CORBA

Java Implementation

thangld@uit.edu.vn
Khoa Mạng máy tính và Truyền thông
Đại học Công nghệ Thông tin

Nội dung

- Tổng quan
- RMI vs. CORBA
- Kiến trúc CORBA
- Qui trình CORBA
- CORBA và Java
- Ứng dụng CORBA
- Phần đọc thêm

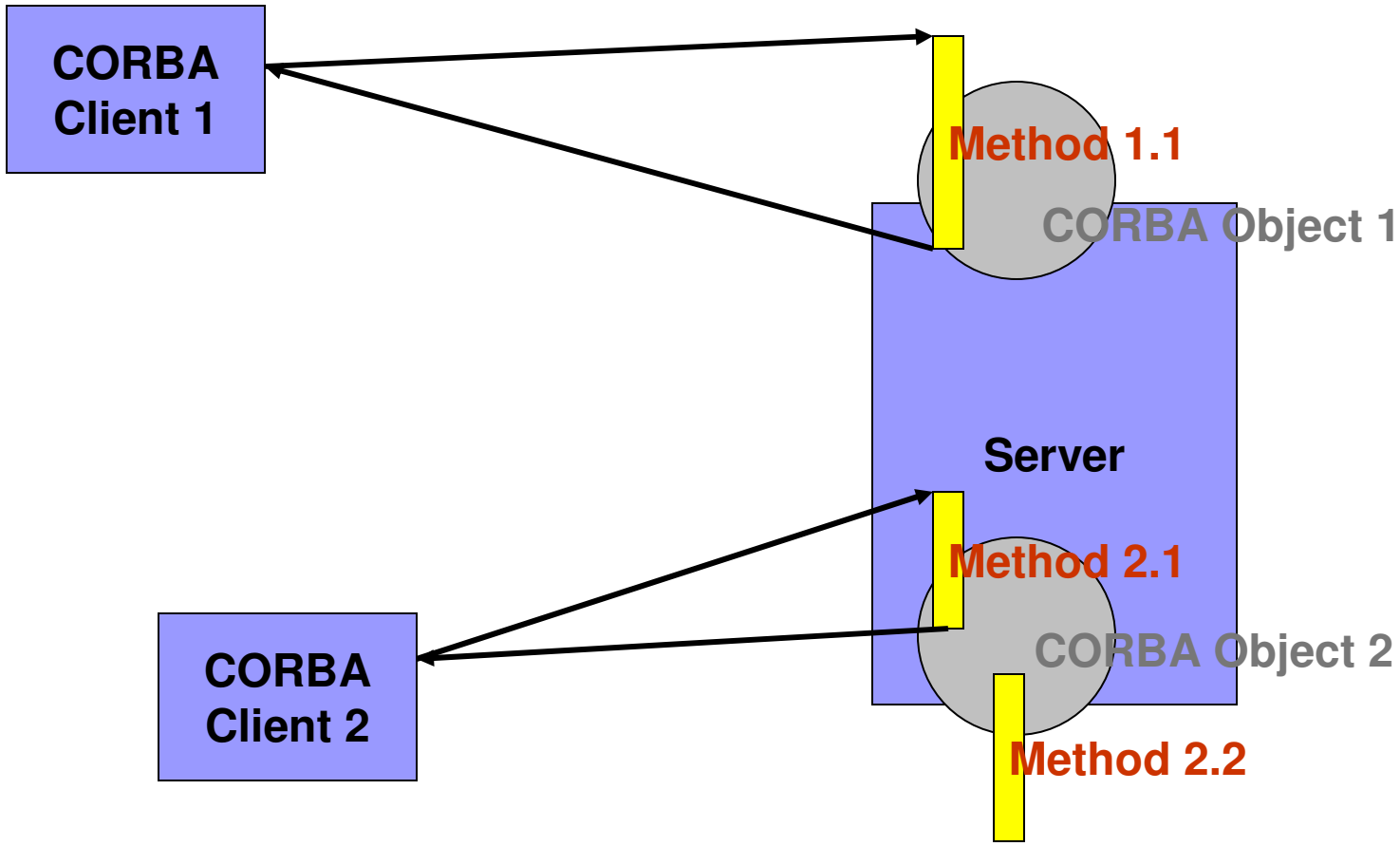
Tổng quan [1]

- CORBA, được phát triển bởi OMG (Object Management Group), cho phép tích hợp các đối tượng thực thi trên các hệ thống khác nhau (*www.omg.org*)
- CORBA cung cấp nền tảng và cơ sở lý thuyết cho phép các đối tượng được truy xuất từ xa qua mạng máy tính

