HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

HOÀNG VĂN DŨNG

ỨNG DỤNG RMI VÀO HỆ THỐNG NGÂN HÀNG

CHUYÊN NGÀNH: KHOA HỌC MÁY TÍNH

MÃ SỐ: 60.48.01

Người hướng dẫn khoa học:
TS. HÀ HẢI NAM

TÓM TẮT LUẬN VĂN THẠC SỸ KỸ THUẬT

HÀ NỘI – 2011

MỞ ĐẦU

Phần mềm đang ngày càng trở nên phức tạp quá mức và dường như đang vượt khỏi khả năng kiểm soát của các mô hình phát triển hiện có. Một số công nghệ trong hệ phân tán như CORBA (Common Object Request Broker Architecture – Kiến trúc môi giới yêu cầu đối tượng chung), RMI (Remote Method Invocation – Triệu gọi Phương thức từ xa), SOA (Service Oriented Architecture – Kiến trúc hướng dịch vụ), Webservice có thể giải quyết được vấn đề tích hợp các hệ thống khác nhau được xây dựng trên các nền tảng công nghệ khác nhau.

Nghiên cứu về hệ phân tán bao gồm nhiều chủ đề khác nhau như: Hệ điều hành phân tán, CSDL phân tán, Hệ thống tính toán phân tán.

Hệ thống phân tán có những tính chất như chia sẻ tài nguyên, tính mở, tính trong suốt với người sử dụng, xử lý đồng thời nâng cao hiệu năng, khả năng mở rộng và chịu lỗi tốt, chính những lợi điểm đó của hệ phân tán đã thôi thúc tác giả tìm hiểu và nghiên cứu hệ phân tán mang tính ứng dụng với đề tài: ***Ứng dụng RMI với vào hệ thống ngân hàng***

Luận văn bao gồm 3 chương: Chương 1 giới thiệu lý thuyết hệ phân tán, chương 2 So sánh đánh giá CORBA và RMI, chương 3 xây dựng ứng dụng phân tán RMI vào hệ thống ngân hàng.

Chương 1: GIỚI THIỆU HỆ PHÂN TÁN

1.1 HỆ PHÂN TÁN

1.1.1 Giới thiệu

Có nhiều định nghĩa về hệ phân tán

Định nghĩa 1: Hệ phân tán là tập hợp các máy tính tự trị được kết nối với nhau bởi một mạng máy tính và được cài đặt phần mềm hệ phân tán.

Định nghĩa 2: Hệ phân tán là một hệ thống có chức năng và dữ liệu phân tán trên các trạm (máy tính) được kết nối với nhau bởi một mạng máy tính.

Định nghĩa 3: Hệ phân tán là một tập các máy tính độc lập giao tiếp với người dùng như một hệ thống thống nhất, toàn vẹn.

Như vậy, có thể nói : Hệ phân tán = mạng máy tính + phần mềm hệ phân tán.

Phân loại hệ phân tán:

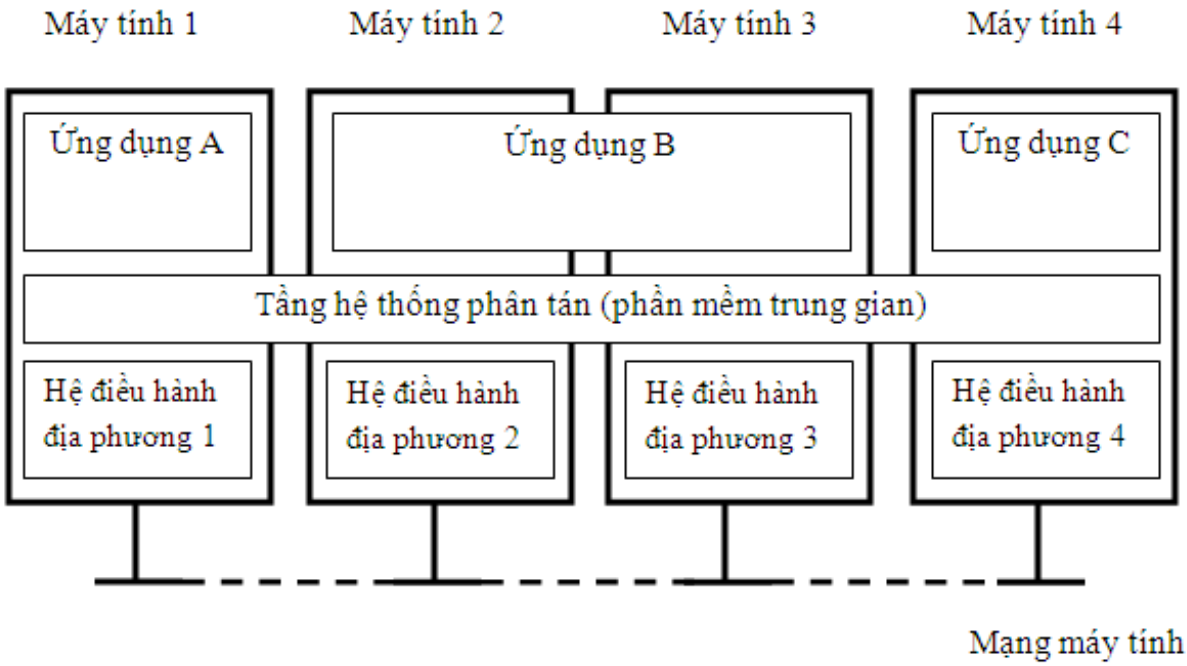
Trước đây, hệ phân tán được chia thành ba loại : hệ điều hành hệ phân tán, cơ sở dữ liệu hệ phân tán và các hệ thống tính toán hệ phân tán.

