packet filtering gateway) hoạt động như một thiết bị định tuyến cho phép hoặc không cho phép gói tin chuyển qua mạng dựa trên địa chỉ nguồn và địa chỉ đích của gói tin, thường dùng để ngăn chặn các gói tin từ ngoài đi vào trong mạng. Tường lửa mức ứng dụng (Application level gateway) không những kiểm tra thông tin điều khiển của gói tin mà còn kiểm tra nội dung của gói tin đó.

2.7.4 Quản lý bảo mật

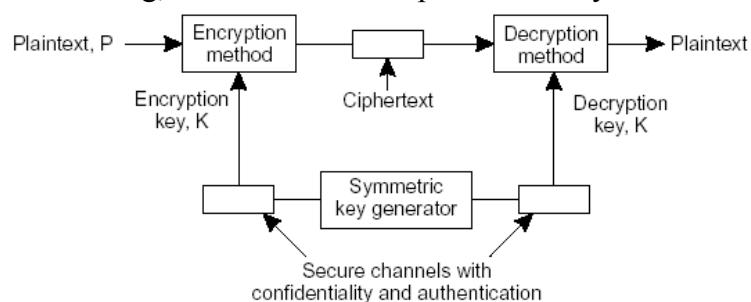
2.7.4.1 Quản lý khóa

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 $(g^x \bmod n)$.



Hình 2.55 Nguyên lý trao đổi khóa Diffie – Hellman

Bên B sẽ thực hiện tính $(g^x \bmod n)^y = g^{xy} \bmod n$, do đó sẽ xác định được khóa bí mật x của bên A. Đồng thời, bên B cũng gửi cho bên A $(g^y \bmod n)$. Bên A thực hiện tính toán $(g^y \bmod n)^x = g^{xy} \bmod n$ nhờ đó cũng xác định được khóa bí mật y của bên B. Trong hệ mã mật đối xứng, khóa bí mật tạo ra phải được truyền đi trên kênh mật riêng.



Hình 2.56 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 xác định danh để xác định được khóa mật liên kết với nó. Khi cần hủy bỏ một chứng chỉ ta có thể thực hiện theo nhiều cách:

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.

2.7.4.2 Quản trị nhóm an toàn

Xét nhóm G, khóa mật CK_G 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 (K_G^+ , K_G^-) 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 $K_{P,G}$ được mã hóa sử dụng khóa công khai K_G^+ của nhóm. JR được gán bởi P và nó được gửi đi cùng với chứng chỉ chứa 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 chia sẻ N. RP được sử dụng để mã hóa khóa giao tiếp của nhóm CK_G . P sử dụng khóa K_G^- để mã hóa cùng với CK_G . Sau đó thông điệp GA được gán cho Q sử dụng khóa $K_{P,G}$.

2.7.4.3 Quản lý ủy quyền

Quản lý bảo mật trong hệ thống phân tán liên quan mật thiết với việc quản lý quyền truy nhập. Để xác định quyền truy nhập tài nguyên người ta định nghĩa một định danh 128 bit với cấu trúc bên trong gồm bốn trường thông tin: 48 bit đầu tiên gọi là cổng máy chủ (server port), 24 bit tiếp theo được sử dụng để xác định đối tượng, 8 bit tiếp theo xác định quyền truy nhập và 48 bit cuối được dùng để tạo ra một chứng chỉ thật. Ủy quyền truy nhập là một kỹ thuật quan trọng để bảo vệ máy tính trong hệ thống phân tán, nó được thực hiện bằng cách chuyển quyền truy nhập từ tiến trình này sang tiến trình khác nhằm mục đích phân tán công việc giữa các tiến trình mà không làm ảnh hưởng tới việc bảo vệ tài nguyên.

2.8 Tính nhất quán và vấn đề nhân bản

Để đảm bảo an toàn thông tin cho hệ thống phân tán, người ta thường tổ chức lưu trữ dữ liệu tại nhiều vị trí vật lý khác nhau và đồng thời tạo ra các bản sao cho các dữ liệu đó. Trong trường hợp này, một vấn đề nảy sinh đó là tính toàn vẹn của các dữ liệu trên toàn bộ hệ thống, nói cách khác cần phải đảm bảo tính đồng nhất dữ liệu.

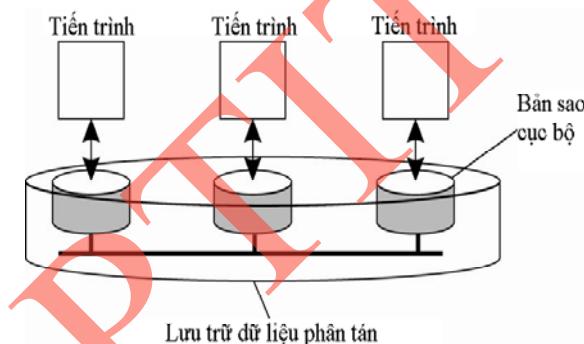
2.8.1 Khái niệm chung

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 chấp nhận sự suy giảm của tính toàn vẹn dữ liệu. 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 toàn vẹn của dữ liệu.

2.8.2 Các mô hình nhất quán lấy dữ liệu làm trung tâm

Khái niệm nhất quán ở đây 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 kho dữ liệu.



Hình 2.57 Tô chúc kho dữ liệu trong hệ thống phân tán

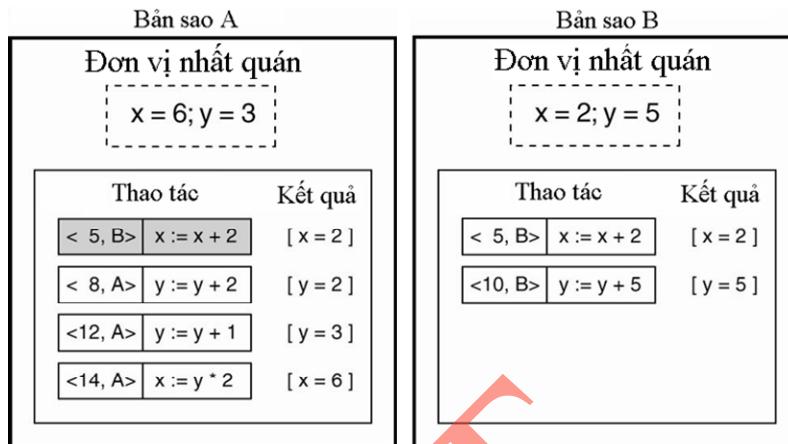
Mô hình nhất quán, về cơ bản là thỏa thuận giữa tiến trình và kho dữ liệu, nếu tiến trình tuân thủ một số qui tắc thì kho dữ liệu hứa hẹn sẽ làm việc chính xác. Thông thường, tiến trình thực hiện thao tác đọc mục dữ liệu nào đó và hy vọng sẽ nhận được giá trị phản ánh kết quả ghi cuối cùng đã thực hiện trên mục dữ liệu đó. Nếu thiếu đồng hồ toàn cục thì khó có thể biết được thao tác ghi nào là thao tác được thực hiện cuối cùng. Vì vậy chúng ta phải định nghĩa một số mô hình nhất quán, mỗi mô hình sẽ hạn chế khả năng đọc giá trị của các khoản mục dữ liệu, nói cách khác sẽ không có mô hình nhất quán toàn diện.

2.8.2.1 Nhất quán liên tục

Hiện nay chưa có một giải pháp nhân bản nào được coi là hoàn hảo tuyệt đối, nhân bản dữ liệu đặt ra những vấn đề liên quan tới tính nhất quán và do đó không có một phương pháp tổng quát nào giải quyết được vấn đề này. Chỉ khi chúng ta nói lỏng tính nhất quán thì mới hy vọng đạt được giải pháp hiệu quả, đáng tiếc lại không có một qui tắc chung nào để nói lỏng tính nhất quán, điều này hoàn toàn dựa vào yêu cầu khi

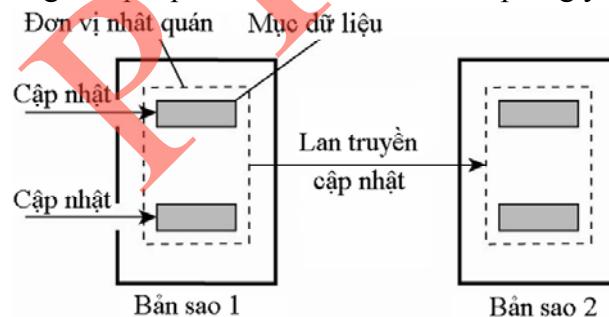
xây dựng phần mềm. Ví dụ, khi xây dựng phần mềm có thể đưa ra các tiêu chí nhất quán sau:

- 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



Hình 2.58 Tính nhất quán với điều kiện nói lỏng số lượng thao tác

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, như vậy nếu độ lệch số lượng thao tác thực hiện trên các bản sao không vượt quá ngưỡng cho phép thì có thể coi như đã đáp ứng yêu cầu nhất quán.



Hình 2.59 Lựa chọn số lượng thao tác trong thủ tục lan truyền cập nhật

Lý tưởng, một thao tác được thực hiện trên một bản sao phải được thực hiện ngay lập tức trên các bản sao khác, hoặc ít nhất sau khi thao tác cập nhật đã hoàn thành trên bản sao thì phải được chuyển sang các bản sao khác. Tuy nhiên thực tế sẽ gặp nhiều trở ngại về mạng, nhất là những hệ thống thường xuyên cập nhật, vì vậy quyết định mỗi lần lan truyền cập nhật gồm bao nhiêu thao tác sẽ do người thiết kế hệ thống phần mềm quyết định.

2.8.2.2 *Nhất quán thứ tự các thao tác*

Bên cạnh tính nhất quán liên tục, mô hình nhất quán lấy dữ liệu làm trung tâm còn phải giải quyết vô số các nhiệm vụ khác, các tiến trình tính toán phân tán và song

song còn phải đương đầu với vấn đề chia sẻ tài nguyên và đảm bảo tính tương tranh. 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:

Giả sử x có giá trị ban đầu là null, ký hiệu $W(x)a$ là thao tác ghi được thực hiện bởi tiến trình $P(i)$ lên mục dữ liệu x với giá trị a. $R(x)b$ là thao tác đọc được thực hiện bởi tiến trình $P(j)$ lên mục dữ liệu x cho kết quả b.

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

Hình 2.60 Mô hình nhất quán chặt

Mô hình nhất quán chặt thỏa mãn 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.
- Sử dụng khái niệm thời gian tuyệt đối. Thời gian tuyệt đối này là tổng thể 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

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

Nhất quán tuần tự

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

Không thỏa mãn nhất quán tuần tự

Hình 2.61 Mô hình nhất quán tuần tự

Mô hình nhất quán tuần tự là mô hình lỏng và yếu 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.

Mô hình nhất quán tuyến tính

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 lên 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.

Mô hình nhất quán nhân quả

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

Hình 2.62 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 toàn vẹn 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.

Mô hình nhất quán FIFO

P1:	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

Hình 2.63 Mô hình nhất quán FIFO

Đây là mô hình yếu nhất vì mô hình này bỏ qua giới hạn về trật tự của bất kì 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 để 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.

Mô hình nhất quán yếu

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 toàn vẹn yếu có ba đặc tính sau:

- Việc truy cập đến một biến đồng bộ hóa được kết hợp với kho dữ liệu là một toàn vẹn 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 quán đi ra (Release)

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

Hình 2.64 Mô hình nhất quán đi ra

Mô hình nhất quán Release 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 toàn vẹn FIFO (Không yêu cầu toàn vẹn tuần tự).

Mô hình nhất quán đi vào

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

Hình 2.65 Mô hình nhất quán đi vào

Cũng giống mô hình nhất quán đi ra, mô hình nhất quán đi vào cũng sử dụng hai lệnh chiếm giữ (acquired) và giải phóng (release) khi muốn sử dụng vào vùng tối hạn. Nhưng các lệnh này thao tác trên từng mục dữ liệu của vùng dữ liệu chia sẻ. Tiến trình nào muốn sử dụng mục dữ liệu thì phải đợi cho tất cả các tiến trình khác giải phóng mục dữ liệu đó.

Để ghi lên một mục dữ liệu, máy trạm phải có được biến đồng bộ hóa của mục đó trong chế độ dành riêng. Điều đó có nghĩa là không trạm nào khác có thể sử dụng biến đó. Khi máy trạm cập nhật xong mục dữ liệu, thì nó giải phóng biến đó. Khi máy trạm 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 trạm có thể giữ một biến đồng bộ hóa ở chế độ không dành riêng. Khi thực hiện một thao tác acquire, máy trạm lấy về phiên bản mới nhất của mục dữ liệu từ tiến 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 tiến trình cho đến khi tất cả các cập nhật lên mục dữ liệu trong tiến trình đó được thực hiện.
- Trước khi một truy cập trong chế độ dành riêng của một tiến trình tới một biến đồng bộ hóa được phép thực hiện thì không tiến 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 thì bất kì sự truy cập của tiến 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.

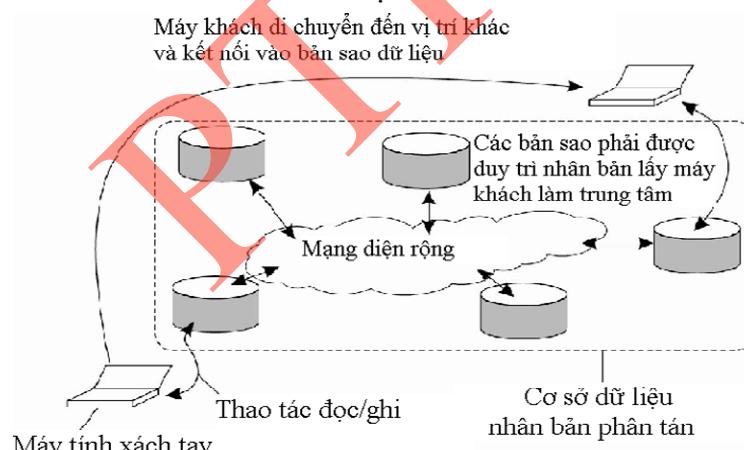
2.8.3 Các mô hình nhất quán lấy máy khách làm trung tâm

Các 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 dữ liệu. Mô hình lấy dữ liệu làm trung tâm chủ yếu giải quyết vấn đề tương tranh và tính tuần tự thực hiện các thao tác. 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

2.8.3.1 Nhất quán sau cùng

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 giống nhau. Yêu cầu này sẽ được thực hiện tốt nếu mỗi máy trạm luôn chịu khó cập nhật cho các bản sao. Nếu các máy khách là 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. Tức là các bản sao luôn luôn là toàn vẹn.

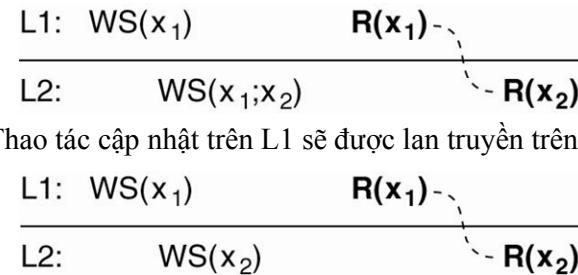


Hình 2.66 Mô hình nhất quán sau cùng

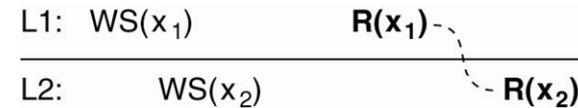
Việc cập nhật các bản sao ngay sau khi cập nhật bản chính có thể kéo dài thời gian thực hiện, do đó lập trình viên cần dự đoán thời gian thực hiện mỗi yêu cầu và lựa chọn phương án thích hợp.

2.8.3.2 Nhất quán đọc đều

Mô hình toàn vẹn đọc đều phải đảm bảo điều kiện 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 đây nhất.



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



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

Hình 2.67 Mô hình nhất quán đọc đều

Một máy trạm 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 máy trạm thực hiện một thao tác đọc trên một bản sao rồi tiếp theo đọc trên một bản sao khác thì bản sao thứ hai kia ít nhất cũng phải được ghi giống với bản sao đầu tiên.

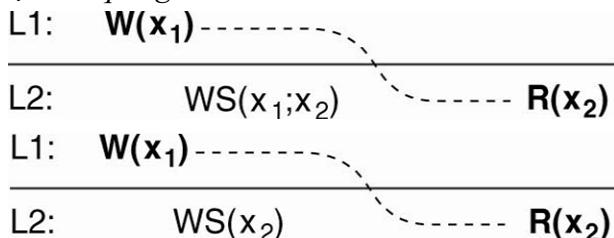
2.8.3.3 Nhất quán ghi đều



Hình 2.68 Mô hình nhất quán ghi đều

Mô hình nhất quán ghi đều phải đảm bảo điều kiện 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.

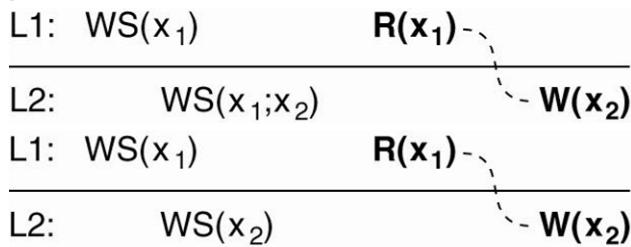
2.8.3.4 Nhất quán đọc kết quả ghi



Hình 2.69 Mô hình nhất quán đọc kết quả ghi

Trong mô hình toàn vẹn 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".

2.8.3.5 Nhát quán ghi sau khi đọc



Hình 2.70 Mô hình nhát quán ghi sau khi đọc

Mô hình nhát quán này ngược với nhát quán 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.

2.8.4 Quản lý các bản sao

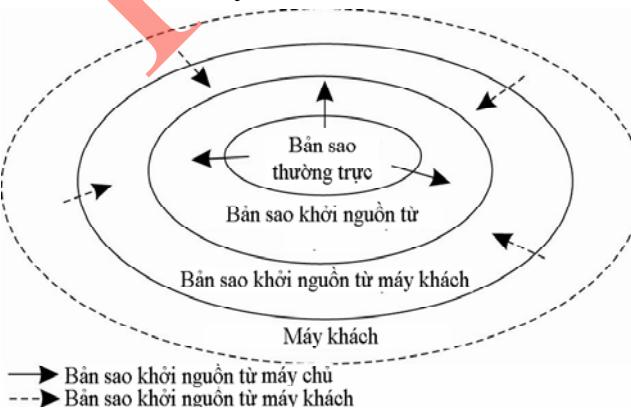
Trước khi nhân bản trong các hệ thống phân tán cần phải trả lời các câu hỏi: ở đâu, khi nào và cho ai và sau đó là các cơ chế dùng để duy trì tính nhát quán của các bản sao. Đặt bản sao ở đâu cần phải lựa chọn vị trí máy chủ và nội dung được lưu trữ trên máy chủ đó.

2.8.4.1 Đặt vị trí máy chủ nhân bản

Lựa chọn vị trí đặt máy chủ nhân bản dựa trên kết quả phân tích các thuộc tính của mạng và máy khách sao cho độ trễ truyền thông giữa máy chủ và máy khách có thể đạt giá trị thấp nhất.