Ngày nay, hệ phân tán được phân chia như sau:

- Hệ phân tán mang tính hệ thống: hệ điều hành phân tán.
- Hệ phân tán mang tính ứng dụng: các hệ thống truyền thông điệp phân tán.

Hệ phân tán thường che giấu nhiều vấn đề phức tạp liên quan đến việc phân tán các tiến trình, dữ liệu, và điều khiển. Tuy nhiên, tính trong suốt phân tán không chỉ có giá trị về hiệu năng.

Một trong những cách tiếp cận trong xây dựng hệ phân tán là sử dụng các hệ thống phần mềm trung gian (middleware) để thực hiện chức năng của hệ phân tán. Phần mềm lớp trung gian này sẽ che giấu sự không đồng nhất của các hệ thống ở lớp bên dưới và cung cấp sự trong suốt của hệ phân tán cho các ứng dụng phân tán ở lớp trên nó. Hình 1.1 minh họa cách tiếp cận này.



Hình 1.1 Hệ phân tán được tổ chức như middleware.

Tầng middleware mở rộng trên nhiều máy tính và cung cấp mỗi ứng dụng cùng giao diện

Hình 1.1 gồm bốn máy tính nối mạng và ba ứng dụng, trong đó ứng dụng B được phân tán trên các máy tính 2 và 3. Mỗi ứng dụng được cung cấp cùng một giao diện.

Bốn mục tiêu quan trọng phải được đáp ứng khi xây dựng hệ phân tán: truy cập tài nguyên dễ dàng, tài nguyên được phân tán qua mạng, có tính mở, có thể mở rộng.

1.1.2 Tính chất hệ phân tán

1.1.2.1 Tính trong suốt phân tán

Một mục tiêu quan trọng của hệ phân tán là để che giấu sự kiện mà nó xử lý và tài nguyên phân tán trên nhiều máy tính. Khả năng thể hiện của một hệ phân tán tới người dùng và các ứng dụng như thể nó là chỉ có một hệ thống máy tính đơn gọi là tính trong suốt phân tán.

1.1.2.2 Tính mở

Một tính chất quan trọng của hệ phân tán là tính mở. Hệ phân tán mở là hệ thống cung cấp dịch vụ theo quy tắc chuẩn mô tả cú pháp và ngữ nghĩa của các dịch vụ.

1.1.2.3 Khả năng mở rộng

Khả năng mở rộng của một hệ thống có thể đo được ít nhất theo ba chiều khác nhau (Neuman, 1994):

Thứ nhất, hệ thống có thể được mở rộng với kích thước

Thứ hai, hệ thống khả năng mở rộng địa lý

Thứ ba, hệ thống có thể được mở rộng về mặt quản trị

1.2 KIẾN TRÚC HỆ PHÂN TÁN

Hệ phân tán thường là phần phức tạp của phần mềm trong đó các thành phần được phân tán trên nhiều máy.

Tổ chức hệ phân tán chủ yếu là các thành phần phần mềm tạo thành hệ thống.

Mục tiêu quan trọng của hệ phân tán là tách các ứng dụng riêng biệt từ nền cơ sở bằng cách cung cấp một lớp phần mềm trung gian (middleware).

1.2.1 Kiểu kiến trúc

Kiến trúc giống như tổ chức logic của hệ phân tán vào trong các thành phần phần mềm, cũng được gọi là kiến trúc phần mềm (Bass et al., 2003).

Một số kiểu kiến trúc quan trọng được xác định trong các hệ phân tán là:

- Kiến trúc phân tầng
– Kiến trúc dựa trên đối tượng
– Kiến trúc trung tâm dữ liệu
– Kiến trúc dựa trên sự kiện

Điều làm cho các kiến trúc phần mềm quan trọng đối với hệ phân tán là tất cả đều nhằm mục đích đạt được sự trong suốt phân tán. Tuy nhiên, như chúng ta đã lập luận, yêu cầu trong suốt phân tán làm cho cân bằng giữa hiệu suất, khả năng chịu lỗi, tính dễ lập trình.

1.2.2 Kiến trúc hệ thống

Xem xét hai kiến trúc hệ thống: Kiến trúc tập trung và Kiến trúc phi tập trung

1.3. TRUYỀN THÔNG TRONG HỆ PHÂN TÁN

Truyền thông đa xử lý (InterProcess) là trái tim của tất cả các hệ phân tán. Không thể nghiên cứu các hệ phân tán mà không xem xét cẩn thận những cách mà các tiến trình trên các máy khác nhau có thể trao đổi thông tin. Truyền thông trong các hệ phân tán luôn luôn dựa trên truyền thông thông điệp mức độ thấp được cung cấp bởi các mạng cơ sở.

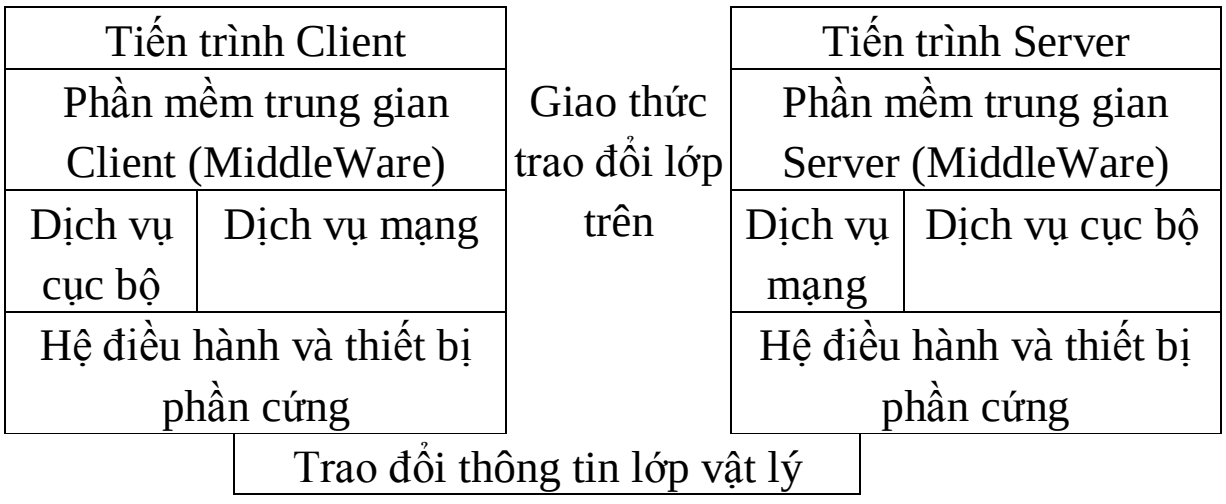
Các quy tắc mà các tiến trình giao tiếp phải tuân thủ, được gọi là giao thức, và tập trung vào việc cơ cấu lại các giao thức trong

mẫu của các lớp. Các mô hình sử dụng rộng rãi trong truyền thông: Gọi thủ tục từ xa (Remote Procedure Call - RPC), Phần mềm trung gian hướng thông điệp (Message-Oriented Middleware - MOM), Truyền thông hướng luồng (Data streaming) và truyền thông Multicasting.

1.4 MÔ HÌNH CLIENT/SERVER

Mô hình Client/Server đóng vai trò quan trọng trong các hệ phân tán, nó có các đặc trưng sau:

- Client và Server là các mô đun chức năng với các giao diện xác định
- Quan hệ Client/Server được thiết lập giữa hai mô đun khi Client đưa ra yêu cầu dịch vụ và được Server đáp lại
- Trao đổi thông tin giữa các mô đun được thực hiện thông qua có chế truyền thông điệp
- Trao đổi thông điệp giữa Client và Server thường được thực hiện theo cơ chế hỏi đáp.



Hình 1.4.1 Các thành phần cơ bản trong mô hình Client/Server

1.5 PHẦN MỀM TRUNG GIAN HỆ PHÂN TÁN

Phần mềm trung gian đơn giản hoá 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).

1.5.1 Phương pháp gọi thủ tục từ xa

Các ứng dụng Client kết nối với máy Server và sử dụng các dịch vụ do máy Server cung cấp. Các bước gọi thủ tục trên máy Server được thực hiện tương tự như gọi thủ tục trên máy cục bộ, Client chuyển các tham số đầu vào khi gọi thủ tục và dịch vụ trên Server sẽ kiểm tra trính hợp lệ của các tham số đó, thực hiện tính toán và trả về các giá trị theo yêu cầu của ứng dụng Client.

1.5.2 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 Client/Server, trong đó tập trung vào việc:

- Chuẩn hoá giao diện lập trình API
- Chuẩn hoá giao thức trao đổi thông tin

1.6. ĐỒNG BỘ TRONG HỆ PHÂN TÁN

Trong hệ phân tán, mỗi máy tính là một đồng hồ nên việc đồng bộ các đồng hồ này là rất cần thiết và rất khó khăn.

1.6.1 Đồng hồ vật lý

Chúng ta có nhiều cách để xác định thời gian. Phổ biến nhất là các hệ đếm thời gian theo thiên văn và ở đây là mặt trời. Có 23h một ngày và 3600 giây. Một giây mặt trời được tính là 1/8600 của một ngày mặt trời. Một trong những mô hình để tính thời gian áp dụng phương pháp trên là Internatinal Atomic Time viết tắt là TAI.

1.6.2 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.

1.6.3 Trạng thái tổng thể

Việc xác định trạng thái tổng thể của hệ thống rất có ích. Một trong những phương pháp được đưa ra là chụp nhanh phân tán (Distributed Snapshort) cùng khái niệm lát cắt (cut).

1.7. TÍNH NHẤT QUÁN VÀ NHÂN BẢN CỦA HỆ PHÂN TÁN

1.7.1 Tính nhân bản

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.

1.7.2 Tính nhất quán

Mô hình nhất quán được sử dụng trong hệ phân tán như hệ thống bộ nhớ chia sẻ phân tán, lưu trữ dữ liệu phấn tán như hệ thống file, CSDL, hệ thống nhân bản lạc quan (optimistic) hoặc web caching.