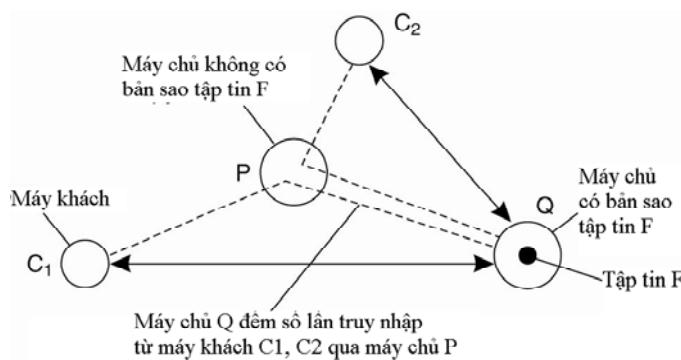
2.8.4.2 Nhân bản nội dung và vị trí lắp đặt

Đối với việc nhân bản nội dung cần phải phân biệt ba kiểu bản sao: Các bản sao thường trực, bản sao khởi đầu từ máy chủ và các bản sao khởi đầu từ máy khách.



Hình 2.71 Phân loại bản sao dữ liệu

Các bản sao thường trực là những dữ liệu thường xuyên được sử dụng trong tiến trình. Số lượng các bản sao thường xuyên này rất ít, thường được tập hợp lại thành nhóm các máy trạm hoặc trong các hệ thống phản chiếu, thường là các máy chủ web hay là các máy chủ có chứa cơ sở dữ liệu dự phòng.



Hình 2.71 Đếm số lượng yêu cầu từ máy khách

Bản sao khởi đầu từ máy chủ 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 chúng được các công ty cho thuê sử dụng để định vị vị trí địa lý của các bản sao gần nhất khi họ cần. Các bản sao khởi đầu từ máy khách được tạo ra từ yêu cầu của những ứng dụng nằm trên chính máy khách đó, chẳng hạn như việc ghi nhớ (cache) dữ liệu của một trình duyệt. Chúng được xếp đặt động tùy thuộc vào yêu cầu của người sử dụng các ứng dụng trên máy khách.

2.8.4.3 Phân bổ nội dung

Quản lý bản sao cũng phải giải quyết vấn đề cập nhật nội dung đến các máy chủ nhân bản tương ứng, có ba phương pháp cập nhật nội dung: Chỉ thông báo đã có cập nhật, truyền dữ liệu cập nhật từ bản sao này tới một bản sao khác và lan truyền các thao tác cập nhật tới các bản sao khác (nhân bản chủ động). Phương pháp chỉ thông báo đã có cập nhật thường được dùng trong việc cache dữ liệu hoặc 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 sẽ 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). Phương pháp 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.

Một vấn đề khác cần lưu ý khi thiết kế cơ chế nhân bản là lựa chọn giao thức nào, có hai loại giao thức: giao thức kéo và đẩy. Đẩy cập nhật là giao thức do máy chủ khởi tạo, trong giao thức này các cập nhật được lan truyền mỗi khi có một máy chủ khác yêu cầu. Kéo cập nhật là giao thức do máy trạm khởi tạo khi máy khách muốn được cập nhật. Đặc điểm cơ bản của các giao thức này như sau:

Vấn đề	Đẩy	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	Ngay lập tức	Thời gian quét

2.8.5 Các giao thức nhất quán

Sau khi phân tích các mô hình nhất quán, phần này sẽ giới thiệu cách cài đặt các mô hình đó trong các hệ thống phân tán.

2.8.5.1 Nhất quán liên tục

Các giao thức nhất quán liên tục dựa trên các giải pháp trong mô hình nhất quán liên tục, đó là những tiêu chí về độ chênh lệch số giữa các bản sao, chênh lệch trạng thái hay thứ tự các thao tác cập nhật. Giao thức lan truyền thường được áp dụng trong mô hình này, ý tưởng cơ bản của thuật toán lan truyền như sau:

- 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.

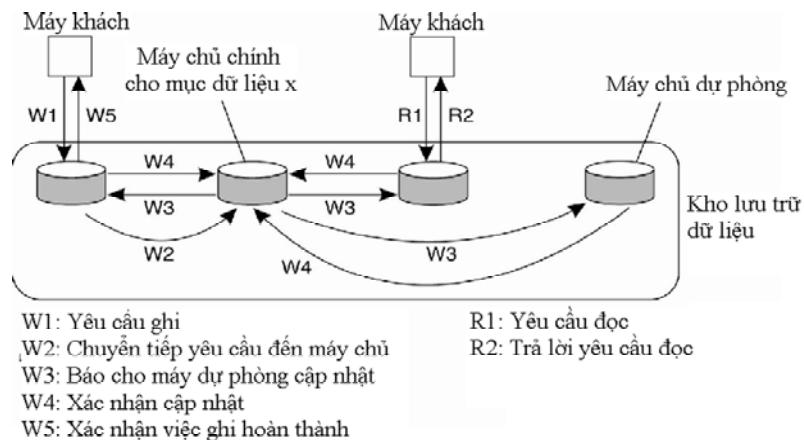
Dựa trên thuật toán bệnh dịch mà có các mô hình lan truyền cập nhật. Điều đáng lưu tâm trong mô hình này là các cập nhật được lan truyền tới các bản sao với càng ít thông điệp càng tốt và càng nhiều bản sao bị "nhiễm" các lan truyền càng nhanh thì càng tốt. Đến cuối cùng nếu bản sao nào mà không lan 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, 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 thì cả hai 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 "kể" 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.

2.8.5.2 Các giao thức dựa trên bản chính

Trong thực tế, các ứng dụng phân tán theo đuổi các mô hình nhất quán tương đối dễ hiểu, đó là những mô hình giới hạn độ chênh lệch trạng thái. Những mô hình được sử dụng khá phổ biến có thể kể đến như mô hình thứ tự nhất quán hoặc mô hình nhất quán tuần tự. Nhân bản dựa trên bản chính thuộc về nhóm nhất quán tuần tự, một bản sao dữ liệu được chỉ định đóng vai trò chủ đạo cập nhật dữ liệu (gọi là bản chính để phân biệt với các bản sao khác), mọi dữ liệu sẽ được cập nhật tại bản chính sau đó mới lan tỏa đến các bản sao khác.

Các giao thức ghi từ xa

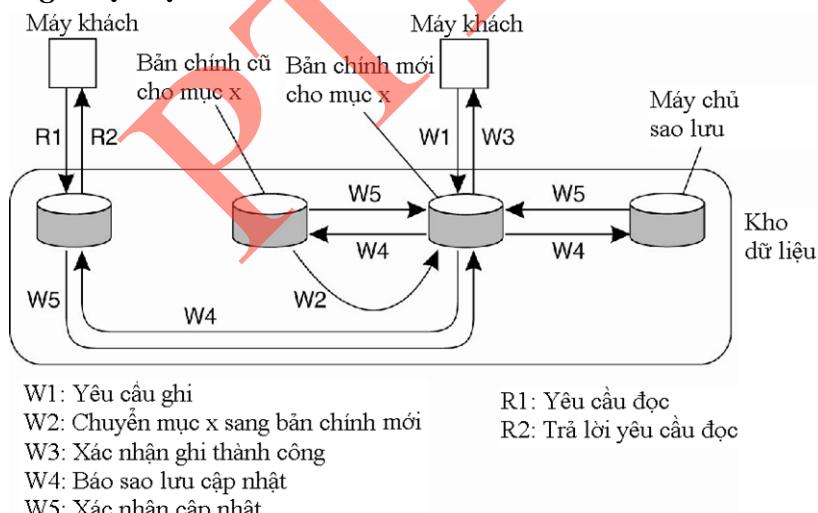
Hệ thống phân tán



Hình 2.72 Các giao thức ghi từ xa

Với giao thức này, tất cả các thao tác ghi được thực hiện chỉ trên một máy chủ từ xa. Giao thức này thường được kết hợp với các hệ thống chủ khách truyền thống. Một dạng giao thức ghi từ xa là giao thức Primary-Backup. Nhược điểm của giao thức này là vẫn 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 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.

Các giao thức ghi cục bộ



Hình 2.73 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 ghi, mục dữ liệu được nhân bản từ máy chủ ở xa chuyển đến máy chủ cục bộ. Việc này được gọi là tiếp cận theo kiểu di trú hoàn toàn. Một vấn đề được đặt ra cho các tiến 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 tiến trình sử dụng nó. Một dạng của giao thức ghi cục bộ là giao thức Primary-backup. Trong

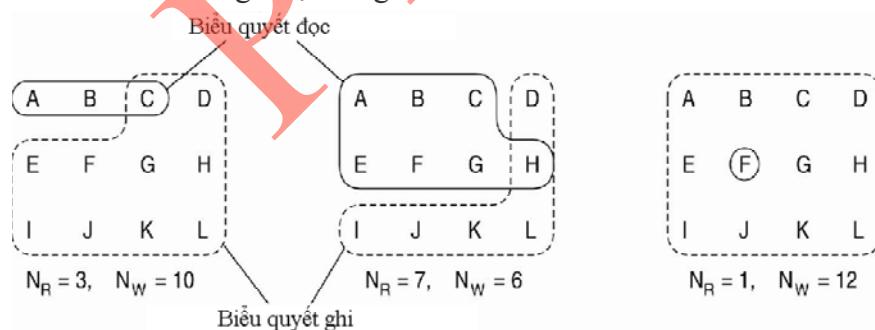
giao thức này bản chính được di trú đến tiến trình đang muốn thực hiện việc cập nhật, rồi sau đó bản dự phòng sẽ được cập nhật.

2.8.5.3 Các giao thức nhân bản ghi

Trong các giao thức này, thay cho việc cập nhật trên một bản sao sau đó mới lan tỏa đến các bản sao khác, thao tác ghi có thể được tiến hành trên nhiều bản sao. Ví dụ: Một tiến trình đặc biệt sẽ chuyển các thao tác cập nhật đến từng bản sao. Một nhãn thời gian Lamport có thể được sử dụng để lấy các thao tác về, 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 tiến trình để gán các số ID duy nhất cho mỗi cập nhật, sau đó truyền các cập nhật tới tất cả các bản sao.

Giao thức nhân bản chủ động cho phép 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 điểm nối điểm hay MULTICAST. Ưu điểm của giao thức này là 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 máy chủ. Tuy nhiên giao thức này lại đòi hỏi phải truyền theo kiểu multicast động hoặc phải có một bộ sắp xếp dãy tập trung mà cả hai phương pháp này đều khó có thể tiếp cận một cách linh hoạt. 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 (máy khách và máy chủ), điều này đảm bảo cho việc chỉ có một lời gọi và một lời đáp được gửi đi.

Giao thức dựa trên đại biểu (Quorum-based) thực hiện các thao tác ghi 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, mỗi lần một mục bị sửa đổi thì số phiên bản của nó cũng được tăng lên.



Hình 2.74 Giao thức dựa trên đại biểu

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 $\frac{1}{2}$ 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 chi phí khi thực hiện phương pháp giao tiếp giữa các nhóm.

2.8.5.4 Các giao thức gắn với cache

Cache là một dạng đặc biệt của nhân bản, nó được điều khiển bởi máy trạm thay vì được điều khiển bởi máy chủ. Có nhiều giải pháp cho việc cache dữ liệu, ví dụ chiến lược phát hiện sự gắn kết sẽ 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, có hai cách thực hiện giải pháp này:

- 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.
- Giải pháp động: 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 máy chủ.

Chiến lược ép buộc sự gắn kết là chiến lược để xác định xem dữ liệu cache được giữ nhất quán với dữ liệu lưu trên máy chủ như thế nào, có hai cách để buộc giữ liệu phải gắn kết với nhau:

- Để cho máy chủ 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:

- Cache chỉ đọc (Read-only Cache): các cập nhật được thực hiện bởi máy chủ (bằng giao thức đẩy) hoặc bởi máy khách (bằng giao thức kéo mỗi khi máy trạm nhận thấy dữ liệu cache đã cũ).
- Cache ghi thẳng (Write-Through Cache): máy khách sẽ thay đổi nội dung của cache sau đó sẽ gửi các cập nhật đến cho máy chủ.
- Cache ghi sau (Write-Back Cache): máy khách 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 máy chủ.

2.8.5.5 Cài đặt nhất quán lấy máy khách làm trung tâm

Nếu bỏ qua yếu tố hiệu năng thì việc cài đặt nhất quán lấy máy khách làm trung tâm sẽ được thực hiện tương đối dễ dàng. Một cách đơn giản, mỗi thao tác ghi sẽ được gán với một định danh toàn cục do máy chủ qui định. Máy chủ phải duy trì hai tập dữ liệu lưu vết thao tác đọc và thao tác ghi của mỗi máy khách. Đối với mô hình nhất quán đọc đều, khi máy khách thực hiện thao tác đọc trên máy chủ, máy chủ sẽ kiểm tra xem tất cả các định danh trên tập đọc đã được thực hiện hay chưa. Nếu một định danh nào đó chưa được thực hiện, máy chủ có thể liên hệ với máy chủ khác để cập nhật dữ liệu mới nhất hoặc chuyển yêu cầu cho máy chủ khác xử lý và sau đó sẽ thực hiện thao tác cập nhật tương ứng với thao tác đọc mà máy khách đã yêu cầu. Như vậy máy chủ cần phải được cài đặt cơ chế phát hiện yêu cầu đọc tương ứng với thao tác ghi trên máy chủ nào (ví dụ máy khách có thể gửi kèm định danh của máy chủ đã thực hiện thao tác ghi). Mô hình nhất quán ghi đều cũng được thực hiện tương tự như mô hình nhất quán đọc đều, nếu máy chủ phát hiện đó là yêu cầu ghi đầu tiên thì nó thực hiện sau đó máy khách bổ sung định danh vào tập ghi của mình.

Để cài đặt mô hình nhất quán đọc kết quả ghi, máy chủ phải kiểm tra xem tất cả các yêu cầu ghi của máy khách đã được thực hiện hay chưa (kể cả trên các máy chủ khác) hoặc máy khách phải tìm kiếm máy chủ mà ở đó thao tác ghi đã được thực hiện.

Hệ thống phân tán

Mô hình ghi theo sau đọc đòi hỏi máy chủ phải đọc dữ liệu và so sánh với dữ liệu của yêu cầu ghi, nếu dữ liệu của yêu cầu ghi mới hơn thì mới thực hiện thao tác ghi.

PTIT

CHƯƠNG 3: CÔNG NGHỆ VÀ CÁCH TIẾP CẬN PHÁT TRIỂN HỆ THỐNG PHÂN TÁN

Hiện nay có nhiều công nghệ hỗ trợ cho việc phát triển các ứng dụng phân tán. Tùy theo đặc tính của hệ thống phân tán, chúng ta có thể lựa chọn các kỹ thuật như dịch vụ web, gọi thủ tục từ xa... Dịch vụ web (Web service) có ưu điểm là dễ phát triển, phù hợp với các ứng dụng trên mạng Internet, tuy nhiên tốc độ xử lý được đánh giá là khá chậm, không đáp ứng yêu cầu thời gian thực.

Java RMI cũng là một kỹ thuật tốt, dễ dàng cài đặt và triển khai cho các hệ thống phân tán, tuy nhiên kỹ thuật này chỉ hỗ trợ ngôn ngữ Java, hiện kỹ thuật này đã được áp dụng phổ biến trong các ứng dụng di động. Mô hình Microsoft DCOM cũng tương tự như Java RMI nhưng nó chỉ cho phép chạy trên nền tảng hệ điều hành Microsoft, đáp ứng yêu cầu thời gian thực.

Kiến trúc CORBA hỗ trợ đa ngôn ngữ, phù hợp cho việc phát triển các ứng dụng phân tán yêu cầu xử lý theo thời gian thực. Trong chương này, chúng ta sẽ tập trung vào mô hình đối tượng thành phần phân tán của Microsoft và kiến trúc môi trường yêu cầu đối tượng chung CORBA. Các mô hình kiến trúc này đều dựa trên phương thức tiếp cận hướng đối tượng phân tán.

3.1 Mô hình gọi thủ tục từ xa

Gọi thủ tục từ xa (RPC) là cách thực cho các thủ tục chạy trên một máy tính giao tiếp với các máy khác. Trong một máy tính riêng lẻ, các thủ tục chạy trên cùng một vùng nhớ, và việc gọi hàm tương đối dễ thực hiện. Gọi thủ tục giữa hai máy tính thông qua kết nối truyền thông nào đó gọi là gọi hàm từ xa. Gọi thủ tục từ xa xuất hiện lần đầu trên các máy tính của Sun Microsystems và Hewlett-Packard chạy trên nền hệ điều hành UNIX.

Các ứng dụng máy khách / máy phục vụ dùng RPC như một cơ chế truyền thông liên hệ thống. Giả sử một ứng dụng thuộc mô hình máy khách / máy phục vụ như một chương trình đã được phân chia, máy phục vụ chạy phần truy xuất dữ liệu bởi vì nó lưu trữ dữ liệu và máy khách chạy phần hiển thị dữ liệu bởi vì nó giao tiếp với người dùng. Theo sự sắp xếp này, RPC được xem như là thành phần kết hợp lại các phần phân chia của chương trình trên mạng.

RPC có khuynh hướng hoạt động trong thời gian thực vì chương trình gọi thường đợi đến khi nhận được trả lời từ chương trình được gọi. RPC được đòi hỏi trong các ứng dụng mà các thủ tục không được tiếp tục tới khi nó nhận được thông tin cần thiết từ hệ thống ở xa. Một biến tấu là sử dụng hoạt động đa luồng trong máy gọi thủ tục để nó có thể tiếp tục với các hoạt động khác trong khi chờ đợi trả lời từ máy được gọi. Một trong các vấn đề khi sử dụng RPC trong môi trường không đồng là các thiết bị khác nhau biểu diễn dữ liệu theo cách khác nhau. RPC tránh vấn đề này bằng cách thêm vào các cuộc gọi thông tin mô tả cách biểu diễn dữ liệu của bên gọi. Khi cuộc gọi được nhận, bên nhận chuyển đổi dữ liệu nếu hai máy biểu diễn dữ liệu khác nhau.

RPC được thiết kế để cung cấp cho việc truyền tải thông tin giữa máy khách và máy chủ dễ dàng hơn, 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 máy khách đến máy chủ. Các thành phần của RPC bao gồm:

Máy khách or máy chủ process: Chương trình hoặc dịch vụ trả lời từ yêu cầu của RPC

RPC stubs: Những hệ thống chương trình con được dùng bởi máy khách hoặc máy chủ khởi động yêu cầu RPC.

Marshalling engine (NDR20 hoặc NDR64): Cung cấp một giao diện chung giữa RPC Máy khách và RPC Máy chủ và được chia làm 2 loại: NDR20 và NDR64. NDR20 được dùng cho hạ tầng 32 bits. Trong khi đó NDR64 được tối ưu dùng cho hạ tầng 64 bits. Máy khách và Máy chủ sẽ thương lượng nên chọn NDR20 hay NDR64 để giao tiếp với nhau

Runtime application programming interface (API): Cung cấp giao diện cho RPC tới Máy kháchs hoặc Máy chủs. Thông thường, RPC Máy kháchs và Máy chủs sẽ gọi hàm giao diện lập trình ứng dụng API để khởi tạo RPC và chuẩn bị cấu trúc dữ liệu sẽ được sử dụng để thực hiện cuộc gọi RPC. Lớp API sẽ quyết định nếu yêu cầu RPC đến từ marshalling engine hoặc trực tiếp từ khách/chủ đến máy chủ nội bộ hoặc máy chủ từ xa. Sau đó lớp API sẽ dẫn đường cho RPC đến Connection RPC, Datagram RPC hoặc Local RPC Layers.

Connection RPC protocol engine: Được sử dụng khi RPC yêu cầu giao thức kết nối. Lớp này sẽ chỉ định sử dụng giao thức kết nối nếu RPC được gửi đi hoặc nhận được một kết nối hướng tới RPC

Datagram RPC protocol engine: Được sử dụng khi RPC yêu cầu giao thức phi kết nối. Lớp này sẽ chỉ định sử dụng giao thức phi kết nối nếu RPC được gửi đi hoặc nhận được một phi kết nối tới RPC

Local RPC protocol engine: Được sử dụng khi Máy chủ và Máy khách đặt trong cùng một host.

Registry: Được truy cập khi dịch vụ RPC đầu tiên được tải về. Các thành phần trong registry sẽ chỉ định ~~dãy~~ port IP và tên thiết bị của các card mạng để RPC có thể kết hợp chúng lại với nhau. Trừ khi API ép buộc RPC phải dùng, Registry sẽ không được sử dụng trong hoạt động của RPC.

Win32 APIs (kernel32.dll, advapi32.dll, ntdll.dll): Kernel32.dll là một file thư viện động 32 bits có trong Windows NT. File này chịu trách nhiệm quản lý bộ nhớ, các hoạt động vào ra của hệ thống Advapi32.dll là file nâng cao của Windows 32 dựa trên giao diện lập trình ứng dụng. File này hỗ trợ về bảo mật và gọi các registry Ntdll.dll là file dll quản lý chức năng các file hệ thống của Windows NT.

SSPI (secur32.dll): Cung cấp giao diện bảo mật cho RPC. File secur32.dll sẽ thương lượng cách dùng cho việc chứng thực và mã hóa như: Kerberos, NTLM, hoặc Secure Sockets Layer (SSL).

Endpoint Mapper (EPM) (rpcss.dll): Rpcss.dll (Remote procedure call subsystem) chủ yếu cung cấp cơ sở hạ tầng cho các dịch vụ COM, nhưng một phần của Rpcss.dll được dùng cho EPM. RPC Máy chủ liên lạc với EPM để nhận những điểm kết thúc động và đăng ký những điểm này vào cơ sở dữ liệu của EPM. Rồi sau đó khi RPC Máy kháchs muốn kết nối tới RPC Máy chủ, nó sẽ liên lạc với EPM để nhờ EPM phân giải những điểm kết thúc.

Active Directory: Chỉ được sử dụng cho quá trình xử lý RPC máy khách khi giao diện bảo mật cụ thể như Kerberos hoặc Negotiate như nhà cung cấp bảo mật hoặc khi Máy chủ dùng NTLM như nhà cung cấp bảo mật.

Network stack: Được sử dụng thông qua các yêu cầu và trả lời của RPC giữa máy khách và máy chủ.

Kernel: Được sử dụng thông qua các yêu cầu và trả lời của RPC giữa Máy khách và Máy chủ

Quá trình xử lý và tương tác của RPC

Các thành phần của RPC sẽ giúp cho các máy trạm xử lý dễ dàng bằng cách gọi hàm nằm trên một chương trình từ xa. Máy khách và máy chủ có một địa chỉ không gian riêng; điều đó có nghĩa là mỗi nguồn tài nguyên bộ nhớ của máy khách và máy cáp phát cho dữ liệu sẽ được dùng bởi hàm. Quá trình xử lý của RPC bắt đầu từ phía Máy khách. Ứng dụng từ phía Máy khách sẽ gọi Máy khách stub thay vì máy khách phải viết code triển khai cho hàm đó. Các stub sẽ được biên soạn và liên kết với các ứng dụng từ phía máy khách trong quá trình phát triển. Thay vì chứa mã code để thực hiện thủ tục gọi hàm từ xa, các code của stub sẽ yêu cầu truy vấn những tham số từ địa chỉ không gian của Máy khách và sau đó chuyển chúng vào thư viện chạy thực của máy khách. Sau đó, thư viện chạy thực của máy khách sẽ biên dịch những tham số cần thiết vào định dạng chuẩn NDR (Network Data Representation) để chuyển giao cho Máy chủ.

Tiếp theo stub của Máy khách sẽ gọi hàm trong thư viện chạy thực của Máy khách (rpcrt4.dll) để gửi các yêu cầu và thông số của nó đến máy chủ. Nếu máy chủ được đặt trong cùng 1 host với máy khách, thư viện chạy thực có thể sử dụng các tính năng của Local RPC (LRPC) và thông qua các yêu cầu của RPC tới Windows kernel cho việc truyền tải đến máy chủ. Nếu máy chủ được đặt ở một host khác, thư viện chạy thực sẽ xác định một giao thức truyền tải thích hợp và thông qua các yêu cầu của RPC đến Network Stack cho việc truyền tải đến máy chủ. RPC có thể dùng các cơ chế trao đổi khác (Interprocess Communications – IPC) như: Name pipes và Winsock để thực hiện truyền tải đến máy chủ.

Khi máy chủ nhận được yêu cầu của RPC (từ phía máy khách trong nội bộ hoặc máy khách từ xa), các hàm trong thư viện chạy thực RPC của máy chủ chấp nhận các yêu cầu và gọi hàm xử lý Máy chủ Stub. Máy chủ stub sẽ truy vấn các tham số từ network buffer và chọn 1 trong 2 loại NDR20 hoặc NDR64 (trong NDR Marshalling Engines), sau đó chuyển đổi chúng từ định dạng truyền tải mạng sang định dạng theo yêu cầu bởi máy chủ. Sau đó các thủ tục từ xa sẽ được chạy, có khả năng xuất ra các tham số và trả về giá trị. Khi các thủ tục từ xa hoàn tất, một chuỗi các bước tương tự sẽ trả về dữ liệu cho máy khách. Các thủ tục từ xa trả dữ liệu của nó về cho Máy chủ Stub, chọn 1 trong 2 loại NDR20 hoặc NDR64 (trong NDR Marshalling Engines), chuyển đổi những tham số được xuất ra thành định dạng truyền tải mạng đến máy khách và trả chúng vào thư viện chạy thực RPC của Máy chủ. Sau đó thư viện chạy thực RPC của Máy chủ sẽ truyền tải dữ liệu đến máy tính của Máy khách bằng LRPC hoặc qua mạng.

Máy khách hoàn tất các thủ tục bằng cách chấp nhận dữ liệu qua mạng và trả dữ liệu về để gọi hàm. Thư viện chạy thực RPC của Máy khách nhận được thủ tục từ xa trả về giá trị, chuyển đổi giá trị từ NDR 20 hoặc NDR64 về định dạng được dùng bởi

Máy khách, và trả chúng về máy khách stub. Đối với Microsoft Windows, thư viện chạy thực được chia làm 2 phần:

Import Library: liên kết với các ứng dụng.

Thư viện chạy thực RPC (RPC Runtime Library): được triển khai như là DLL

3.2 Mô hình DCOM

Để các đối tượng có thể liên lạc với nhau trong môi trường mạng, ActiveX sử dụng mô hình đối tượng thành phần phân tán DCOM đóng vai trò tương tự ORB trong kiến trúc CORBA. Các đặc điểm cơ bản của mô hình DCOM bao gồm:

- Cung cấp các dịch vụ cho ActiveX và OLE.
- Hỗ trợ các giao diện API để gọi các đối tượng động và tĩnh.
- Các đối tượng tương tác với nhau bằng cách sử dụng cơ chế gọi thủ tục từ xa RPC trong môi trường tính toán phân tán.

Mô hình đối tượng thành phần

Mô hình đối tượng thành phần COM (Component Object Model) là một thành phần kiến trúc cơ bản của OLE để xác định giao diện giữa các đối tượng thành phần.

Mô hình đối tượng thành phần phân tán

Mô hình đối tượng thành phần phân tán DCOM (Distributed Component Object Model) là mô hình mở rộng của COM nhằm hỗ trợ truyền thông giữa các đối tượng trên nhiều máy khác nhau qua mạng LAN, WAN, Internet.

Các đặc trưng cơ bản của DCOM

DCOM là một mô hình nhằm hỗ trợ việc xây dựng các ứng dụng phân tán trên cơ sở mô hình đối tượng, với việc dùng các phần mềm trung gian (middleware). Với tiêu chí đó, DCOM được thiết kế có các đặc trưng cơ bản như sau:

- Độc lập vị trí (Location independence)
- Độc lập ngôn ngữ (Language neutrality)
- Quản lý kết nối (Connection management)
- Khả năng thay đổi quy mô (Scalability)
- Hiệu năng (Performance)
- Bảo mật (Security)
- Cân bằng tải (Load balancing)
- Tính chịu lỗi (Fault tolerance)

Kiến trúc DCOM

DCOM là mô hình phát triển kết tiếp của COM. COM xác định phương thức kết nối của các thành phần với các chương trình máy trạm. Sự tương tác này có thể là trực tiếp, không cần thành phần hệ trung gian. DCOM có một số ưu điểm sau:

- Hỗ trợ các giao diện ngôn ngữ hướng đối tượng
- DCOM sẵn có trong các hệ điều hành Windows
- Có khả năng giao tiếp dễ dàng với DCE.
- Các mạng tương thích với DCOM có giá thành thấp
- Có sự phân biệt rõ ràng của các giao diện và cài đặt

Cài đặt DCOM lại phụ thuộc nền tảng hệ thống (phần cứng, hệ điều hành), các đối tượng và các giao diện được lập trình phức tạp, không trực quan.

3.3 Kiến trúc CORBA

Kiến trúc CORBA (Common object request broker architecture) do OMG (Object Management Group) đưa ra từ năm 1990. OMG cung cấp các sườn kiến trúc chung cho lập trình hướng đối tượng, trong đó mô tả rõ các kiến trúc quản lý đối tượng (OMA). OMA bao gồm: các chức năng ORB, các dịch vụ đối tượng, các phạm vi giao diện, các đối tượng ứng dụng. CORBA được xây dựng và phát triển dựa trên OMA nhằm mục đích chuẩn hoá việc xây dựng các ứng dụng phân tán. Trong kiến trúc CORBA các đối tượng phân tán được định nghĩa để có thể lưu giữ và gọi từ xa, do đó nó phải đảm bảo tất cả các nhiệm vụ chung liên quan tới việc lập trình trên mạng bao gồm việc đăng ký đối tượng, gửi/nhận yêu cầu trên mạng...

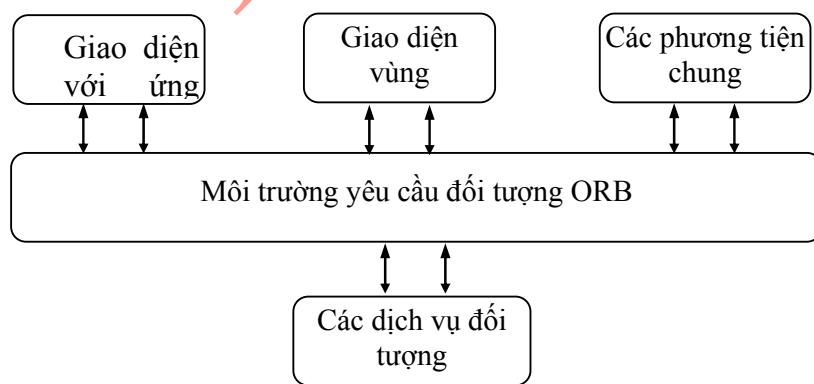
Kiến trúc CORBA do nhóm OMG¹ xây dựng và phát triển dựa trên kiến trúc quản lý đối tượng OMA nhằm mục đích chuẩn hoá việc xây dựng các ứng dụng phân tán. Trong kiến trúc CORBA các đối tượng phân tán được định nghĩa để có thể lưu giữ và gọi từ xa, do đó nó phải đảm bảo tất cả các nhiệm vụ chung liên quan tới việc lập trình trên mạng bao gồm việc đăng ký đối tượng, gửi/nhận yêu cầu trên mạng...

3.3.1 Các thành phần cơ bản của CORBA

CORBA bao gồm các tiêu chuẩn hỗ trợ việc phát triển các ứng dụng theo mô hình hướng đối tượng, trong đó các đối tượng có thể thực hiện trên một bộ xử lý hoặc được phân tán trên mạng.

Để đạt được mục tiêu tích hợp các hệ thống² của các nhà cung cấp khác nhau, kiến trúc CORBA đảm bảo ba đặc điểm quan trọng sau:

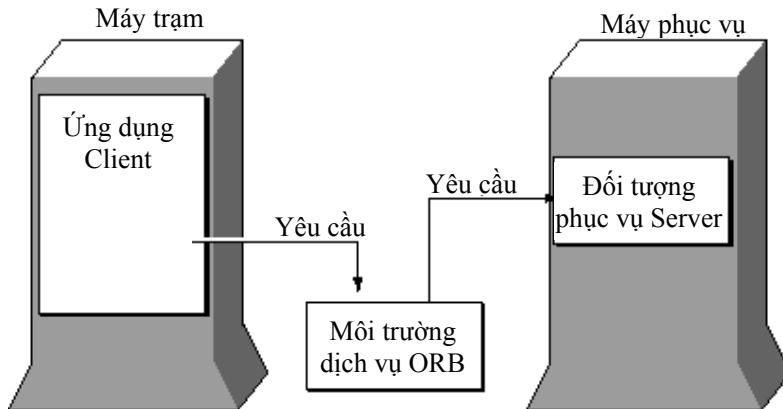
- Không phụ thuộc hệ điều hành và thiết bị phần cứng
- Không phụ thuộc ngôn ngữ lập trình
- Không phụ thuộc vị trí vật lý cài đặt các ứng dụng



Hình 3.1 Các thành phần cơ bản trong kiến trúc CORBA

¹ OMG (Object Management Group) được thành lập năm 1989, hiện nay nhóm này có khoảng 800 thành viên là các công ty hoạt động trong lĩnh vực máy tính – viễn thông

Các đối tượng trong kiến trúc CORBA được thể hiện và cài đặt tương tự như các lớp trong ngôn ngữ lập trình hướng đối tượng do đó có tính tương thích cao và hoàn toàn độc lập với hệ điều hành. Nhằm mục đích chuẩn hóa kiến trúc này, nhóm OMG đưa ra các đặc tả giao diện, hình 3.1 thể hiện các thành phần cơ bản trong kiến trúc CORBA.



Hình 3.2 Quá trình thực hiện yêu cầu của Máy trạm

Thành phần cơ bản trong kiến trúc CORBA là môi trường yêu cầu đối tượng ORB, nó cung cấp các giao diện, dịch vụ và các phương tiện phục vụ cho các đối tượng CORBA. Khi ORB nhận được yêu cầu của Máy trạm nó có nhiệm vụ chuyển yêu cầu đó đến Máy chủ thích hợp (hình 3.2). Theo cách lập trình Máy trạm/Máy chủ truyền thông, Máy trạm phải biết tên và vị trí của máy chủ, phương pháp thiết lập và duy trì kênh liên lạc, phương pháp gửi/nhận các tham số. Trong CORBA, Máy trạm chỉ cần biết tên của thủ tục cần gọi và các tham số cần thiết để gửi/nhận, mọi thao tác khác do các thành phần của CORBA đảm nhiệm.

Các dịch vụ đối tượng: là các giao diện không phụ thuộc giao diện vùng để các đối tượng phân tán sử dụng (ví dụ dịch vụ xác định tên đối tượng, dịch vụ bảo mật...), các dịch vụ này được cài đặt độc lập với ORB.

Các phương tiện chung: là các phương tiện dùng cho tất cả các loại ứng dụng trên cho sản phẩm của các hãng khác nhau.

Giao diện vùng: thực hiện vai trò tương tự như các dịch vụ đối tượng và các phương tiện chung nhưng tập trung vào các lĩnh vực ứng dụng riêng, ví dụ CORBA cho viễn thông, CORBA cho tài chính, ngân hàng...

Các giao diện ứng dụng: là các giao diện riêng cho các ứng dụng đã xác định, giao diện này chưa được chuẩn hóa

Môi trường ORB

Môi trường ORB là thành phần cơ bản trong kiến trúc CORBA, đó là phần mềm nằm giữa lớp ứng dụng và lớp giao vận (hình 3.3). Bản thân ORB không thuộc thành phần của hệ điều hành mà là một dạng phần mềm trung gian dưới sự kiểm soát của hệ điều hành. Môi trường ORB đảm nhiệm các công việc sau:

- Sẵn sàng nhận các yêu cầu của Máy trạm

Hệ thống phân tán

- Cung cấp tất cả các thủ tục cần thiết để tìm ra đối tượng thực hiện yêu cầu của Máy trạm.
- Trao đổi dữ liệu và gửi yêu cầu đến Servant
- Cung cấp một số dịch vụ khi Servant yêu cầu trong thời gian thực hiện

Ứng dụng Máy trạm chỉ cần liên hệ với giao diện giao diện hoàn toàn độc lập với vị trí cài đặt đối tượng thực hiện.

Máy trạm: là đối tượng yêu cầu dịch vụ.

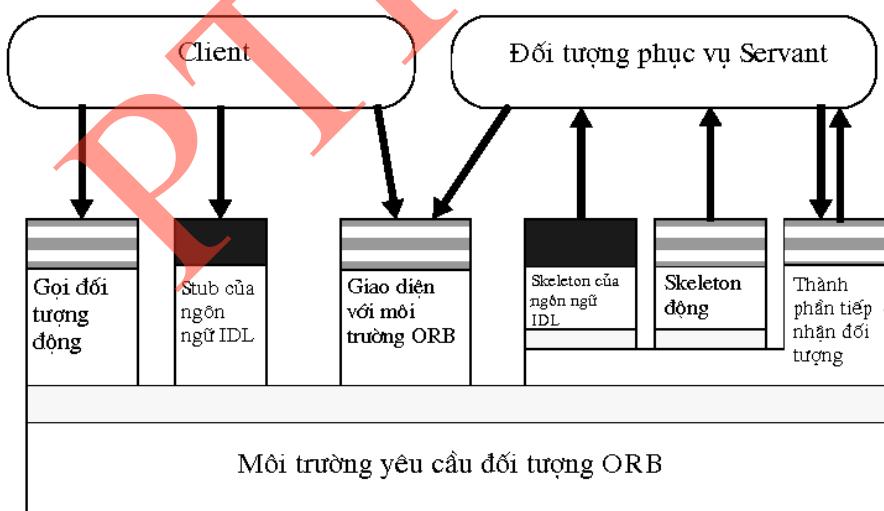
Servant: là các đối tượng phục vụ yêu cầu của các Máy trạm, nó định nghĩa các thao tác, hỗ trợ giao diện IDL của CORBA.