1.8. CHỊU LỖI VÀ AN TOÀN CỦA HỆ PHÂN TÁN

1.8.1 Chịu lỗi

Tính chịu lỗi liên quan nhiều tới khái niệm hệ có thể tin cậy được (dependable system). Thuật ngữ "có thể tin cậy được" bao gồm các thuộc tính sau:

Tính sẵn sàng (availability)

Tính tin cậy (Reliability)

Tính an toàn (Safety)

Khả năng bảo trì (Maintainability)

1.8.2 An toàn

Có 4 cơ chế an toàn, an ninh được đưa ra:

Mật mã (Cryptography)

Xác thực (Authentication)

Ủy quyền (Authorization)

Kiểm toán (Auditing)

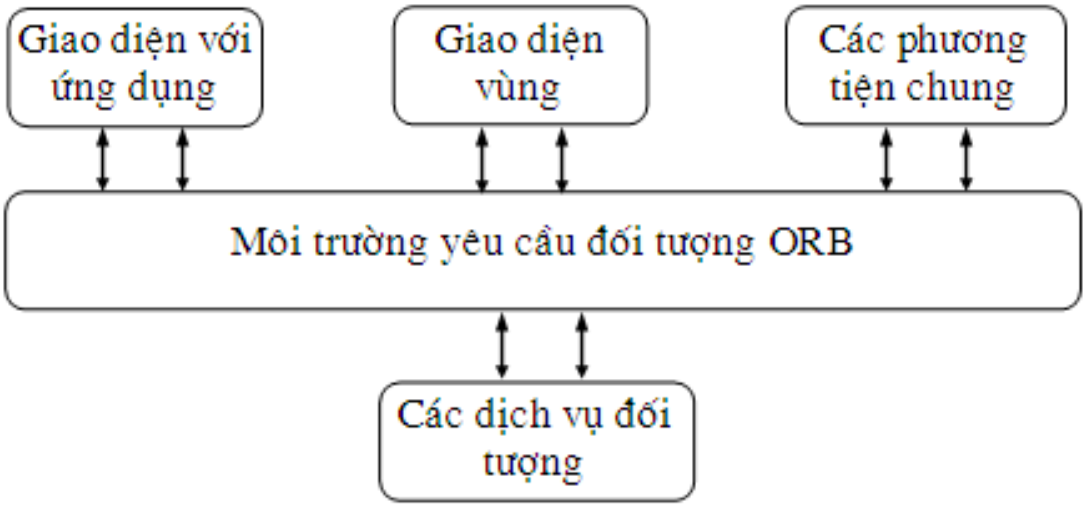
Chương 2: SO SÁNH ĐÁNH GIÁ RMI VÀ CORBA

2.1 GIỚI THIỆU

Kiến trúc CORBA do nhóm OMG 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...

2.2 KIẾN TRÚC 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.



Hình 2.2 1 Các thành phần cơ bản trong kiến trúc CORBA

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

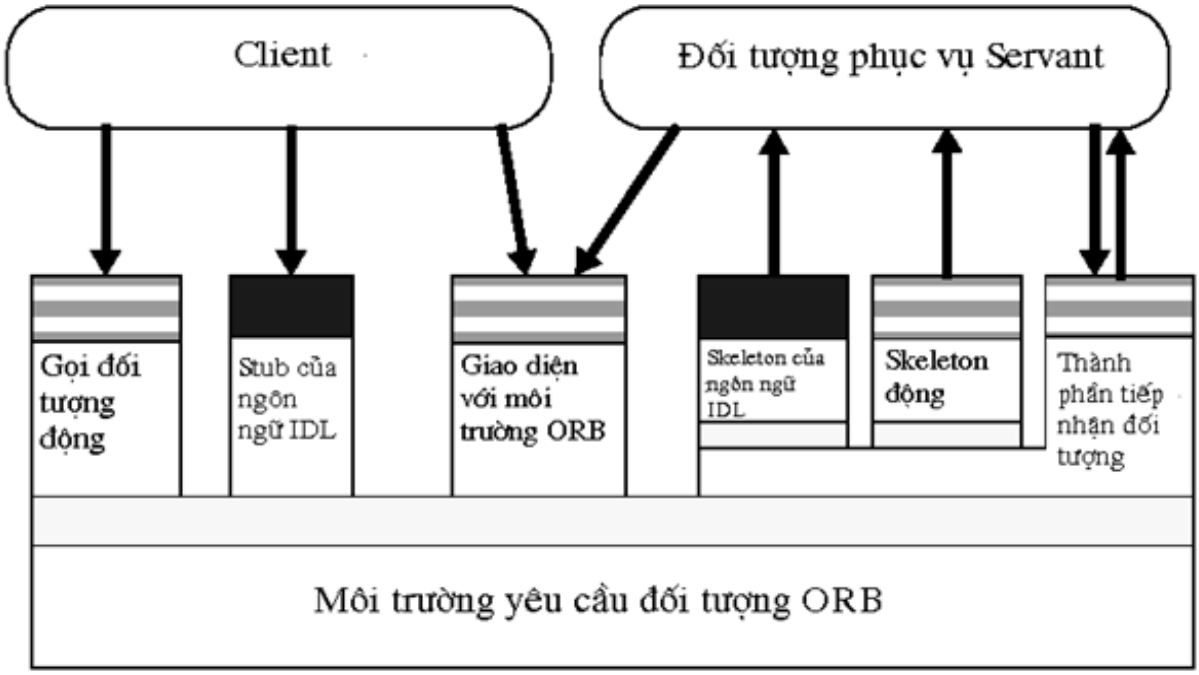
- Không phụ thuộc hệ điều hành và thiết bị phần cứng
- Không phụ thuộc ngôn ngữ lập trình
- Không phụ thuộc vị trí vật lý cài đặt các ứng dụng

2.2.1Môi 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 2.2 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 Client

- Cung cấp tất cả các thủ tục cần thiết để tìm ra đối tượng thực hiện yêu cầu của Client.
- 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 Client chỉ cần liên hệ với giao diện giao diện hoàn toàn độc lập với vị trí cài đặt đối tượng thực hiện.