ORB: là môi trường trung gian thiết lập quan hệ Máy trạm/Máy chủ giữa các đối tượng. bằng cách sử dụng ORB, đối tượng Máy trạm có thể gọi các hàm trên máy cục bộ hoặc trên mạng.

Giao diện với môi trường ORB: Cung cấp các giao diện để Máy trạm và Servant kết nối với nhau qua môi trường ORB.

Stub và Skeleton của ngôn ngữ IDL: Stub gồm các lệnh cho phép Máy trạm truy nhập tới các thành phần của Máy chủ tương tự như cơ chế gọi gọi thủ tục từ xa RPC. Skeleton gồm các lệnh trên Máy chủ để liên lạc với các thành phần CORBA, nó đóng vai trò cầu nối giữa ORB với đối tượng thực hiện trên Máy chủ.

Thành phần Object Adapter (BOA hoặc POA): Hỗ trợ ORB trong việc phân phát các yêu cầu đến các đối tượng thực hiện.



Hình 3.3 Cấu trúc và giao diện của ORB

Khi có yêu cầu, máy khách sử dụng giao diện gọi đối tượng động DII hoặc sử dụng Stub của ngôn ngữ định nghĩa giao diện IDL, đối với một số chức năng Máy trạm có thể trực tiếp tương tác với môi trường ORB. Môi trường ORB chuyển yêu cầu của Máy trạm đến đối tượng phục vụ Servant thông qua IDL Skeleton tĩnh hoặc Skeleton động.

Có hai phương pháp định nghĩa giao diện: phương pháp tĩnh và phương pháp động. Theo phương pháp thứ nhất, giao diện được định nghĩa trong ngôn ngữ IDL để qui định các loại đối tượng và các thao tác trên đối tượng. Phương pháp thứ hai sử dụng dịch vụ kho lưu trữ giao diện (Interface Repository) để thể hiện các thành phần của giao diện như các đối tượng phục vụ và cho phép truy nhập các thành phần này trong thời gian chạy. Đối với môi trường ORB thì phương pháp sử dụng ngôn ngữ IDL hay sử dụng dịch vụ kho giao diện đều có ý nghĩa như nhau.

Ứng dụng máy khách khởi tạo yêu cầu bằng cách gọi các thủ tục trong Stub liên quan đến đối tượng hoặc bằng cách tạo yêu cầu theo phương pháp động. Môi trường ORB xác định mã thực hiện thích hợp, chuyển các tham số và chuyển quyền điều khiển cho đối tượng phục vụ Servant thông qua Skeleton của ngôn ngữ IDL hoặc qua Skeleton động. Trong quá trình thực hiện đối tượng Servant có thể sử dụng một số dịch vụ của môi trường ORB thông qua thành phần thính nghỉ đối tượng. Khi thực hiện xong yêu cầu, quyền điều khiển và các giá trị trả về sẽ được chuyển cho ứng dụng Máy trạm.

Việc cài đặt môi trường ORB phải dựa trên các yêu cầu của hệ thống phân tán, đối với các hệ thống hạn chế về mặt tài nguyên nên cài đặt ORB tối thiểu và các ứng dụng phân tán sử dụng phương pháp tĩnh (sử dụng Máy trạm stub và Máy chủ skeleton). Nếu có môi trường truyền thông thích hợp thì có thể cài đặt môi trường ORB thường trú trong ứng dụng Máy trạm và trong đối tượng phục vụ Servant. Stub trong Máy trạm có thể sử dụng cơ chế trao đổi thông tin liên tiến trình hoặc sử dụng dịch vụ xác định vị trí để thiết lập thông tin với đối tượng thực hiện trên Máy chủ. Môi trường ORB có thể là chương trình bình thường cài đặt trên máy Máy chủ giống như hệ điều hành đang sử dụng, liên lạc với Máy trạm và Servant vẫn được đảm bảo thông qua cơ chế trao đổi thông tin liên tiến trình. Để bảo đảm tính bảo mật, khả năng mở rộng và hiệu suất hoạt động của hệ thống, có thể cài đặt môi trường ORB như một dịch vụ của hệ điều hành đang sử dụng.

Ứng dụng máy khách

Theo kiến trúc của CORBA, ứng dụng Máy trạm sử dụng tham chiếu đối tượng để gọi các hàm do đối tượng cung cấp. Để gọi một đối tượng cần phải tiến hành các bước như: xác định đối tượng sẽ gọi, hàm thực hiện và các tham số cung cấp cho hàm cũng như giá trị trả về của nó. Môi trường ORB quản lý các quá trình chuyển quyền điều khiển và chuyển dữ liệu giữa đối tượng thực hiện và ứng dụng Máy trạm, ứng dụng Máy trạm gọi các hàm của đối tượng phục vụ tương tự như gọi các hàm trên máy cục bộ.

Đối tượng thực hiện

Đối tượng thực hiện là các đối tượng bao hàm cả dữ liệu và các thao tác thực hiện trên dữ liệu, theo cách hiểu trong lập trình hướng đối tượng đây là các thể hiện (instances). Chức năng thực hiện của đối tượng này hoàn toàn độc lập với môi trường ORB (môi trường này chỉ đảm nhiệm liên kết giữa máy khách và đối tượng thực hiện). Tham chiếu đối tượng (Object Reference) là thao tác nhận thông tin cần thiết dụng để xác định một đối tượng thực hiện trong môi trường ORB (tương tự như mô hình lập trình trên máy tính đơn lẻ, có thể hình dung thao tác này tương tự như thao tác lấy địa chỉ của đối tượng phục vụ). Để gọi các thao tác trên đối tượng phục vụ, Máy trạm phải nhận được tham chiếu đối tượng do môi trường ORB cung cấp. Môi trường ORB s

dụng phương pháp ánh xạ như nhau đối với tất cả các ngôn ngữ lập trình, vì vậy ứng dụng viết trên các ngôn ngữ khác nhau hoàn toàn có thể giao tiếp với nhau trong môi trường ORB của kiến trúc CORBA.

Các yêu cầu cài đặt nói chung phụ thuộc nhiều vào ba yếu tố: thiết bị phần cứng, hệ điều hành và ngôn ngữ lập trình sử dụng để viết mã lệnh. Thành phần thích nghi đối tượng (Object Adapter) là phương tiện chủ yếu để Servant truy nhập các dịch vụ của ORB, nó đảm nhiệm các chức năng sau:

- Tạo và biên dịch tham chiếu đối tượng
- Gọi các hàm của đối tượng phục vụ
- Bảo mật các tương tác giữa ứng dụng Máy trạm và các ứng dụng phục vụ
- Kích hoạt và huỷ các đối tượng thực hiện
- Ánh xạ tham chiếu đối tượng Máy trạm với các đối tượng phục vụ tương ứng
- Đăng ký việc cài đặt đối tượng phục vụ

Trong hệ thống có thể tồn tại nhiều thành phần thích nghi đối tượng và nó chỉ thích hợp cho một số loại đối tượng riêng.

Thành phần thích nghi đối tượng cơ bản (Basic Object Adapter - BOA) được định nghĩa trong phiên bản CORBA 1.0 với mục tiêu đơn giản và đáp ứng các mục đích cơ bản của Object Adapter, do đó nó chỉ bao hàm các thao tác chính thể hiện trên hình 3.4 (viết trên ngôn ngữ IDL).

```
interface BOA {
    Object create (in ReferenceData id,
                   in InterfaceDef intf,
                   in ImplementationDef impl);
    void dispose (in Object obj);

    ReferenceData get_id (in Object obj);

    void change_implementation (in Object obj,
                                in ImplementationDef impl);

    void impl_is_ready (in ImplementationDef impl);
    void deactivate_impl (in ImplementationDef impl);
    void obj_is_ready (in Object obj,
                        in ImplementationDef impl);
    void deactivate_obj (in Object obj);
};
```

Hình 3.4 Đặc tả Basic Object Adapter

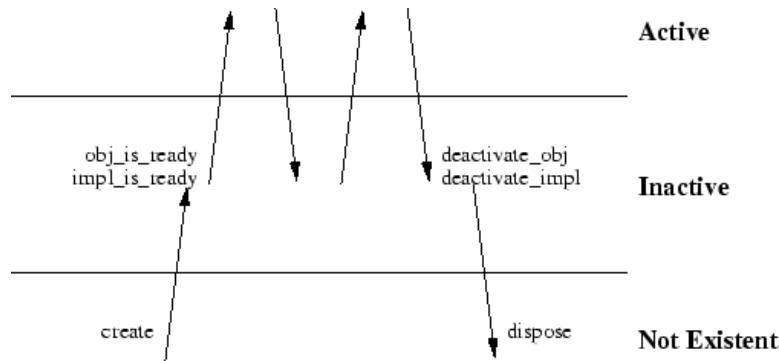
Thành phần thích nghi đối tượng cơ bản định nghĩa ba trạng thái mà đối tượng có thể nhận được:

Trạng thái không tồn tại (Not Existential): đối tượng không tồn tại, môi trường ORB không xác định được đối tượng, vì vậy khi có yêu cầu gọi môi trường ORB sẽ từ chối phục vụ và gửi kèm theo thông báo lỗi.

Trạng thái tồn tại: Đối tượng đã được tạo ra và môi trường ORB nhận biết được đối tượng, tuy nhiên đối tượng có thể ở hai trạng thái sau:

Trạng thái không kích hoạt (inactive): ứng dụng Máy trạm có thể nhận được tham chiếu đến đối tượng nhưng không thể thực hiện các thao tác thực hiện trên đối tượng.

Trạng thái kích hoạt (active): ứng dụng Máy trạm có thể sử dụng tất cả các dịch vụ do đối tượng cung cấp.



Hình 3.5 Chu kỳ sống của đối tượng sử dụng BOA

Sự thay đổi trạng thái của các **đối tượng** được thực hiện trên máy chủ, hình 3.5 thể hiện chu kỳ sống của đối tượng và các hàm làm thay đổi trạng thái của nó. Thành phần thích nghi đối tượng động (Portable Object Adapter - POA) là thành phần trung gian giữa Object Implementation và môi trường ORB, nó được thiết kế không những để thay thế BOA mà còn đảm bảo hai đặc tính quan trọng sau:

Khả năng dễ dàng chuyển đổi: Cho phép các ứng dụng chạy trên các môi trường ORB khác nhau mà không cần thay đổi mã nguồn.

Tính linh hoạt: Cho phép kiểm soát chu kỳ sống và tính sẵn sàng của đối tượng phục vụ trong việc nhận các yêu cầu từ phía Máy trạm.

Mục tiêu của POA bao gồm:

Cho phép lập trình viên xây dựng cài đặt đối tượng dễ dàng chuyển đổi giữa các sản phẩm ORB khác nhau.

Cho phép cài đặt đối tượng để cung cấp dịch vụ cho các đối tượng mà thời gian tồn tại của nó vượt quá thời gian phục vụ.

Hỗ trợ việc kích hoạt trong suốt các đối tượng

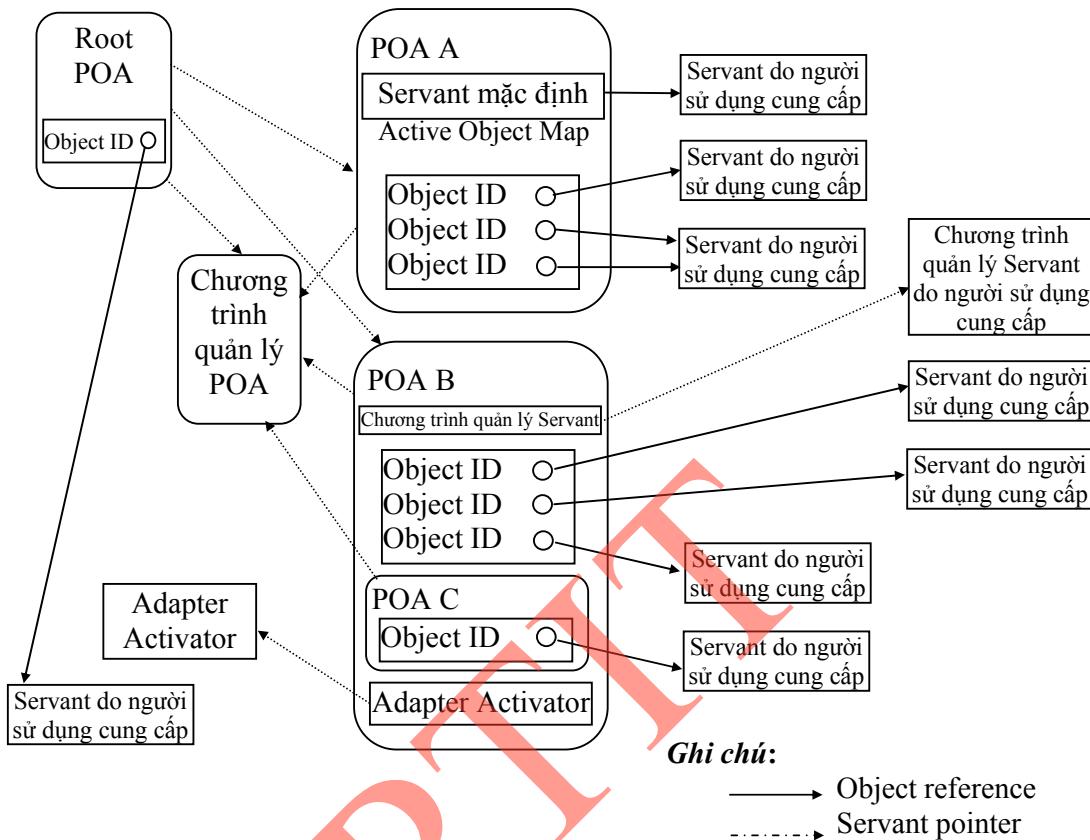
Cho phép một Servant đồng thời hỗ trợ nhiều Object ID

Như vậy, trên đối tượng phục vụ có thể tồn tại nhiều POA, chúng được tổ chức theo phân cấp trong đó Root POA do môi trường ORB tạo ra, các POA khác do người sử dụng tạo lập và là các POA con. Hình 9.6 thể hiện kiến trúc và tương tác giữa các thành phần khác nhau của POA, chức năng của mỗi thành phần như sau:

Servant: Servant là các mã lệnh thể hiện các đối tượng phục vụ và được cài đặt trong các mô đun cung cấp dịch vụ.

Object ID: Object ID là giá trị do POA sử dụng để xác định một đối tượng CORBA, các giá trị này do POA quản lý.

Object reference: Object reference là thao tác trong mô hình đối tượng CORBA nhằm mục đích nhận tên của POA và Object ID (để đơn giản có thể hiểu thao tác này tương tự như việc lấy địa chỉ của một đối tượng).



Hình 3.6 Kiến trúc của POA

Chương trình quản lý POA: Chương trình quản lý POA là đối tượng phục vụ cho việc quản lý trạng thái xử lý của POA, bằng cách sử dụng các thao tác của chương trình quản lý này có thể thực hiện việc huỷ bỏ hoặc đưa vào hàng đợi các yêu cầu liên quan với POA. Thông qua chương trình quản lý POA, lập trình viên có thể đưa POA về trạng thái không hoạt động.

Chương trình quản lý Servant: Chương trình quản lý Servant là một đối tượng mà người phát triển ứng dụng có thể gán với chương trình quản lý POA, môi trường ORB gọi các thao tác trên chương trình quản lý Servant để kích hoạt Servant theo yêu cầu hoặc đưa Servant về trạng thái không hoạt động. Chương trình quản lý Servant có nhiệm vụ quản lý mối liên kết đối tượng (đặc trưng bằng giá trị của Object ID) với Servant tương ứng và quyết định sự tồn tại của đối tượng.

Adapter Activator: Adapter Activator là đối tượng mà người phát triển ứng dụng có thể gán với POA. Môi trường ORB gọi thao tác trên Adapter Activator khi có yêu cầu đối với POA con nhưng POA con đó không tồn tại, Adapter Activator có thể tạo POA cần thiết theo yêu cầu.

Mỗi POA có một bản đồ đối tượng hoạt động (Active Object Map) dùng để theo dõi Object ID của các đối tượng hoạt động và các Servant hoạt động tương ứng. Mỗi Servant được ánh xạ tương ứng với một hoặc nhiều Object ID trong bản đồ đối tượng hoạt động, việc xác định Servant và gán với đối tượng do trình quản lý Servant đảm nhiệm.

Nhiệm vụ đồng bộ giữa các POA do chương trình quản lý POA thực hiện, chương trình quản lý này kiểm soát tính sẵn sàng của các POA nhận yêu cầu từ phía Máy trạm. Ngoài ra chương trình quản lý POA còn cung cấp các khả năng mở rộng để người sử dụng có thể tác động đến tiến trình xử lý yêu cầu:

Chương trình quản lý đối tượng phục vụ có thể điều khiển và giám sát chu kỳ sống của các đối tượng phục vụ.

Một đối tượng phục vụ mặc định có thể đồng thời phục vụ nhiều đối tượng

Trong trường hợp cần thiết, có thể sử dụng Adapter Activator để tạo POA mới.

Khi nhận được yêu cầu từ phía Máy trạm, POA lựa chọn Servant và thực hiện việc gọi Servant đó. Trên Máy chủ có thể tồn tại nhiều POA nhưng ít nhất có một POA gọi là root POA do môi trường ORB cung cấp, từ root POA này có thể tạo thêm các POA con và thiết lập cấu hình cho chúng để thực hiện các hành vi khác nhau đồng thời cũng có thể định nghĩa các tính chất của đối tượng do POA kiểm soát.

Mỗi POA có tập các chính sách riêng nhằm mục đích thay đổi các thuộc tính của POA, các chính sách đó do người sử dụng lựa chọn khi tạo lập POA và không thể thay đổi trong thời gian tồn tại của nó. Bản thân root POA đã có tập các chính sách được qui định trước và không thể thay đổi các chính sách đó, để có hành vi khác cần phải tạo ra POA con và thiết lập các chính sách cho nó, POA con không thừa kế các chính sách của POA cha. CORBA 2.0 qui định 7 chính sách của POA sau:

Chính sách về luồng xử lý (Thread Policy): Qui định POA sử dụng trong môi trường đa luồng (multi-thread) hay đơn luồng (single thread). Trong chế độ đa luồng, POA xử lý đồng thời nhiều yêu cầu. Trong chế độ đơn luồng, các yêu cầu lần lượt được POA xử lý.

Chính sách về phạm vi tồn tại (Lifespan Policy): thuộc tính này có thể nhận giá trị **TRANSIENT** hoặc **PERSISTENT**. Khi có yêu cầu từ phía Máy trạm, nếu POA ở trạng thái không kích hoạt và POA được đặt thuộc tính **TRANSIENT** thì sẽ trả về lỗi **OBJECT_NOT_EXIST**, trường hợp POA có thuộc tính **PERSISTENT** thì môi trường ORB sẽ tìm kiếm POA dựa trên tên và các POA cha hoặc thông báo cho dịch vụ định vị để tạo hoặc xoá bỏ POA

Chính sách về tính duy nhất của tên định danh đối tượng (Object Id Uniqueness Policy): Qui định một đối tượng phục vụ có thể đăng ký với POA theo một hoặc nhiều tên định danh

Chính sách gán tên định danh (Id Assignment Policy): Qui định việc đặt tên định danh cho đối tượng phục vụ do POA hay do người sử dụng quyết định. Khi kích hoạt một đối tượng, tên định danh và đối tượng phục vụ tương ứng của nó thường được lưu trong bản đồ đối tượng hoạt động.

Chính sách duy trì đối tượng phục vụ (Servant Retention Policy): Qui định đối tượng phục vụ đã kích hoạt có được đăng ký trong bản đồ đối tượng hoạt động hay không.

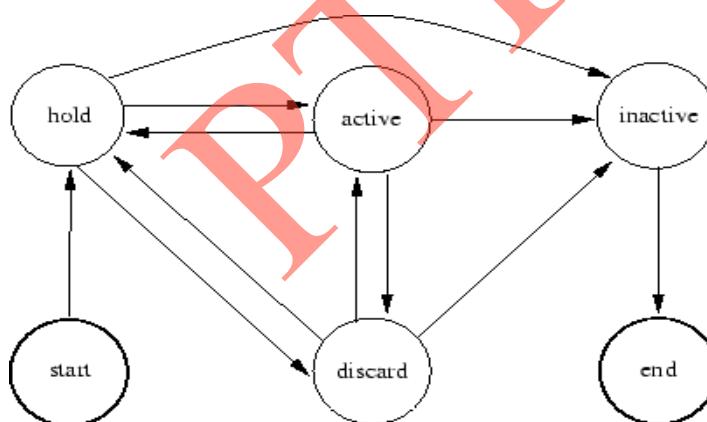
Chính sách về xử lý yêu cầu (Request Processing Policy): Qui định phương pháp xử lý khi nhận được yêu cầu từ phía Máy trạm theo một trong ba cách sau:

- POA tìm kiếm đối tượng phục vụ yêu cầu trong bản đồ đối tượng kích hoạt.
- Chuyển yêu cầu đến chương trình quản lý đối tượng phục vụ .
- Sử dụng đối tượng phục vụ mặc định.

Chính sách kích hoạt ngầm (Implicit Activation Policy): Một số thao tác trên đối tượng phục vụ đòi hỏi đối tượng này phải ở trạng thái hoạt động (ví dụ thao tác nhận tham chiếu đối tượng), nếu thiết lập chính sách kích hoạt ngầm sẽ loại bỏ thông báo lỗi trong các trường hợp trên.

Các chính sách của POA có mối liên hệ với nhau, việc đặt giá trị cho chính sách này có thể ảnh hưởng tới việc thiết lập các chính sách khác.

Thành phần thích nghi đối tượng cơ bản (BOA) không có cơ chế đồng bộ các đối tượng phục vụ và không có khả năng kiểm soát tinh sẵn sàng tiếp nhận các yêu cầu, nhược điểm này đã được POA khắc phục bằng cách đưa ra chương trình quản lý POA. Thực chất chương trình quản lý kiểm soát các trạng thái và các vấn đề liên quan đến việc chuyển trạng thái của POA, hình 9.7 thể hiện 4 trạng thái của POA và các bước chuyển trạng thái của chúng.



Hình 3.7 Sơ đồ chuyển trạng thái của POA

Trạng thái kích hoạt (Active): Các yêu cầu từ phía Máy trạm sẽ được chuyển đến đối tượng phục vụ tương ứng để xử lý.

Trạng thái chờ (Holding): Các yêu cầu không được xử lý ngay mà chuyển đến hàng đợi để chờ xử lý.

Trạng thái từ chối (Discarding): Các yêu cầu đều bị từ chối phục vụ và ứng dụng Máy trạm nhận được mã lỗi TRANSIENT báo hiệu đã xảy ra sự cố tạm thời.

Trạng thái không kích hoạt (Inactive): Tất cả các yêu cầu đều bị từ chối phục vụ.

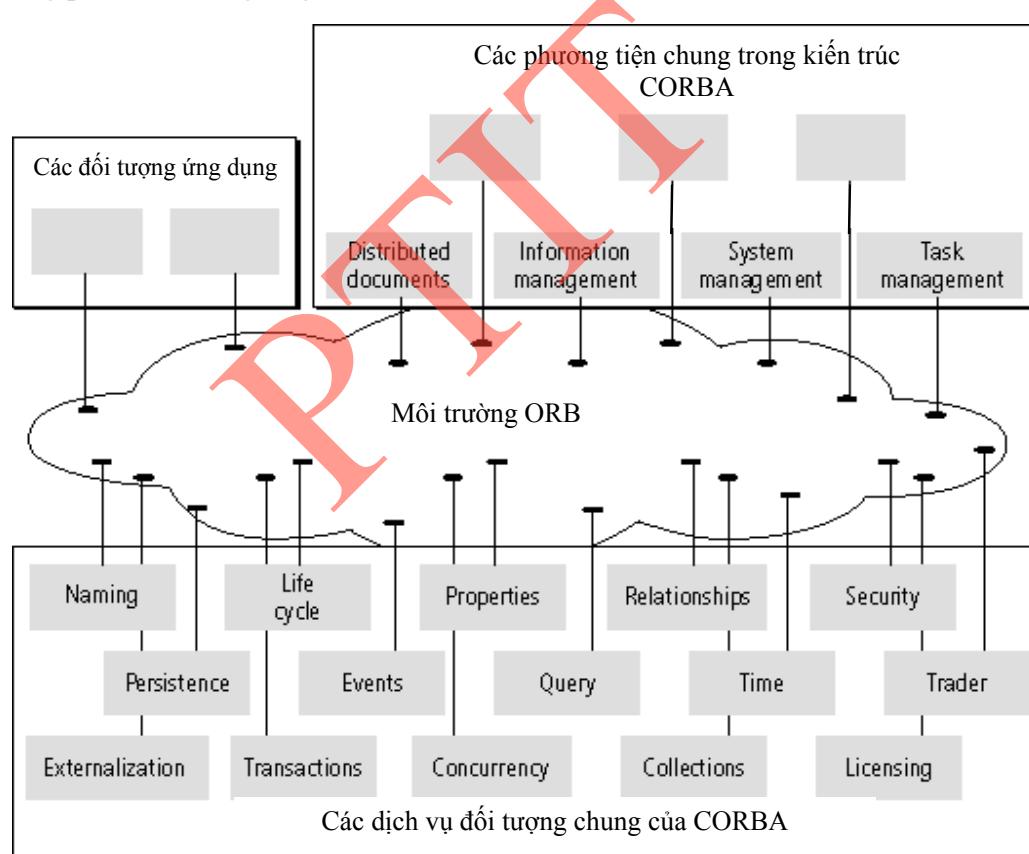
Thao tác thiết lập POA và gán các đối tượng phục vụ được thực hiện theo các bước sau:

Nhận tham chiếu đến root POA: Tất cả các ứng dụng Máy chủ phải nhận tham chiếu đến rootPOA để phục vụ cho việc quản lý các đối tượng hoặc tạo các POA mới. RootPOA do môi trường ORB quản lý và được khởi tạo bằng cách tạo đối tượng có tên là **rootPOA**.

Định nghĩa các chính sách cho POA: Qui định các chính sách cụ thể đối với POA phục vụ cho các yêu cầu riêng. Nếu không thiết lập các tham số chính sách, POA sẽ nhận các tham số mặc định.

Tạo POA con của root POA: Trong một ứng dụng không hạn chế số lượng POA, tuy nhiên POA con không kế thừa các chính sách của POA cha.

Tạo và kích hoạt đối tượng phục vụ: Mã lệnh thực hiện của đối tượng phục vụ phải được cài đặt trên máy chủ, để máy khách sử dụng các dịch vụ cần phải kích hoạt Servant bằng cách đăng ký tên định danh của đối tượng phục vụ trong bản đồ đối tượng hoạt động - thực chất đó là bảng dùng để ánh xạ tên định danh với đối tượng phục vụ tương ứng.



Hình 3.8 Các dịch vụ và các phương tiện của CORBA

Bình thường khi POA mới được tạo lập, trình quản lý POA ở trạng thái nghỉ (holding) - trong trạng thái này tất cả các yêu cầu đều được chuyển đến hàng đợi nhưng không được xử lý, để xử lý các yêu cầu đó, trình quản lý POA phải chuyển về

trạng thái kích hoạt. Trình quản lý POA thực chất là một đối tượng kiểm soát trạng thái của POA (trạng thái chờ, xử lý hay chối bỏ), nó gắn với POA ngay khi mới tạo lập.

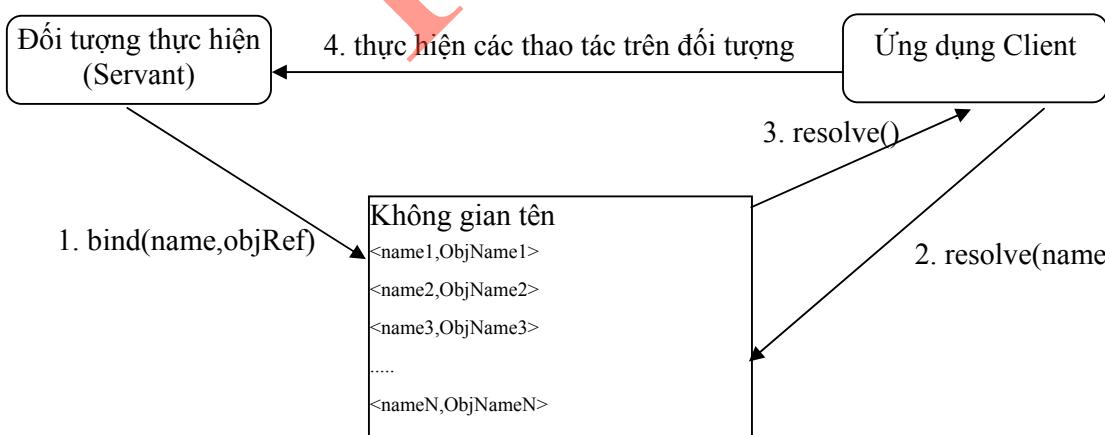
Trong giao diện tĩnh, đối tượng Máy trạm sử dụng các thủ tục Stub để gọi các hàm trong đối tượng phục vụ. Đối với giao diện động DII, đối tượng Máy trạm gọi các hàm của đối tượng phục vụ bằng cách sử dụng các thông tin lưu giữ trong kho giao diện (Interface Repository). Giao diện Skeleton động cho phép các mã lệnh thực hiện trên Servant đăng ký với môi trường ORB các hàm thực hiện bằng cách mô tả các tham số của các thao tác. Việc sử dụng giao diện này không nhất thiết đòi hỏi giao diện gọi đối tượng động DII tương ứng trên phía Máy trạm.

Môi trường ORB tạo điều kiện thuận lợi cho việc trao đổi thông tin giữa các đối tượng và để các đối tượng xác định nhau. Tuy nhiên các tính năng đó chưa đủ để xây dựng các ứng dụng phân tán qui mô lớn, đặc biệt đối với các ứng dụng thích hợp cho chuyên ngành hẹp, vì vậy hãng OMG đã đưa ra một số tính năng mới thể hiện dưới dạng các dịch vụ và các phương tiện (hình 3.8).

Nhóm OMG chỉ nêu đặc tả các dịch vụ (giao diện mà các dịch vụ cung cấp) mà không thể hiện cách cài đặt các dịch vụ như thế nào, các dịch vụ đó độc lập với môi trường ORB và do đó số lượng dịch vụ phụ thuộc vào sản phẩm CORBA của từng hãng.

Dịch vụ đặt tên

Dịch vụ đặt tên (Naming Service) là một trong các dịch vụ cơ bản nhất trong các dịch vụ của kiến trúc CORBA nói riêng và trong các ứng dụng mạng nói chung, nó cho phép các ứng dụng Máy trạm xác định được đối tượng phục vụ bằng cách cung cấp tên sau đó sẽ nhận được tham chiếu đến đối tượng (Object Reference). Dịch vụ này cho phép các đối tượng Servant được đăng ký và xác định bằng tên, nó sử dụng "naming context" để bảo đảm tập các tên là duy nhất. Sự kết hợp giữa tên với đối tượng gọi là ràng buộc tên "Name binding" và luôn được định nghĩa liên quan với ngữ cảnh của tên.



Hình 3.9 Mô hình dịch vụ đặt tên

Dịch vụ đặt tên cho phép kết hợp một hoặc nhiều tên logic với một tham chiếu đối tượng, lưu giữ các tên trong không gian tên dưới dạng phân cấp và cho phép đăng ký tên đối tượng trong thời gian chạy. Các ứng dụng Máy trạm có thể tìm ra đối tượng

cần sử dụng bằng cách xác định tham chiếu đối tượng trong không gian tên, qui trình của dịch vụ này được thể hiện trên hình 3.9. Hàm bind(name,objRef) dùng để tạo ràng buộc tên, hàm resolve(name) xác định đối tượng ràng buộc với tên trong ngữ cảnh đã cho. Dịch vụ đặt tên được mô tả trong mô đun CosNaming và được viết dưới dạng ngôn ngữ IDL như sau:

```
module CosNaming
{
    typedef string Istring;
    struct NameComponent {
        Istring id;
        Istring kind;
    };
    typedef sequence <NameComponent> Name;
    enum BindingType {nobject, ncontext};
    struct Binding {
        Name binding_name;
        BindingType binding_type;
    };
    typedef sequence <Binding> BindingList;
    interface BindingIterator;
    interface NamingContext {
        enum NotFoundReason { missing_node, not_context, not_object};
        exception NotFound {
            NotFoundReason why;
            Name rest_of_name;
        };
        exception CannotProceed {
            NamingContext ctxt;
            Name rest_of_name;
        };
        exception InvalidName {};
        exception AlreadyBound {};
        exception NotEmpty {};
        void bind(in Name n, in Object obj)
            raises(NotFound, CannotProceed, InvalidName, AlreadyBound);
        void rebind(in Name n, in Object obj)
            raises(NotFound, CannotProceed, InvalidName);
        void bind_context(in Name n, in NamingContext nc)
            raises(NotFound, CannotProceed, InvalidName, AlreadyBound);
        void rebind_context(in Name n, in NamingContext nc)
            raises(NotFound, CannotProceed, InvalidName);
        Object resolve (in Name n)
            raises(NotFound, CannotProceed, InvalidName);
        void unbind(in Name n)
            raises(NotFound, CannotProceed, InvalidName);
        NamingContext new_context();
    };
}
```

```
NamingContext bind_new_context(in Name n)
    raises(NotFound, AlreadyBound, CannotProceed, InvalidName);
void destroy()
    raises(NotEmpty);
void list (in unsigned long how_many,out BindingList bl, out
BindingIterator bi);
};

interface BindingIterator {
    boolean next_one(out Binding b);
    boolean next_n(in unsigned long how_many, out BindingList bl);
    void destroy();
};

};
```

Mô hình CosNaming bao gồm hai thành phần giao diện chính:

Giao diện NamingContext: Cung cấp các thao tác phục vụ cho việc nhúng các đối tượng và giải quyết các vấn đề về ngữ cảnh của tên.

Giao diện BindingIterator: Cung cấp các thao tác lặp lại việc nhúng các đối tượng.

Dịch vụ sự kiện

Các ứng dụng dựa trên kiến trúc CORBA phải có cơ chế đồng bộ giữa các đối tượng khi thực hiện, dịch vụ dịch vụ sự kiện (Event Service) cung cấp các công cụ để các đối tượng CORBA có thể gửi/nhận các sự kiện, dịch vụ này phải đảm bảo các yêu cầu sau:

Đảm bảo việc phân phát sự kiện chính xác, đối tượng CORBA chỉ đưa sự kiện vào và sẽ đảm bảo sự kiện đó đến đích theo yêu cầu.

Các sự kiện được phân phát theo kiểu hàng đợi.

Cho phép các tin báo nặc danh (bên phát/nhận không cần biết tên định danh của bên kia).

Cho phép kênh sự kiện, bên nhận được phép đăng ký chỉ nhận một số loại sự kiện.

Dịch vụ sự kiện thực hiện việc liên lạc giữa các đối tượng, để đảm nhiệm vai trò này nó định nghĩa hai vai trò cho đối tượng: vai trò phát sinh sự kiện (Event Supplier) và vai trò xử lý sự kiện (Event Consumer), trao đổi dữ liệu của sự kiện giữa bên phát sinh và bên xử lý được thực hiện bằng cách đưa ra các yêu cầu theo tiêu chuẩn CORBA. Để phục vụ việc trao đổi thông tin giữa bên phát sinh và bên xử lý sự kiện, kiến trúc CORBA đề xuất hai mô hình sau:

Mô hình đẩy (push): Cho phép bên phát sinh sự kiện chủ động kích hoạt việc truyền dữ liệu của sự kiện đến bên sử dụng, như vậy phía phát sinh sự kiện đóng vai trò chủ động.

Mô hình kéo (pull): Cho phép bên xử lý sự kiện được phép yêu cầu dữ liệu từ bên phát sinh sự kiện, bên xử lý sự kiện đóng vai trò chủ động.

Hai mô hình trên cùng thực hiện nhiệm vụ trao đổi thông tin giữa bên phát sinh và bên xử lý sự kiện, tuy nhiên trong mô hình thứ nhất bên phát sinh sự kiện chủ động kích hoạt việc chuyển dữ liệu cho bên xử lý, ngược lại trong mô hình thứ hai bên xử lý sự kiện chủ động đưa ra yêu cầu chuyển dữ liệu. Để thực hiện các yêu cầu của dịch vụ này, nhóm OMG đưa ra đặc tả gồm bốn mô đun: CosEventComm, CosEventChannelAdmin, CosTypedEventComm, CosTypedEventChannelAdmin.