Hình 2.2 2 Kiến trúc và giao diện của ORB

- **Client:** là đối tượng yêu cầu dịch vụ.
- **Servant:** là các đối tượng phục vụ yêu cầu của các Client, nó định nghĩa các thao tác, hỗ trợ giao diện IDL của CORBA.
- **ORB**: là môi trường trung gian thiết lập quan hệ Client/Server giữa các đối tượng. bằng cách sử dụng ORB, đối tượng Client có thể gọi các hàm trên máy cục bộ hoặc trên mạng.
- **Giao diện với môi trường ORB**: Cung cấp các giao diện để Client và Servant kết nối với nhau qua môi trường ORB.

- **Stub và Skeleton của ngôn ngữ IDL**: Stub gồm các lệnh cho phép Client truy nhập tới các thành phần của Server tương tự như cơ chế gọi gọi thủ tục từ xa RPC. Skeleton gồm các lệnh trên Server để liên lạc với các thành phần CORBA, nó đóng vai trò cầu nối giữa ORB với đối tượng thực hiện trên Server.
- **Thành phần Object Adapter (BOA hoặc POA)**: Hỗ trợ ORB trong việc phân phát các yêu cầu đến các đối tượng thực hiện.

2.3. KIẾN TRÚC JAVA RMI

Mục đích thiết kế kiến trúc RMI để tạo một mô hình đối tượng phân tán Java được tích hợp trong ngôn ngữ lập trình Java và mô hình đối tượng cục bộ. Kiến trúc RMI đã thành công, tạo một hệ thống an toàn và mạnh mẽ trong mô hình kiến trúc Java với khả năng tính toán phân tán.

Giao diện: trái tim của RMI

Kiến trúc RMI được dựa trên một nguyên tắc quan trọng: định nghĩa các hành vi và cài đặt các hành vi đó là các khái niêm riêng biệt. RMI cho phép mã chương trình định nghĩa hành vi và mã chương trình cài đặt các hành vi còn lại riêng rẽ và chạy độc lập trên các máy ảo Java (JVM).

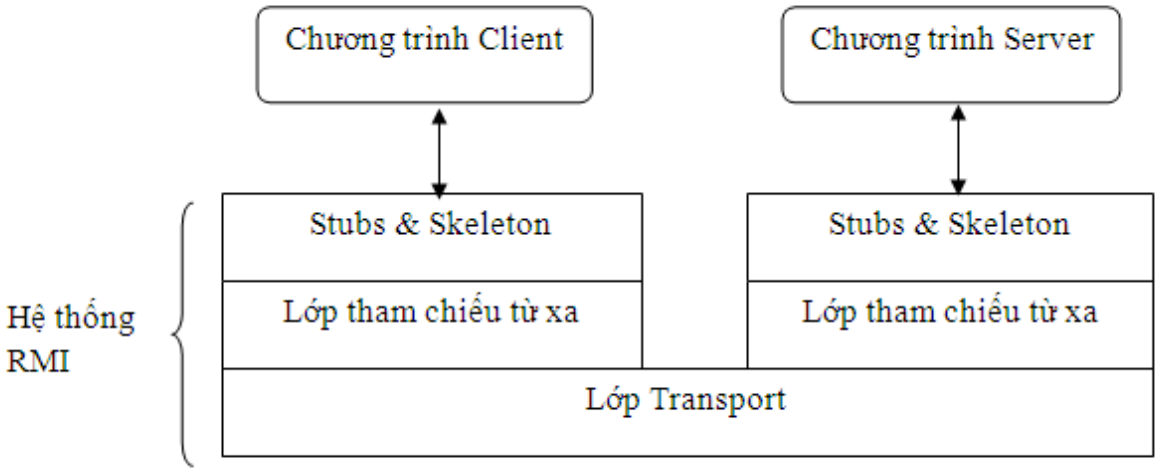
Tầng kiến trúc RMI

Cài đặt RMI cơ bản được xây dựng từ 3 tầng trừu tượng:

Thứ nhất là tầng Stub và Skeleton, nằm phía dưới các khung nhìn của nhà phát triển. Lời gọi phương thức các tầng này được tạo bởi client tới các biến tham chiếu giao diện và chuyển hướng các lời gọi đó tới dịch vụ RMI từ xa.

Lớp tiếp theo là tầng tham chiếu từ xa (Remote Reference Layer). Lớp này quản lý các tham chiếu được tạo từ client tới các đối tượng dịch vụ từ xa.

Lớp transport được dựa trên kết nối TCP/IP giữa các máy tính trong một mạng. Nó cung cấp kết nối cơ bản, cũng như một số chiến lược thâm nhập tường lửa.



Hình 2.3 1 Mô hình kiến trúc Java RMI

Bằng cách sử dụng một kiến trúc nhiều lớp, mỗi lớp có thể được tăng cường hoặc thay thế mà không ảnh hưởng đến phần còn lại của hệ thống. Ví dụ, lớp vận chuyển có thể được thay thế bởi một lớp UDP/IP mà không ảnh hưởng đến các tầng trên.

2.4 So sánh đánh giá CORBA và RMI

So sánh RMI và CORBA không thể hiện một giải pháp tối ưu - không phải cái này "tốt" hơn cái khác. Các đặc tính của hai công nghệ này bổ trợ nhau trong tình huống khác nhau. So sánh về RMI và CORBA giúp làm nổi bật những điểm mạnh và điểm yếu riêng, có thể ứng dụng một công nghệ trên vào một bài toán cụ thể phụ thuộc phần lớn vào mục đích mà nó được sử dụng, kinh nghiệm

của các nhà phát triển, những người thiết kế, cài đặt và duy trì hệ phân tán, dù hệ thống không dùng Java được định để truy cập hệ thống bây giờ hoặc trong tương lai.

2.4.1 RMI ưu và khuyết điểm

Ưu điểm

Nhiều trên các nền tảng di động

Có thể giới thiệu mã mới tới JVM ngoài

Người phát triển Java có thể đã có kinh nghiệm với RMI (có sẵn từ JDK1.02)

Hệ thống hiện tại có thể đã sử dụng RMI - chi phí và thời gian để chuyển đổi sang một công nghệ mới có thể bị hạn chế.

Nhược điểm

Chỉ gắn với các nền tảng hỗ trợ Java

Mối đe dọa bảo mật với thực thi mã từ xa, và hạn chế về chức năng thực thi bởi các hạn chế bảo mật.

Vênh kiến thức cho các nhà phát triển mà không có kinh nghiệm RMI có thể so sánh với CORBA

Chỉ có thể hoạt động với các hệ thống Java - không hỗ trợ cho các hệ thống được viết bằng C ++, Ada, Fortran, Cobol, và những người khác.

2.4.2 CORBA ưu và khuyết điểm

Ưu điểm

Dịch vụ có thể được viết bằng nhiều ngôn ngữ khác nhau, thực thi trên nhiều nền tảng khác nhau, và truy cập bằng bất kỳ ngôn ngữ với một ánh xạ ngôn ngữ định nghĩa giao diện (IDL).

Với IDL, giao diện rõ ràng là tách khỏi cài đặt, và người phát triển có thể tạo ra cài đặt khác nhau dựa trên cùng một giao diện.

CORBA hỗ trợ dữ liệu nguyên gốc, và một loạt các cấu trúc dữ liệu, như các tham số

CORBA là lý tưởng để sử dụng với các hệ thống di sản (legacy system), và để đảm bảo rằng các ứng dụng bằng được viết sẽ được truy cập trong tương lai.

CORBA là một cách dễ dàng để liên kết các đối tượng và các hệ thống với nhau

Hệ thống CORBA có thể cung cấp hiệu năng lớn hơn

Nhược điểm

Mô tả dịch vụ yêu cầu sử dụng một ngôn ngữ định nghĩa giao diện (IDL) mà phải có kiến thức sẵn. Cài đặt hoặc sử dụng dịch vụ yêu cầu một ánh xạ IDL tới ngôn ngữ yêu cầu - một văn bản cho một ngôn ngữ không được hỗ trợ sẽ có được một số lượng lớn công việc.

IDL vào công cụ ánh xạ ngôn ngữ tạo ra mã stubs dựa trên giao diện - một số công cụ không thể tích hợp những thay đổi mới với mã hiện có.

CORBA không hỗ trợ việc chuyển giao đối tượng, hoặc mã.

Tương lai là không chắc chắn - nếu CORBA không đạt được thông qua đầy đủ của ngành công nghiệp, sau đó CORBA cài đặt trở thành các hệ thống di sản.

Đào tạo một số vẫn còn cần thiết, và đặc tả CORBA vẫn trong tình trạng liên tục.

Không phải tất cả các lớp của các ứng dụng cần thời gian thực hiệu suất, và tốc độ có thể được giao dịch giảm so với tính dễ sử dụng cho các hệ thống hoàn toàn Java.

Chương 3: ỨNG DỤNG RMI VÀO HỆ THỐNG NGÂN HÀNG

3.1 GIỚI THIỆU

Hệ thống ngân hàng là một ứng dụng duy trì một tài khoản của khách hàng trong một Ngân hàng

Hệ thống cung cấp Khách hàng truy cập tạo tài khoản (Create account), gửi tiền vào tài khoản (deposit)/rút tiền (withdraw) từ tài khoản của KH, và hiển thị các báo cáo tới tất cả các Tài khoản.

Hệ thống banking phân tán bao gồm một hoặc nhiều server và nhiều ATM (clients). Server quản lý tất cả thông tin tài khoản của người dùng.

Xây dựng các ứng dụng/đối tượng phân tán (Distributed object) được cài đặt trên các server và client, từ client thực hiện lời gọi từ xa (RMI) để thực hiện giao dịch deposit hay withdraw,… Khi server nhận được các yêu cầu xử lý từ client, server sẽ xử lý các yêu cầu của client và phản hồi kết quả tới client.