Ví dụ, mô đun CosEventComm đảm nhiệm chức năng liên lạc giữa bên phát sinh và bên xử lý sự kiện, nó cung cấp 4 giao diện và được thể hiện trên ngôn ngữ IDL như sau:

```
module CosEventComm {  
    exception Disconnected{};  
    interface PushConsumer {  
        void push (in any data) raises(Disconnected);  
        void disconnect_push_consumer();  
    };  
    interface PushSupplier {  
        void disconnect_push_supplier();  
    };  
    interface PullSupplier {  
        any pull () raises(Disconnected);  
        any try_pull (out boolean has_event)  
            raises(Disconnected);  
    };  
};
```

```

        void disconnect_pull_supplier();
    };
    interface PullConsumer {
        void disconnect_pull_consumer();
    };
}

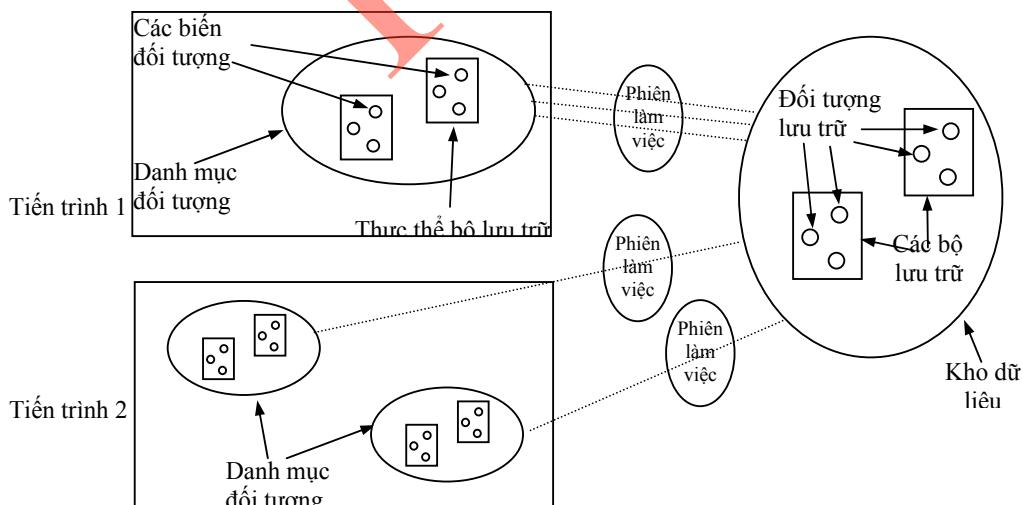
```

Giao diện PushConsumer dùng để nhận dữ liệu của sự kiện, như vậy bên cung cấp sự kiện sử dụng hàm push(...) để truyền dữ liệu cho bên nhận sự kiện. Giao diện PushSupplier cung cấp hàm disconnect_push_supplier() để giải phóng các tài nguyên mà bên cung cấp sự kiện đã dùng trong quá trình truyền sự kiện. Giao diện PullSupplier dùng để phát đi dữ liệu của sự kiện, bên sử dụng sự kiện gọi hàm pull() hoặc try_pull(...) để yêu cầu nhận dữ liệu của sự kiện. Giao diện PullConsumer cung cấp hàm disconnect_pull_consumer để kết thúc việc trao đổi thông tin và giải phóng các tài nguyên mà bên sử dụng sự kiện đã dùng để hỗ trợ cho việc truyền sự kiện.

Giữa đối tượng phát sinh sự kiện và đối tượng xử lý sự kiện thường được cài đặt lớp trung gian gọi là Event Channel (kênh sự kiện), đây là dịch vụ đóng cả hai vai trò: phát sinh và xử lý sự kiện. Thực chất việc cài đặt kênh sự kiện nhằm mục đích hỗ trợ cho việc trao đổi thông tin theo mô hình nhiều đối tượng phát sinh sự kiện với nhiều xử lý sự kiện đồng thời tăng hiệu suất hoạt động của hệ thống.

Dịch vụ duy trì bền bỉ

Dịch vụ duy trì bền bỉ (Persistence Service) cung cấp giao diện chung để bảo đảm việc duy trì và quản lý trạng thái liên tục của các đối tượng. Khi quá thời hạn tồn tại, thường các đối tượng sẽ được lưu trữ trong cơ sở dữ liệu và khi cần thiết có thể phục hồi lại. Trạng thái của một đối tượng gồm hai phần: trạng thái động (dynamic) và trạng thái bền vững (persistent). Trạng thái thứ nhất là thể hiện khi đối tượng được lưu trữ trong bộ nhớ, điều đó không đảm bảo tính an toàn của dữ liệu khi xảy ra sự cố vì vậy đối tượng sử dụng trạng thái bền để có thể dễ dàng khôi phục trạng thái động.



Hình 3.10 Nguyên tắc dịch vụ duy trì bền bỉ

Hình 3.10 minh họa cơ chế hoạt động của dịch vụ trạng thái bền vững với nguyên tắc: Các đối tượng được lưu trữ trong từng bộ lưu trữ (storage home) trong kho dữ

liệu, mỗi tiến trình tạo ra các biến đối tượng (object instance) và ánh xạ tương ứng với các đối tượng trong kho dữ liệu. Để truy nhập vào đối tượng trong kho dữ liệu cần thiết phải tạo liên kết giữa tiến trình với kho dữ liệu và nhúng đối tượng logic với đối tượng lưu trong kho dữ liệu, như vậy việc cập nhật các biến đối tượng sẽ kéo theo việc cập nhật đối tượng tương ứng trong kho dữ liệu.

Dịch vụ về chu kỳ sống của đối tượng

Dịch vụ về chu kỳ sống (Life Cycle Service) qui định các qui tắc để tạo/xoá/sao chép/di chuyển các đối tượng CORBA. Các môi trường dựa trên kiến trúc CORBA hỗ trợ các đối tượng phân tán, do đó dịch vụ này xây dựng các qui ước để ứng dụng Máy trạm có thể thực hiện các thao tác trên đối tượng phân tán tại các vị trí khác nhau. Trong thực tế các thao tác này thường gặp những vấn đề sau:

Vấn đề khi tạo một đối tượng mới: Ứng dụng Máy trạm cần phải kiểm soát được vị trí để tạo đối tượng mới và vị trí đó phải được quyết định theo chính sách quản trị. Khi tạo một đối tượng mới ứng dụng Máy trạm phải tìm ra thực thể tạo lập đối tượng và liên kết với thực thể đó để tạo đối tượng mới, mức độ ảnh hưởng của Máy trạm đối với các giá trị ban đầu của đối tượng mới tạo lập

Vấn đề di chuyển hoặc sao chép đối tượng: Ứng dụng Máy trạm có thể kiểm soát được vị trí của đối tượng nguồn và đích tuân thủ theo chính sách quản trị. Trong quá trình này ứng dụng Máy trạm phải liên kết với thực thể phục vụ cho việc sao chép hoặc di chuyển đối tượng đồng thời xác định những ảnh hưởng đối với mã lệnh thực hiện của các đối tượng được sao chép hoặc di chuyển.

Ảnh hưởng đến các đối tượng khác: Các đối tượng phân tán thường có mối liên kết với nhau (gọi là quan hệ giữa các đối tượng), việc sao chép/di chuyển hoặc xoá các đối tượng có thể sẽ ảnh hưởng tới các đối tượng khác, vì vậy cần phải xác định ranh giới quan hệ của đối tượng khi thực hiện các thao tác trên.

Để giải quyết các vấn đề trên, dịch vụ về chu kỳ sống xây dựng mô hình xử lý của Máy trạm về chu kỳ tồn tại cho đối tượng phục vụ cho các thao tác tương ứng khi tạo lập, xoá, sao chép và di chuyển các đối tượng. Đặc tả của dịch vụ này viết trên ngôn ngữ IDL như sau:

```
#include <CosNaming.idl>
#pragma prefix "omg.org"
module CosLifeCycle{
    typedef CosNaming::Name Key;
    typedef Object Factory;
        typedef sequence <Factory> Factories;
    typedef struct NVP {
        CosNaming::Istring name;
        any value;
    } NameValuePair;
    typedef sequence <NameValuePair> Criteria;
    exception NoFactory {
        Key search_key;
    };
    exception NotCopyable { string reason; };
    exception NotMovable { string reason; };
```

```
exception NotRemovable { string reason; };
exception InvalidCriteria{Criteria invalid_criteria; };
exception CannotMeetCriteria {Criteria unmet_criteria; };
interface FactoryFinder {
Factories find_factories(in Key factory_key)
raises(NoFactory);
};
interface LifeCycleObject {
LifeCycleObject copy(in FactoryFinder there,
in Criteria the_criteria)
raises(NoFactory, NotCopyable, InvalidCriteria,
CannotMeetCriteria);
void move(in FactoryFinder there,
in Criteria the_criteria)
raises(NoFactory, NotMovable, InvalidCriteria,
CannotMeetCriteria);
void remove()
raises(NotRemovable);
};
interface GenericFactory {
#ifndef NO_ESCAPED_IDENTIFIERS
boolean _supports(in Key k);
#else
boolean _supports(in Key k);
#endif
Object create_object (
in Key k,
in Criteria the_criteria)
raises (NoFactory, InvalidCriteria,
CannotMeetCriteria);
};
};
```

Tạo đối tượng:

Tồn tại một đối tượng tên là Factory có nhiệm vụ tạo ra các đối tượng mới, ứng dụng Máy trạm liên tạo đối tượng mới bằng cách tham chiếu đến đối tượng Factory và gửi yêu cầu tạo lập đối tượng.

Xoá đối tượng, sao chép hoặc di chuyển đối tượng:

Tồn tại giao diện LifeCycleObject cung cấp các thao tác liên quan tới sự tồn tại của các đối tượng. Để thực hiện các thao tác này, trước hết Máy trạm phải nhận tham chiếu đến đối tượng LifeCycleObject sau đó gọi các thao tác tương ứng.

Dịch vụ điều khiển tranh

Dịch vụ điều khiển tranh (Concurrency Service) cung cấp giao diện để quản lý tranh trong các đối tượng CORBA nhằm mục đích đảm bảo cho tất cả các ứng dụng Máy trạm có thể đồng thời truy nhập các tài nguyên chung nhưng vẫn đảm bảo tính nhất quán của tài nguyên chung đó. Dịch vụ này điều khiển sử dụng tài nguyên chung bằng khoá đọc/ghi, mỗi khoá được gán với một tài nguyên và một ứng

dụng Máy trạm và ứng dụng Máy trạm phải nhận được khoá trước khi truy nhập vào tài nguyên chung. Dịch vụ điều khiển tương tranh được định nghĩa trong mô đun CosConcurrencyControl như sau:

```
#include <CosTransactions.idl>
module CosConcurrencyControl {
    enum lock_mode {read, write, upgrade, intention_read,
intention_write};
    exception LockNotHeld{};
    interface LockCoordinator
    {
        void drop_locks();
    };
    interface LockSet
    {
        void lock(in lock_mode mode);
        boolean try_lock(in lock_mode mode);
        void unlock(in lock_mode mode)
        raises(LockNotHeld);
        void change_mode(in lock_mode held_mode,
in lock_mode new_mode)
        raises(LockNotHeld);
        LockCoordinator get_coordinator(
in CosTransactions::Coordinator which);
    };
    interface TransactionalLockSet
    {
        void lock(in CosTransactions::Coordinator current,
in lock_mode mode);
        boolean try_lock(in CosTransactions::Coordinator current,
in lock_mode mode);
        void unlock(in CosTransactions::Coordinator current,
in lock_mode mode)
        raises(LockNotHeld);
        void change_mode(in CosTransactions::Coordinator current,
in lock_mode held_mode,
in lock_mode new_mode)
        raises(LockNotHeld);
        LockCoordinator get_coordinator(
in CosTransactions::Coordinator which);
    };
    interface LockSetFactory
    {
        LockSet create();
        LockSet create_related(in LockSet which);
        TransactionalLockSet create_transactional();
        TransactionalLockSet create_transactional_related(in
TransactionalLockSet which);
    };
}
```

```
};  
};
```

Dịch vụ thể hiện đối tượng

Dịch vụ thể hiện đối tượng (Externalization Service) định nghĩa các giao thức và các qui ước để xuất/tiếp nhận các đối tượng, các đối tượng có thể được thể hiện dưới dạng các dòng bit trong bộ nhớ, trên đĩa hoặc luân chuyển trên mạng. Dạng thể hiện của đối tượng có thể tồn tại trong khoảng thời gian không hạn chế và được chuyển bằng bất kỳ phương tiện nào không thuộc môi trường ORB. Trong thực tế, Máy trạm thường yêu cầu dữ liệu được lưu trong tập tin sử dụng dạng đã được chuẩn hoá. Kiến trúc CORBA chưa có cơ chế truyền giá trị, vì vậy có thể ứng dụng dịch vụ này để thực hiện chức năng trên.

Dịch vụ quan hệ của đối tượng

Các đối tượng phân tán thường được sử dụng để mô hình hóa thế giới thực, do đó chúng thường không tồn tại tách biệt mà ngược lại giữa các đối tượng phân tán đó luôn tồn tại các quan hệ. Dịch vụ quan hệ của các đối tượng (Relationship Service) cho phép thể hiện quan hệ giữa các đối tượng nhằm mục đích giám tính phức tạp trong việc quản lý các mối quan hệ.

Dịch vụ quản lý giao dịch

Dịch vụ quản lý giao dịch (Transaction Service) cung cấp các giao diện hỗ trợ khả năng trao đổi thông tin giữa các đối tượng CORBA, việc quản lý các giao dịch đóng vai trò quan trọng nhằm đảm bảo tính an toàn của hệ thống, điều này càng trở nên quan trọng hơn trong môi trường tương tranh.

Dịch vụ truy vấn

Dịch vụ truy vấn (Query Service) cho phép ứng dụng thực hiện các thao tác truy vấn trên các đối tượng CORBA (tương tự như truy vấn các bảng trong CSDL), tuy nhiên khái niệm truy vấn trong được mở rộng theo nghĩa có thể chọn/thêm/cập nhật/xoá tập các đối tượng.

Dịch vụ cấp phép

Dịch vụ cấp phép (Licensing Service) cho phép định nghĩa các chính sách để điều khiển việc sử dụng các dịch vụ, có ba loại chính sách sau:

Cấp phép theo thời gian: ngày bắt đầu, ngày hết hạn, khoảng thời gian hết hạn sử dụng dịch vụ.

Cấp phép dựa trên số liệu thực tế, ví dụ số liệu thực về việc sử dụng tài nguyên, số người đồng thời sử dụng dịch vụ

Cấp phép riêng cho từng người sử dụng

Dịch vụ về quyền sở hữu

Dịch vụ quyền sở hữu (Properties Service) cho phép các đối tượng xác định tập các tài sản thuộc quyền sở hữu theo từng cặp tên/giá trị, khái niệm tài sản được hiểu là các giá trị đã xác định gắn với các đối tượng theo phương pháp động.

Dịch vụ thời gian

Dịch vụ thời gian (Time Service) cho phép người sử dụng có thể tạo ra các sự kiện dựa trên việc đặt thời gian. Hiện nay mỗi hệ thống sử dụng cách thể hiện thời gian khác nhau, kiến trúc CORBA lựa chọn cách thể hiện thời gian quốc tế UTC.

Dịch vụ bảo mật

Dịch vụ bảo mật (Security Service) qui định giao diện cho các thuộc tính bảo mật:

Tên định danh và ủy quyền: Kiểm tra người sử dụng

Ủy quyền và kiểm soát truy nhập: Xác định người sử dụng nào được phép truy nhập tới các đối tượng dịch vụ nào

Kiểm tra bảo mật: Ghi lại các thao tác của người sử dụng

Bảo mật truyền tin: bao gồm việc ủy quyền người sử dụng đến các dịch vụ, bảo vệ độ tin cậy và độ tin cậy.

Không thừa nhận: cung cấp các tính năng bảo mật tương tự như chữ ký điện tử

Quản lý các chính sách bảo mật

Dịch vụ giới thiệu đối tượng

Giống như dịch vụ đặt tên, dịch vụ giới thiệu đối tượng (Object Trader Service) cho phép một đối tượng khác có thể xác định các đối tượng CORBA. Thay vì sử dụng tên, đối tượng Máy trạm tìm các dịch vụ dựa trên tên của thao tác, các tham số và các kiểu giá trị trả về. Như vậy sự khác nhau cơ bản giữa dịch vụ này và dịch vụ đặt tên là ở chỗ: dịch vụ đặt tên tìm kiếm dịch vụ của đối tượng khi biết chính xác tên của đối tượng trong khi đó dịch vụ giới thiệu đối tượng xác định dịch vụ dựa trên vị trí/chức năng/tên của dịch vụ.

Dịch vụ nhóm

Dịch vụ tập hợp (Collections Service) hỗ trợ việc nhóm các đối tượng hoặc hỗ trợ các thao tác thực hiện trên nhóm các đối tượng. Ví dụ: tập hợp, ngăn xếp, hàng

đợi.... đều là những tập hợp, nhiệm vụ của dịch vụ tập hợp là cung cấp một phương pháp thống nhất để tạo lập và thao tác với tập các đối tượng chung nhất.

3.3.2 Kiến trúc Corba và các yêu cầu phần mềm trung gian

Kiến trúc CORBA tuân thủ các yêu cầu của phần mềm trung gian đã nêu trong chương 1, điều này được thể hiện trong các khía cạnh về định nghĩa giao diện, xác định đối tượng và cơ chế gọi các thao tác trên đối tượng phục vụ.

Ngôn ngữ định nghĩa giao diện

Kiến trúc CORBA sử dụng ngôn ngữ định nghĩa giao diện IDL để mô tả giao diện của các đối tượng. IDL là ngôn ngữ mô tả phục vụ cho mô hình lập trình hướng đối tượng, do đó hoàn toàn có khả năng sử dụng các tính chất kế thừa và đa hình trong ngôn ngữ lập trình hướng đối tượng. Mục tiêu của ngôn ngữ IDL là mô tả giao diện của các đối tượng, do đó độc lập với ngôn ngữ lập trình và có thể sử dụng các chương trình dịch để biên dịch sang các ngôn ngữ lập trình theo yêu cầu.

Ngôn ngữ IDL định nghĩa các loại đối tượng bằng cách xác định các giao diện của chúng, mỗi giao diện bao gồm các hàm và các tham số của hàm. Mã lệnh thực hiện của các hàm nằm ngoài phạm vi định nghĩa của ngôn ngữ này, như vậy nó chỉ thể hiện các thao tác phục vụ cho Máy trạm.

Xác định địa chỉ đối tượng phục vụ

Để truy nhập các đối tượng từ xa, ứng dụng Máy trạm cần có địa chỉ của đối tượng phục vụ, thông tin này chứa trong bảng tham chiếu đối tượng. Vấn đề tham chiếu đối tượng đã được chuẩn hóa trong đặc tả CORBA 2.0 nhằm mục đích cho phép các ứng dụng phân tán có thể chạy trên môi trường ORB của các hãng khác nhau.

Gọi đối tượng phục vụ

Khi đã xác định được giao diện và địa chỉ của đối tượng phục vụ, có thể gọi các thao tác từ xa, quá trình này tuân thủ các bước thực hiện của phần mềm trung gian. Bắt đầu từ việc biên dịch tập tin mô tả giao diện viết trên ngôn ngữ IDL sẽ nhận được Máy trạm stub và Máy chủ skeleton. Yêu cầu của ứng dụng Máy trạm được chuyển đến đối tượng phục vụ thông qua môi trường ORB, quá trình này hoàn toàn trong suốt đối với người phát triển.

3.3.3 Áp dụng CORBA trong xây dựng ứng dụng phân tán

Để xây dựng phần mềm của hệ thống phân tán theo kiến trúc CORBA cần phải xác định các đối tượng Máy chủ và các đối tượng Máy trạm của hệ thống.

Xây dựng ứng dụng Máy chủ:

Định nghĩa các giao diện Máy chủ:

Viết mô đun định nghĩa các giao diện của Máy chủ (tức là các dịch vụ Máy chủ cung cấp và hướng pháp truy nhập các dịch vụ đó) dưới dạng tập tin .idl.

Biên dịch tập tin .idl bằng trình biên dịch IDL ra ngôn ngữ phù hợp, kết quả sẽ nhận được Máy trạm Stub và Máy chủ Skeleton

Viết mã nguồn thể hiện các giao diện Máy chủ cung cấp dựa trên Máy chủ Skeleton

Viết chương trình chính cho Máy chủ với cấu trúc cơ bản của hàm main() như sau:

Khởi tạo ORB và BOA.

Tạo các đối tượng Máy chủ.