3.2 MÔ HÌNH UML CỦA HỆ THỐNG

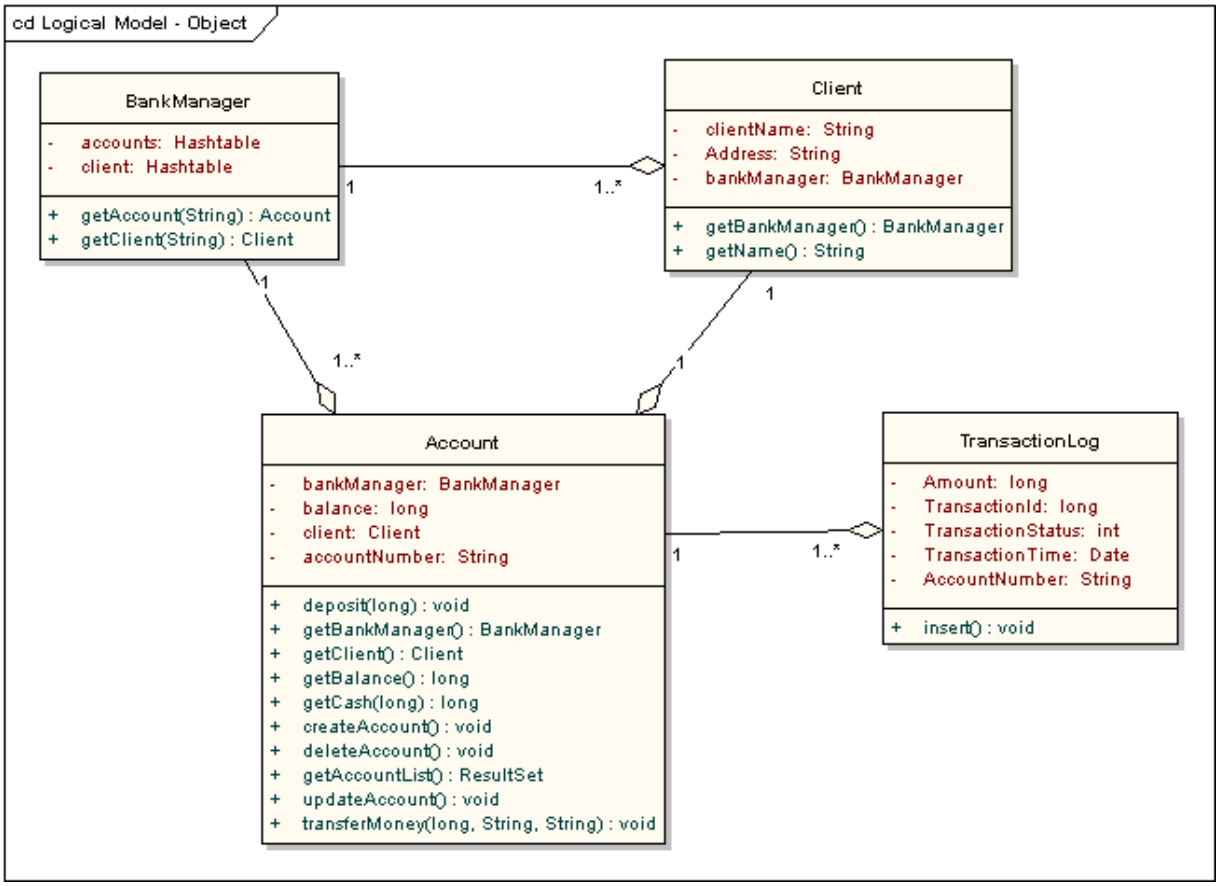
Các đối tượng tham gia hệ thống:

BankManager: là tổ chức cung cấp dịch vụ tài khoản hiện hành cho các client.

Client: là cá nhân hoặc một công ty sở hữu một tài khoản trong ngân hàng.

Account: là một thỏa thuận pháp lý giữa ngân hàng và khách hàng nêu rõ các điều kiện cho việc sử dụng tài khoản hiện tại.

TransactionLog: Ghi lại lịch sử các giao dịch deposit hoặc withdraw



Hình 3. 1 Mô hình UML định nghĩa các đối tượng cài đặt dựa trên các Interface

Các chức năng thực hiện các hoạt động chu kỳ sống của tài khoản hiện hành:

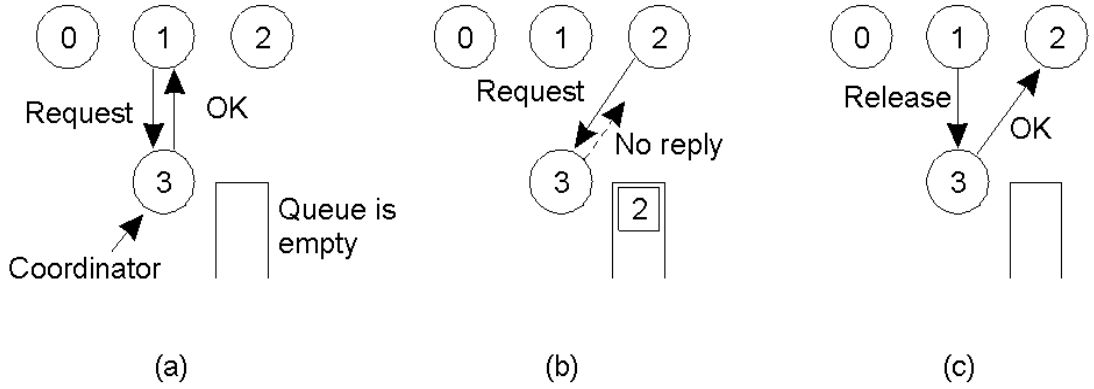
- createAccount: Tạo tài khoản hiện hành
- Mở tài khoản
- getBalance: Xem chi tiết tài khoản
- deposit: Chức năng gửi tiền vào tài khoản
- getCash: rút tiền (withdraw) từ tài khoản
- transferMoney: chuyển khoản
- getAccountList: Thống kê danh sách tất cả các tài khoản
- deleteAccount: Đóng tài khoản

3.3 ĐIỀU KHIỂN TƯƠNG TRANH

Vấn đề tương tranh xẩy ra khi có nhiều người dùng cùng thực hiện giao dịch tại cùng một thời điểm, một tài khoản cùng lúc được yêu cầu thực hiện nhiều giao dịch (một tài khoản A vừa thực hiện rút tiền, nhưng cùng thời điểm cũng có một tài khoản B chuyển tiền sang tài khoản A)

Có nhiều thuật toán điều khiển tương tranh (concurrency control) như: thuật toán Centralized, thuật toán Decentralized, thuật toán phân tán, thuật toán Token ring, trong khuôn khổ đề tài học viên xin trình bày thuật toán Centralized.