Thông báo cho BOA biết đối tượng Máy chủ đã sẵn sàng

Chờ sự kiện CORBA, thực hiện và kết thúc

Biên dịch ứng dụng Máy chủ

Xây dựng ứng dụng Máy trạm:

Cài đặt Máy trạm:

Sử dụng Máy trạm Stub đã được tạo trong bước 2 trên đây để tạo tập tin cài đặt các tính năng của Máy trạm.

Viết chương trình chính cho Máy trạm với khung hàm main() như sau:

Khởi tạo ORB và BOA/POA.

Tạo các đối tượng Máy trạm.

Thông báo cho BOA /POA biết đối tượng máy khách đã sẵn sàng

Thực hiện các tác vụ (xác định các đối tượng máy chủ...)

3.4 Gọi phương thức từ xa

Gọi phương thức từ xa (Java RMI) thực chất là việc áp dụng phương pháp gọi thủ tục từ xa trong ngôn ngữ lập trình Java. Java RMI cho phép một đối tượng chạy trên một máy ảo Java này có thể kích hoạt một phương thức của một đối tượng đang chạy trên một máy ảo Java khác. Đối tượng có phương thức được gọi từ xa gọi là các đối tượng ở xa (Remote Object). Một ứng dụng RMI thường bao gồm 2 phần phân biệt: Một chương trình máy chủ và một chương trình máy khách.

Chương trình máy chủ tạo một số các đối tượng ở xa, tạo các *tham chiếu* (reference) đến chúng và chờ những chương trình máy khách kích hoạt các phương thức của các Đối tượng ở xa này. Chương trình máy khách lấy một *tham chiếu* đến một hoặc nhiều Đối tượng ở xa trên Máy chủ và kích hoạt các phương thức từ xa thông qua các tham chiếu. Một chương trình máy khách có thể kích hoạt các phương thức ở xa trên một hay nhiều Máy chủ. Tức là sự thực thi của chương trình được trải rộng trên nhiều máy tính. Đây chính là đặc điểm của các ứng dụng phân tán. Nói cách khác, RMI là cơ chế để xây dựng các ứng dụng phân tán dưới ngôn ngữ Java.

Kiến trúc của chương trình khách/chủ:

Kiến trúc một chương trình khách/chủ theo mô hình RMI bao gồm: Máy chủ là chương trình cung cấp các đối tượng có thể được gọi từ xa. Máy khách là chương trình có tham chiếu đến các phương thức của các đối tượng ở xa trên máy chủ. Stub chứa các tham chiếu đến các phương thức ở xa trên máy chủ. Skeleton đón nhận các tham chiếu từ Stub để kích hoạt phương thức tương ứng trên Máy chủ. Remote Reference Layer là hệ thống truyền thông của RMI.

Các cơ chế liên quan trong một ứng dụng đối tượng phân tán:

Cơ chế định vị đối tượng từ xa: Cơ chế này xác định cách thức mà chương trình Máy khách có thể lấy được *tham chiếu* (Stub) đến các đối tượng ở xa. Thông thường người ta sử dụng một Dịch vụ danh bạ (Naming Service) lưu giữ các tham khảo đến các đối tượng cho phép gọi từ xa mà Máy khách sau đó có thể tìm kiếm.

Cơ chế giao tiếp với các đối tượng ở xa (Communicate with Đối tượng ở xa): Chi tiết của cơ chế giao tiếp với các đối tượng ở xa được cài đặt bởi hệ thống RMI.

Tải các lớp dạng bytecodes cho các lớp mà nó được chuyển tải qua lại giữa Máy ảo (Load class bytecodes for objects that are passed around): Vì RMI cho phép các chương trình gọi phương thức từ xa trao đổi các đối tượng với các phương thức ở xa dưới dạng các tham số hay giá trị trả về của phương thức, nên RMI cần có cơ chế cần thiết để tải mã Bytecodes của các đối tượng từ máy ảo này sang máy ảo khác.

Tiến trình vận hành của một ứng dụng khách/chủ theo kiểu RMI gồm các bước sau:

Máy chủ tạo ra các đối tượng cho phép gọi từ xa cùng với các Stub và Skeleton của chúng.

Máy chủ sử dụng lớp đặt tên để đăng ký tên cho một đối tượng từ xa.

Lớp đặt tên đăng ký Stub của đối tượng từ xa với Registry Máy chủ.

Thành phần đăng ký (Registry) Máy chủ sẵn sàng cung cấp tham khảo đến đối tượng từ xa khi có yêu cầu.

Máy khách yêu cầu định vị đối tượng xa qua tên đã được đăng ký với dịch vụ tên.

Bộ phận quản lý tên tải Stub của đối tượng xa từ dịch vụ tên mà đối tượng xa đã đăng ký về Máy khách.

Cài đặt đối tượng Stub và trả về tham khảo đối tượng xa cho Máy khách.

Máy khách thực thi một lời gọi phương thức xa thông qua đối tượng Stub.

Các lớp hỗ trợ chương trình theo kiểu khách/chủ trong Java

Java hỗ trợ các lớp cần thiết để cài đặt các ứng dụng Khách/chủ theo kiểu RMI trong các gói: java.rmi. Trong số đó các lớp thường được dùng sau:

```
java.rmi.Naming  
java.rmi.RMISecurityManager  
java.rmi.RemoteException;  
java.rmi.máy chủ.RemoteObject  
java.rmi.máy chủ.RemoteMáy chủ  
java.rmi.máy chủ.UnicastRemoteObject
```

Xây dựng một ứng dụng phân tán với RMI

Xây dựng một ứng dụng phân tán bằng cơ chế RMI gồm các bước sau:

Thiết kế và cài đặt các thành phần của ứng dụng.

Biên dịch các chương trình nguồn và tạo ra Stub và Skeleton.

Tạo các lớp có thể truy xuất từ mạng.

Thực thi ứng dụng

Thiết kế và cài đặt các thành phần của ứng dụng:

Định nghĩa các giao diện cho các phương thức ở xa (Remote Interface): Một giao diện ở xa mô tả các phương thức mà nó có thể được kích hoạt từ xa bởi các Máy khách. Đi cùng với việc định nghĩa Giao diện ở xa là việc xác định các lớp cục bộ làm tham số hay giá trị trả về của các phương thức được gọi từ xa.

Cài đặt các đối tượng từ xa (Đối tượng ở xa): Các Đối tượng ở xa phải cài đặt cho một hoặc nhiều Giao diện ở xas đã được định nghĩa. Các lớp của Đối tượng ở xa class cài đặt cho các phương thức được gọi từ xa đã được khai báo trong Giao diện ở

xa và có thể định nghĩa và cài đặt cho cả các phương thức được sử dụng cục bộ. Nếu có các lớp làm đối số hay giá trị trả về cho các phương thức được gọi từ xa thì ta cũng định nghĩa và cài đặt chúng.

Cài đặt các chương trình Máy khách: Các chương trình Máy khách có sử dụng các Đối tượng ở xa có thể được cài đặt ở bất kỳ thời điểm nào sau khi các Giao diện ở xa đã được định nghĩa.

Biên dịch các tập tin nguồn và tạo Stubs và Skeleton:

Giai đoạn này gồm hai bước:

Dùng chương trình biên dịch javac để biên dịch các tập tin nguồn như các giao diện ở xa, các lớp cài đặt cho các giao diện ở xa, lớp máy chủ, lớp máy khách và các lớp liên quan khác.

Dùng trình biên dịch rmic để tạo ra stub và skeleton cho các đối tượng từ xa từ các lớp cài đặt cho các giao diện ở xa.

Tạo các lớp có thể truy xuất từ mạng:

Tạo một tập tin chứa tất cả các tập tin có liên quan như các giao diện ở xa stub, các lớp hỗ trợ mà chúng cần thiết phải tải về Máy khách và làm cho tập tin này có thể truy cập đến thông qua một Web máy chủ.

Thực thi ứng dụng:

Thực thi ứng dụng bao gồm việc thực thi đăng ký RMI trên máy chủ, thực thi máy chủ, và thực thi máy khách. Các công việc bao gồm:

Tạo giao diện (interface) khai báo các phương thức được gọi từ xa của đối tượng.

Tạo lớp cài đặt (implement) cho giao diện đã được khai báo.

Viết chương trình Máy chủ.

Viết chương trình Máy khách.

Dịch các tập tin nguồn theo dạng RMI để tạo ra các lớp tương ứng và stub cho máy khách, skeleton cho máy chủ.

Khởi động dịch vụ registry.

Thực hiện chương trình Máy chủ.

Thực thi chương trình Máy khách.

3.5 Dịch vụ web

Dịch vụ web (Web Service) là một mô hình phát triển các ứng dụng phân tán dựa trên các dịch vụ web, nó bao gồm tập các chuẩn cho phép các phát triển viên xây dựng các ứng dụng phân tán mà không phụ thuộc nền tảng hệ thống (phần cứng cũng như hệ điều hành). Dịch vụ web là các thành phần ứng dụng, nó tương tác sử dụng các giao thức mở và bên trong tự chứa các đặc tả dữ liệu theo chuẩn XML.

Tốc độ trao đổi thông tin dựa trên WebService thường không cao, tuy nhiên việc phát triển các ứng dụng này tương đối đơn giản, rút ngắn thời gian phát triển phần mềm. WebService xây dựng dựa trên ngôn ngữ đặc tả XML, cho phép ứng dụng có thể liên lạc với nhau mà không phụ thuộc nền tảng hay ngôn ngữ lập trình. XML là ngôn ngữ có thể mô tả bất cứ dữ liệu nào mà không phụ thuộc môi trường nền tảng thực trong việc trao đổi giữa các hệ thống.

WebService liên lạc với máy khách thông qua các thông điệp XML được chuyển qua mạng nhờ giao thức HTTP. Do đó, các ứng dụng khác nhau trên Web có thể truy cập WebService tự động, giải quyết vấn đề giao tiếp giữa các hệ thống khác nhau. Chúng có thể được viết bằng nhiều loại ngôn ngữ lập trình, như Java, C++, hay Perl. Ngay khi một ứng dụng cung cấp tính năng mới được đóng gói như một WS, thì các hệ thống sử dụng WS lập tức có thể sử dụng chức năng của ứng dụng này.

Để tạo ra những ứng dụng lớn, được xây dựng bởi nhiều tổ chức, thì những ứng dụng này sẽ được chia thành những thành phần nhỏ, hay dịch vụ chia sẻ, đặt tại các máy tính khác nhau, cũng có thể được cài đặt bởi các công nghệ khác nhau. Sau đó, những module, dịch vụ chia sẻ này được đóng gói và giao dịch qua mạng sử dụng các giao thức web chuẩn. Ưu điểm của các giao thức chuẩn là khiến ứng dụng chúng có thể được truy cập một cách dễ dàng qua môi trường internet, khắc phục được tính đóng cửa quan hệ của các hệ thống nguyên khôi. Nhờ khả năng độc lập về nền tảng hệ thống, ngôn ngữ lập trình, mà các ứng dụng WebService có thể được tái sử dụng.

3.5.1 Các thành phần trong kiến trúc dịch vụ Web

Nền tảng của WS platform là XML và HTTP. XML là ngôn ngữ có thể được sử dụng giữa các nền tảng và ngôn ngữ lập trình khác nhau, mà vẫn mô tả được những chức năng, thông điệp phức tạp. Còn HTTP là một giao thức rất phổ biến, được sử dụng nhiều nhất trên mạng Internet. Các thành phần dịch vụ của WebService bao gồm:

- SOAP (Simple Object Access Protocol): là một giao thức dựa trên ngôn ngữ XML, được dùng để cho các ứng dụng trao đổi thông tin qua giao thức HTTP.
- UDDI (Universal Description, Discovery and Integration): là một dịch vụ thư mục, nơi các ứng dụng có thể đăng ký hay tìm kiếm dịch vụ.
- WSDL (Web Services Description Language): là một ngôn ngữ dựa trên ngôn ngữ XML, dùng để định vị và mô tả WS.

3.5.2 Cách thức trao đổi thông tin của dịch vụ Web

Kiến trúc WebService gồm ba thành phần chính: Thành phần cung cấp dịch vụ (provider), thành phần sử dụng dịch vụ (consumer) và môi trường trao đổi thông tin (broker). Thành phần cung cấp dịch vụ WebService sẽ thông báo sự có mặt của dịch vụ (publish) WS tới broker. Sau đó, phía yêu cầu dịch vụ tìm kiếm và yêu cầu thông tin dịch vụ tại broker. Mỗi khi thông tin yêu cầu này được tìm thấy, phía người sử dụng sẽ ràng buộc (bind) ứng dụng này với WebService. Quy trình đơn giản tạo nên một WebService có thể được chia thành các giai đoạn sau:

- Bên cung cấp dịch vụ tạo ra một WS, sử dụng WSDL để mô tả.
- Bên cung cấp dịch vụ đăng ký dịch vụ (dưới dạng UDDI registry và/hoặc ebXML registry)
- Một dịch vụ hay người dùng khác định vị và yêu cầu dịch vụ đã được đăng ký bằng cách truy vấn UDDI và/hoặc ebXML. Dịch vụ hay người dùng này sử dụng SOAP nếu truy vấn sử dụng UDDI và/hoặc bản đăng ký exXML viết một ứng dụng để ràng buộc dịch vụ đã đăng ký vừa yêu cầu.
- Dữ liệu và thông điệp được giao dịch dưới dạng XML trên nền giao thức HTTP.

Giao thức truy nhập đối tượng đơn giản SOAP (Simple Object Access Protocol): là một giao thức truyền thông giữa các ứng dụng qua môi trường internet. SOAP dùng

để định dạng thông điệp trao đổi, dựa trên ngôn ngữ XML trên nền giao thức HTTP. XML định dạng thông tin dưới dạng text, do đó thông điệp SOAP có thể dễ dàng đi qua tường lửa. SOAP đơn giản, có khả năng mở rộng, độc lập với ngôn ngữ lập trình cũng như độc lập với platform. SOAP mô tả rõ định dạng của thông điệp REQUEST và RESPONSE, dùng trong việc gửi và nhận thông tin thông qua cổng giao thức HTTP nhờ phương thức POST. Về cú pháp, khái SOAP bao gồm những thành phần:

- Envelope: Phần bắt buộc phải có, định dạng tài liệu XML của thông điệp SOAP.
- Header: Có thể bỏ qua, chứa các thông tin header.
- Body: Phần bắt buộc phải có, chứa thông tin gọi & hồi đáp.
- Fault: chứa thông tin trạng thái & lỗi.

SOAP hỗ trợ hai kiểu truyền thông khác nhau :

- Remote procedure call (RPC): cho phép gọi hàm hoặc thủ tục qua mạng. Kiểu này được khai thác bởi nhiều web service và có nhiều trợ giúp.
- Document: được biết như kiểu hướng message : kiểu này cung cấp một lớp thấp của sự trừu tượng hóa, và yêu cầu người lập trình nhiều hơn khi làm việc.

Các định dạng thông điệp, tham số, và lời gọi đến các API thì tương ứng trong RPC và document là khác nhau. Nên việc quyết định chọn cái nào tùy thuộc vào thời gian xây dựng và sự phù hợp của service cần xây dựng.

Ngôn ngữ mô tả dịch vụ web WSDL (Web Services Description Language): là một ngôn ngữ mô tả giao tiếp và thực thi dựa trên XML. WSDL dùng để mô tả và định vị dịch vụ web, cho biết WS này cung cấp chức năng gì, được đặt tại đâu, cách để truy nhập và gọi phương thức được cung cấp bởi nó. Cấu trúc WSDL gồm 4 thành phần:

- Type: Định nghĩa những kiểu dữ liệu được sử dụng bởi WebService.
- Message: Định nghĩa thành phần dữ liệu của một phương thức
- PortType: là thành phần quan trọng nhất của WS, mô tả một WebService, các phương thức có thể được thực hiện, và các thông điệp liên quan.
- Binding: định nghĩa định dạng thông điệp và chi tiết giao thức cho mỗi cổng.

Mô tả tổng thể, Phát hiện và tích hợp UDDI (Universal Description, Discovery and Integration): sử dụng ngôn ngữ XML và các giao thức HTTP, DNS và truyền thông qua SOAP. UDDI đóng vai trò môi trường trung gian trong quá trình trao đổi thông tin của các dịch vụ web, nó là một thư mục đơn nhất dùng để lưu thông tin của mọi dịch vụ web. Giao diện của thư mục này được mô tả bằng WSDL. Thư mục UDDI bao gồm 3 thành phần thông tin:

White page: Mô tả thông tin nghiệp vụ. Có vai trò xuất bản (publish), nghĩa là chứa thông tin đăng ký dịch vụ của các nhà cung cấp dịch vụ, bao gồm các thông tin nghiệp vụ, như: Tên, mô tả nghiệp vụ, thông tin liên hệ ...

Yellow page: Bao gồm các thông tin dịch vụ, mô tả một nhóm các WS bao gồm đối tượng dịch vụ nghiệp vụ. Yellow Page có vai trò trong việc tìm kiếm (Find), nghĩa là nó chứa các thông tin cần thiết để một ứng dụng có thể tìm tới một WS cụ thể. Một nghiệp vụ có thể cung cấp nhiều dịch vụ, do đó với mỗi White page có thể có tới vài Yellow Page.

Green page: Chứa các thông tin ràng buộc (bind), mô tả cách một ứng dụng kết nối và tương tác với WS sau khi tìm được WS mong muốn. Thông tin bao gồm: URLs, Tên phương thức, kiểu tham số .. của WS.

Danh sách các dịch vụ web được mô tả bằng WSDL sẽ được gửi tới UDDI để đăng ký, tại đây sẽ ánh xạ danh sách dịch vụ thành bản UDDI định dạng bởi XML. Thành phần bản đăng ký UDDI bao gồm 3 phần: White page, Yellow Page và Green Page.

3.5.3 Quy trình xây dựng ứng dụng dịch vụ Web

Để xây dựng một ứng dụng phân tán dựa trên dịch vụ Web cần thực hiện các bước sau :

Định nghĩa và xây dựng các chức năng , các dịch vụ mà servive sẽ cung cấp.

Tạo WSDL cho dịch vụ, chuyển đổi thông tin giữa máy chủ và WSDL.

Xây dựng SOAP trên máy chủ.

Đăng ký WSDL với UDDI registry để cho phép các máy trạm có thể tìm thấy và truy nhập.

Client nhận file WSDL và từ đó xây dựng SOAP client để có thể kết nối với SOAP server.

Xây dựng ứng dụng phía khách, sau đó gọi thực hiện dịch vụ thông qua việc kết nối tới máy chủ SOAP.

3.6 Kiến trúc hướng dịch vụ

Sự phát triển nhanh chóng của công nghệ làm nảy sinh môi trường giao tiếp không đồng nhất giữa các thành phần trong hệ thống phân tán. Một vấn đề đặt ra đối với các tổ chức công nghệ thông tin là làm sao xây dựng được một kiến trúc phân mềm có khả năng tích hợp và sử dụng các thành phần mới nhằm giảm thiểu chi phí phát triển và bảo trì hệ thống phần mềm. Chúng ta đã có các kiến trúc hướng đối tượng phân tán, tuy nhiên các kiến trúc này đều có đặc tính ràng buộc chặt chẽ giữa các thành phần với nhau làm cho các kiến trúc này chưa thật hiệu quả. Kiến trúc hướng dịch vụ ra đời gần nhằm mục đích quyết những vấn đề khó khăn trong giao tiếp giữa các thành phần của hệ thống phân tán.

3.6.1 Giới thiệu về kiến trúc hướng dịch vụ

Kiến trúc hướng dịch vụ (SOA—Service Oriented Architectural) là một cách tiếp cận tổ chức hệ thống thông tin sao cho có thể truy nhập các tài nguyên thông qua các giao diện và thông điệp. Kiến trúc SOA dùng trong các chuẩn mở để biểu diễn các dịch vụ thông qua các giao diện đã được chuẩn hóa, từng thành phần riêng lẻ sẽ trở thành những khối cơ bản để có thể tái sử dụng trong các ứng dụng khác, vì vậy kiến trúc này được sử dụng để tích hợp các ứng dụng.

Dịch vụ là các hàm chức năng thực hiện theo quy trình nghiệp vụ nào đó, chúng có thể dễ dàng tìm thấy và liên thông với nhau, tuy nhiên mức độ gắn kết không cao. Mỗi dịch vụ bao gồm nhiều thành phần và được đóng gói ở mức cáo, người sử dụng không cần biết vị trí của chúng. Kiến trúc SOA bao gồm các dịch vụ kết nối mềm dẻo với nhau, mỗi dịch vụ có giao diện được định nghĩa rõ ràng và độc lập với nền tảng của hệ thống và có thể tái sử dụng. SOA là cấp độ cao hơn của sự phát triển ứng dụng,

chú trọng đến quy trình nghiệp vụ và dùng giao diện chuẩn để che giấu sự phức tạp bên trong.

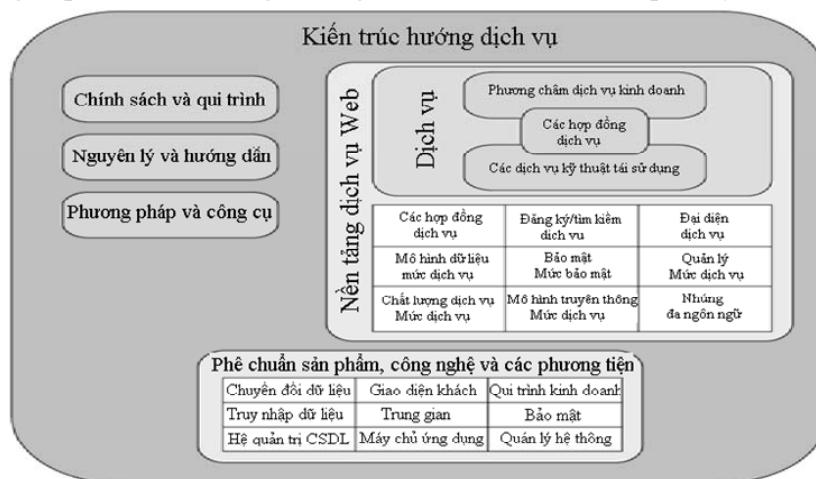
Thiết kế SOA tách riêng phần thực hiện dịch vụ với giao diện gọi dịch vụ, điều này tạo nên một giao diện nhất quán cho ứng dụng sử dụng dịch vụ mà không cần quan tâm tới công nghệ thực hiện dịch vụ. Thay vì xây dựng các ứng dụng đơn lẻ và đồ sộ, nhà phát triển sẽ xây dựng các dịch vụ tinh gọn hơn có thể triển khai và tái tạo sử dụng trong toàn bộ quy trình nghiệp vụ. Điều này cho phép tái sử dụng phần mềm tốt hơn, cũng như tăng sự mềm dẻo vì các nhà phát triển có thể cải tiến dịch vụ mà không làm ảnh hưởng đến ứng dụng sử dụng dịch vụ.

Kiến trúc SOA không hoàn toàn mới, DCOM và CORBA cũng có kiến trúc tương tự. Tuy nhiên, các kiến trúc cũ ràng buộc các thành phần với nhau quá chặt ví dụ như các ứng dụng phân tán muốn làm việc với nhau phải được thoả thuận về chi tiết tập hàm API, một thay đổi mã lệnh trong thành phần COM sẽ yêu cầu những thay đổi tương ứng đối với mã lệnh truy cập thành phần DCOM này. Ưu điểm lớn nhất của SOA là khả năng kết nối mềm dẻo và tái sử dụng. Các dịch vụ có thể được sử dụng trên mọi nền tảng và được viết bằng bất kỳ ngôn ngữ lập trình nào.

Kiến trúc SOA dựa trên hai nguyên tắc thiết kế quan trọng : Module hóa và đóng gói. Module hóa tách vấn đề lớn thành nhiều vấn đề nhỏ để thuận lợi cho việc xử lý, đóng gói là tính năng che giấu dữ liệu và nghiệp vụ bên trong đối với người sử dụng bên ngoài. Dịch vụ được thiết kế phù hợp với kiến trúc SOA cần phải được đóng gói cao ở mức độ cao và có thể dễ dàng tái sử dụng, việc cài đặt đảm bảo tính độc lập và trong suốt về vị trí.

3.6.2 Các dịch vụ

Kiến trúc hướng dịch vụ phân rã các chức năng của hệ thống thành các dịch vụ, mỗi dịch vụ lại được phân rã thành các dịch vụ nhỏ hơn... Có thể nói dịch vụ là nhân tố chủ yếu hình thành nền SOA, nói cách khác kiến trúc SOA lấy dịch vụ làm trọng tâm để xây dựng các ứng dụng. Từ các quy trình, chính sách, nguyên lý hay phương pháp hiện thực trong SOA đều hướng đến 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.



Hình 3.11 Kiến trúc hướng dịch vụ

Về mặt kỹ thuật, dịch vụ (Services) là những sản phẩm phần mềm được định nghĩa một cách rõ ràng thông qua các giao diện. Với góc nhìn của doanh nghiệp, dịch vụ được gắn với một chức năng thực tiễn mà nó đảm nhận trong hệ thống. Mỗi dịch vụ thường kèm theo những chính sách sử dụng như: quyền truy xuất, thời gian truy xuất, mức độ bảo mật, chi phí sử dụng dịch vụ...

Dịch vụ kỹ thuật tái sử dụng (Reusable Technical Services) định nghĩa các dịch vụ chỉ phục vụ cho mục đích nghiệp vụ và được tái sử dụng trong nhiều dòng dịch vụ khác nhau. Các dịch vụ thuộc loại này có thể tính đến dịch vụ truy xuất dữ liệu, đăng nhập, quản lý người dùng.

Phương châm dịch vụ kinh doanh (Lines of Business Services) là tập hợp các dịch vụ hỗ trợ cho nghiệp vụ nhằm mục đích phục vụ trực tiếp hay gián tiếp cho khách hàng thông qua hệ thống tự động. Các kênh dịch vụ kinh doanh thường được định nghĩa thành miền dịch vụ (service domain) như tài chính, bán hàng, quảng cáo, sản xuất, vận chuyển, kỹ thuật, quản lý lợi nhuận, chăm sóc khách hàng.... Tất cả các dịch vụ trong miền dịch vụ nên có sự kết nối với nhau thông qua một từ điển dữ liệu chung để có thể dễ dàng vận hành trong hệ thống. Các dịch vụ trong những miền dịch vụ khác nhau có thể không đồng nhất về từ điển dữ liệu, do đó cần có các chính sách truyền dữ liệu khi có yêu cầu đồng bộ dữ liệu giữa các miền dịch vụ.

Hợp đồng dịch vụ (Service Contracts) là giao diện giữa dịch vụ nghiệp vụ và dịch vụ kỹ thuật nhằm che dấu những hiện thực chi tiết của dịch vụ kỹ thuật. Nền tảng dịch vụ Web (Web Services Platform) gồm các chuẩn và phương tiện giúp các dịch vụ có thể giao tiếp với nhau một cách độc lập với công nghệ hiện thực. Các quy trình và chính sách hướng dẫn của SOA (SOA Governance Policies and Processes) bao gồm các chỉ dẫn cho sự phối hợp của các dịch vụ nhằm đạt mức lợi nhuận cao nhất cho doanh nghiệp.

Các phương pháp và công cụ (SOA Methods and Tools) là các công cụ SOA sẽ dùng trong quá trình quản lý dự án, mô hình dịch vụ, mô hình dữ liệu, quản lý và phát triển hệ thống. Các nguyên lý và chỉ dẫn (SOA Principles and Guidelines) bao gồm các nguyên lý giúp cho các nhà kiến trúc và các nhà phát triển trong quá trình xác định các dịch vụ kỹ thuật và dịch vụ nghiệp vụ. Một số chính sách và nguyên lý quan trọng.

Dịch vụ là khái niệm chính trong kiến trúc hướng dịch vụ, mỗi dịch vụ được định nghĩa bởi một hợp đồng dịch vụ phân biệt rõ ràng giữa chức năng và hiện thực. Các dịch vụ chỉ nên giao tiếp với các dịch vụ khác thông qua những giao diện được định nghĩa một cách rõ ràng. Dịch vụ có thể được truy xuất thông qua những chuẩn dùng trong môi trường giao tiếp rộng như SOAP, WSDL, XML, HTTP, UDDI... và không nên có ràng buộc quá chặt chẽ. Mỗi dịch vụ nên thực hiện những tác vụ rời rạc và cung cấp giao diện đơn giản khi truy xuất nhằm khuyến khích việc tái sử dụng chúng cho các hệ thống về sau. Các dịch vụ nên cung cấp các siêu dữ liệu định nghĩa các ràng buộc cũng như chức năng của chúng. Kiến trúc SOA lấy dịch vụ làm trung tâm, do đó cần cung cấp các công cụ giúp mô hình hóa, phát triển, triển khai, liên kết, quản lý và kiểm tra độ bảo mật của các dịch vụ, đó là các sản phẩm, công nghệ và tiện ích đã được phê chuẩn (Approved Products, Technologies, and Facilities).

3.6.3 Mô hình cặp lỏng

Khái niệm gắn kết (coupling) ám chỉ đến một số điều kiện ràng buộc giữa các thành phần với nhau, chúng có thể là những điều kiện chặt hoặc lỏng, thậm chí có

những ràng buộc không hề được biết trước. Hầu hết mọi kiến trúc phần mềm đều hướng đến tính ràng buộc lỏng giữa các thành phần và gọi là mô hình cặp lỏng. Mức độ gắn kết của mỗi hệ thống ảnh hưởng trực tiếp đến khả năng chỉnh sửa hệ thống, gắn kết càng chặt thì càng ảnh hưởng đến phía sử dụng dịch vụ mỗi khi có thay đổi nào đó xảy ra. Mức độ gắn kết tăng dần khi khi bên sử dụng dịch vụ biết nhiều thông tin ngầm định của bên cung cấp dịch vụ. Ngược lại, nếu bên sử dụng dịch vụ biết ít thông tin chi tiết bên trong dịch vụ trước khi gọi nó thì quan hệ giữa hai bên càng lỏng.

Kiến trúc SOA hỗ trợ gắn kết lỏng thông qua việc sử dụng hợp đồng và nhúng (contract and binding). Người dùng truy vấn đến nơi lưu trữ và cung cấp thông tin dịch vụ để lấy thông tin, một thành phần trong hệ thống cung cấp dịch vụ truy vấn sẽ trả về tất cả những dịch vụ thỏa mãn tiêu chuẩn tìm kiếm, từ đó người dùng chỉ việc chọn dịch vụ cần thiết và thực thi phương thức trên đó theo mô tả dịch vụ. Bên sử dụng dịch vụ không cần phụ thuộc trực tiếp vào cài đặt của dịch vụ mà chỉ dựa trên hợp đồng mà dịch vụ đó hỗ trợ.

Mô hình cặp lỏng giúp gỡ bỏ những ràng buộc điều khiển giữa những thành phần đầu cuối. Mỗi thành phần trong hệ thống phân tán có thể tự quản lý độc lập nhằm tăng hiệu suất, khả năng mở rộng và khả năng đáp ứng cao. Những thay đổi cài đặt cũng được che dấu đi. Gắn kết lỏng đảm bảo tính độc lập giữa bên cung cấp và bên sử dụng nhưng nó đòi hỏi các giao diện phải theo chuẩn và cần một thành phần trung gian quản lý, trung chuyển yêu cầu giữa các thành phần đầu cuối.

3.6.4 Chu kỳ sống dịch vụ

Dịch vụ là một phần của phần mềm trong hệ thống phân tán, do đó chu kỳ sống của dịch vụ cũng gần tương tự như chu kỳ sống của phần mềm. Dịch vụ có thể ở giai đoạn phát triển hoặc đã đưa vào hoạt động sản xuất kinh doanh. Nếu ở giai đoạn phát triển cần phải chỉ ra được các tương tác dịch vụ và chỉ ra những dịch vụ cần thiết phải xây dựng, do đó dịch vụ ở giai đoạn này được hiểu là một phần xây dựng qui trình kinh doanh. Phần mềm đã được đưa vào hoạt động kinh doanh sẽ phát sinh ra nhiều vấn đề và yêu cầu mới cần phải chỉnh sửa hoặc nâng cấp, có một số dịch vụ sẽ được sửa trực tiếp trong khi hệ thống vẫn đang vận hành. Tuy nhiên, nhiều trường hợp phức tạp sẽ đòi hỏi phải tạm thời ngừng dịch vụ để tập trung cho việc sửa đổi và nâng cấp.

3.6.5 Phân loại dịch vụ

Kiến trúc SOA phân các dịch vụ thành ba nhóm: Dịch vụ cơ bản, dịch vụ tích hợp và dịch vụ qui trình, ba loại này liên quan mật thiết đến quá trình cung cấp dịch vụ của kiến trúc SOA. Dịch vụ cơ bản cung cấp các tính năng kinh doanh cơ bản nhất, chúng chưa được phân cho các dịch vụ khác. Đặc điểm của chúng là thời gian chạy tương đối ngắn và thuộc loại không trạng thái, do đó các dịch vụ này rất phù hợp với phương thức gọi đồng bộ. Thực tế, các dịch vụ cơ bản thường được cài đặt để truy nhập dữ liệu và một số nghiệp vụ cơ bản, ví dụ như tạo một người dùng mới hoặc thay đổi mật khẩu...

Các dịch vụ tích hợp được cấu thành từ một số dịch vụ cơ bản, nhìn chung thời gian thực hiện của các dịch vụ này cũng tương đối ngắn và thuộc loại không trạng thái, sự tích hợp ở đây có thể thuộc về một hoặc nhiều nền tảng. Dịch vụ qui trình khác với hai loại trên, nó phản ánh một qui trình kinh doanh, do đó thời gian thực hiện khá dài và thuộc loại có trạng thái.

3.6.6 Trục dịch vụ doanh nghiệp

Trục dịch vụ doanh nghiệp (ESB – Enterprise Service Bus) là hạ tầng kiến trúc cho phép sử dụng các dịch vụ của hệ thống sản xuất, thường triển khai các ứng dụng, các nền tảng và các quy trình nghiệp vụ. Các dịch vụ này được liên kết và trao đổi thông tin với nhau nhưng không sử dụng một loại định dạng dữ liệu chung và cũng không có một chuẩn giao tiếp chung. Nếu cần giao tiếp với các hệ thống bên ngoài, vấn đề tích hợp sẽ mở rộng ra khỏi phạm vi của doanh nghiệp, nó bao chùm lên các hệ thống và quy trình nghiệp vụ của các doanh nghiệp khác nhau.

Những năm gần đây, một số giải pháp như tích hợp ứng dụng doanh nghiệp (EAI - Enterprise Application Integration), doanh nghiệp với doanh nghiệp (B2B - Business to Business), kiến trúc hướng dịch vụ và dịch vụ Web đã tập trung giải quyết những vấn đề liên quan tới tích hợp hệ thống thông tin của các doanh nghiệp. Những giải pháp trên tập trung vào một vài vấn đề về tích hợp, nhưng chúng thường là sản phẩm của một công ty nào đó, giá thành đắt và tốn thời gian triển khai.

Trục dịch vụ doanh nghiệp ESB theo tiêu chuẩn sẽ giải quyết các vấn đề liên quan đến việc tích hợp mà không cần phải loại bỏ những giải pháp đang sử dụng. Mục đích của ESB là làm cho việc tích hợp các ứng dụng và quy trình trở nên thuận tiện hơn bằng cách cung cấp một quy trình phân tán, điều hướng thông minh, bảo mật và có thể tự động chuyển đổi dữ liệu. Trong hệ thống ESB những dịch vụ trên là những dịch vụ nền tảng do đó các ứng dụng không cần phải thi hành riêng biệt những yêu cầu trên theo một cách thức riêng biệt của chúng.

Trục dịch vụ doanh nghiệp ESB giải quyết những điểm yếu của những giải pháp có sẵn bằng cách tạo ra một nền tảng chuẩn cho việc tích hợp. Giải pháp điểm–điểm yêu cầu cứ n thành phần tham gia hệ thống thì phải có n-1 giao diện để có thể giao tiếp được với các thành phần còn lại được thay thế bằng giải pháp trực, mỗi thành phần chỉ cần một giao diện để giao tiếp với trực và như vậy sẽ giao tiếp với các thành phần còn lại. Trục dịch vụ doanh nghiệp ESB đảm bảo giao tiếp phân tán, chuyển hướng, xử lý nghiệp vụ, ổn định và bảo mật, đồng thời cũng cung cấp các dịch vụ có khả năng cắm và chạy (plug-and-play).

3.6.7 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 trục 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.

3.6.8 Các mẫu trao đổi thông điệp

So với kiến trúc dựa trên thành phần, điểm khác biệt chính của SOA là cung cấp khả năng giao tiếp giữa các dịch vụ sử dụng thông điệp dựa trên các giao thức phổ biến như HTTP, FTP, SMTP, và vì vậy kiến trúc SOA mới có khả năng độc lập với nền tảng. Các dịch vụ hoạt động trên nền tảng khác nhau vẫn có thể giao tiếp với nhau nhờ vào các giao diện đã được chuẩn hóa để cộng tác xử lý một tác vụ nào đó. Phương thức trao đổi thông điệp đã được tất cả các nền tảng và ngôn ngữ lập trình hỗ trợ, do đó các dịch vụ trên các nền tảng nào sẽ hoạt động với cấu trúc dữ liệu đặc thù của nền

Hệ thống phân tán

tảng đó. Trao đổi thông điệp có thể thực hiện theo cơ chế không đồng bộ, bên gửi và và bên nhận không cần phải chờ nhau, điều này giúp cho mỗi bên tiếp tục xử lý công việc sau khi gửi thông điệp mà không cần dừng thực thi để chờ thông điệp trả lời.

PTIT

TÀI LIỆU THAM KHẢO

- [1] A. S. Tanenbaum, M. V. Steen, "*Distributed Systems: Principles and Paradigms*", 2nd Edition, Prentice-Hall, 2007.
- [2] G. Coulouris, J. Dollimore, T. Kinberg, G. Blair, "*Distributed systems: Concept and Design*", 5th Edition, Addison-Wesley, 2012.
- [3] N.M. Josuttis, "*SOA in Practice – The Art of Distributed System Design*", O'Reilly, 2007

PTIT