Giải thuật tập trung (Centralized Algorithm)



a) Tiến trình 1 yêu cầu điều phối cho phép nhập vào vùng giới hạn. Được cấp phép

b). Tiến trình 2 sau đó yêu cầu cho phép vào cùng vùng giới hạn. Điều phối không trả lời.

c). Khi Tiến trình 1 ra khỏi vùng giới hạn, nó báo với điều phối, sau đó điều phối trả lời tới 2 [2]

Giả thiết: mỗi tiến trình có một số ID duy nhất. Tiến trình được bầu chọn làm điều phối là tiến trình có số hiệu ID cao nhất.

Nội dung thuật toán: Khi một tiến trình nào đó cần vào vùng giới hạn (Critial Section - CS) nó sẽ gửi một thông điệp xin cấp quyền.

Nếu không có một tiến trình nào đang trong vùng giới hạn thì tiến trình điều phối sẽ gửi phản hồi cho phép. Còn nếu có một tiến trình khác đang ở trong vùng tới 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 tới 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.

Coordinator

```
loop
   receive(msg);
     case msg of
       REQUEST: if nobody in CS
                then reply GRANTED
                else queue the REQ;
                     reply DENIED
       RELEASE: if queue not empty then
                 remove 1st on the queue
                 reply GRANTED
     end case
 end loop
```

Client

```
   send(REQUEST);
   receive(msg);
   if msg != GRANTED then receive(msg);
       enter CS;
       send(RELEASE)
```

Đánh giá: Thuật toán này có đảm bảo sự tồn tại duy nhất một tiến trình trong vùng tới hạn và chỉ cần 3 thông điệp để thiết lập là:

Request - Grant - Release. Nhược điểm duy nhất là nếu tiến trình điều phối bị hỏng thì hệ thống sẽ sụp đổ. Vì nếu một tiến trình đang trong trạng thái Block nó sẽ không thể biết được tiến trình điều phối có bị DEAD hay không. Trong một hệ thống lớn nếu chỉ có một tiến trình điều phối sẽ xuất hiện hiện tượng thắt cổ chai.

Hệ thống gồm ba phần chính:

- RMI Registry giữ tham chiếu đến các dịch vụ từ xa.
- Chương trình RMI cài đặt trên máy chủ tạo ra các dịch vụ từ xa, đăng ký với Registry và đợi các yêu cầu của client.
- Chương trình RMI phía client. Một chương trình có tham chiếu đến đối tượng dịch vụ từ xa từ RMI Registry và sau đó sử dụng các dịch vụ.

Đánh giá kết quả thử nghiệm

Xây dựng ứng dụng phân tán sử dụng RMI vẫn tuân theo mô hình 3 lớp, hệ thống được an toàn bảo mật cao, khả năng phân tải tốt khi sử dụng nhiều máy tính để thực hiện các chức năng tiến trình hệ thống, các ứng dụng cài đặt trên nhiều máy tính khác nhau nhưng khi sử dụng người dụng cảm nhận như đang thao tác với một hệ thống độc lập duy nhất tạo sự trong suốt với người sử dụng.

Áp dụng giải thuật centralized để xử lý vấn đề tương tranh giữa các tiến trình trong hệ thống, giải quyết được vấn đề có nhiều giao dịch thực hiện cùng một tài khoản trong ngân hàng. Tuy nhiên giải thuật này vẫn có hạn chế trong một hệ thống lớn có nhiều tiến trình cần xử lý đồng thời mà chỉ có một tiến trình điều phối thì sẽ dẫn tới hiện tượng thắt cổ trai.

KẾT LUẬN

Chương 1 trình bày tổng quan về hệ phân tán bao gồm những vấn đề chính sau: định nghịa hệ phân tán, kiến trúc hệ phân tán, mô hình hệ phân tán, truyền thông, đồng bộ, phần mềm trung gian, nhất quán và nhân bản, an toàn của hệ phân tán.

Chương 2 có sử dụng những kết quả nghiên cứu và thực nghiệm so sánh đánh giá RMI và CORBA, đưa ra mô hình kiến trúc của CORBA và RMI, so sánh nhưng ưu và nhược điểm của 2 công nghệ này.

Chương 3 Ứng dụng RMI để xây dựng bài toán Quản lý Ngân hàng phân tán. Từ phân tích thiết kế hệ thống sử dụng UML, tới cài đặt chương trình bằng Java RMI và demo chương trình.

KIẾN NGHỊ VÀ HƯỚNG NGHIÊN CỨU TIẾP THEO

Bài toán có tính ứng dụng thực tế rất cao khi mà nhu cầu thanh toán qua tài khoản ngân hàng để mua sắm hàng hóa, chi trả các hóa đơn hàng tháng của gia đình như hóa đơn điện, nước, điện thoại , internet, … Nếu có điều kiện nghiên cứu và phát triển tiếp tác giả sẽ hoàn thiện hệ thống để ứng dụng bài toán thực tế hiện nay cho các ngân hàng ở nước ta.