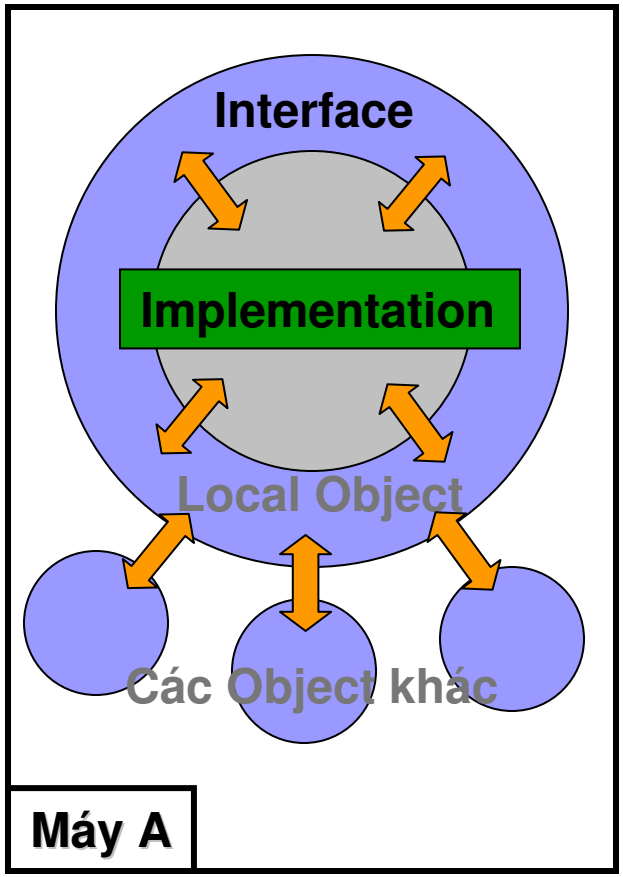
Tổng quan [2]

- Mục tiêu của CORBA là cho phép triệu gọi các phương thức của các đối tượng đang thực thi trên các máy tính khác nhau
  - Một ứng dụng thực thi trên máy tính A triệu gọi một phương thức của một đối tượng đang thực thi trên máy tính B
  - Phương thức được triệu gọi thực thi trên máy tính B, do đó, sử dụng tài nguyên (CPU, RAM, …) của máy tính B
  - Ứng dụng trên máy tính A có thể phải chờ phương thức hoàn tất quá trình thực thi để nhận kết quả trả về

Triệu gọi phương thức từ xa

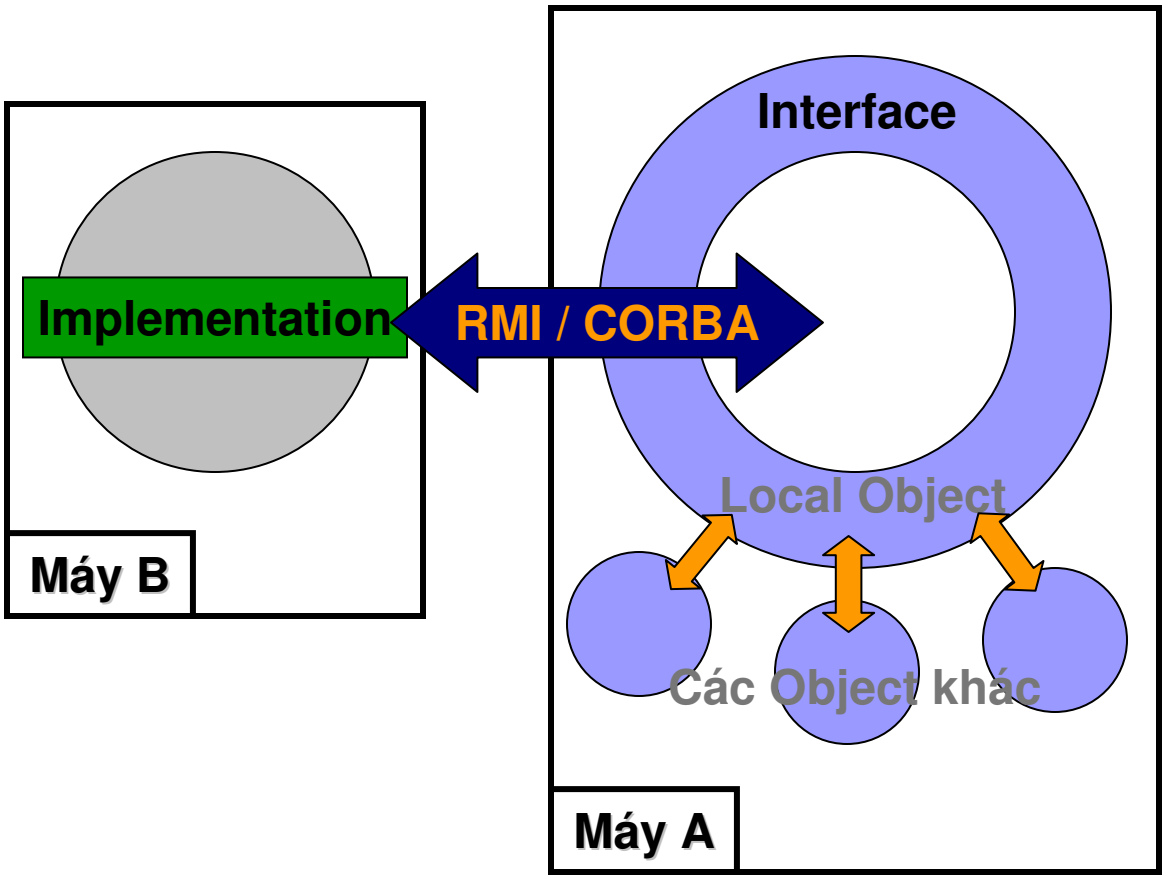


Nguyên tắc cơ bản



Triệu gọi phương thức cục bộ

Triệu gọi phương thức từ xa



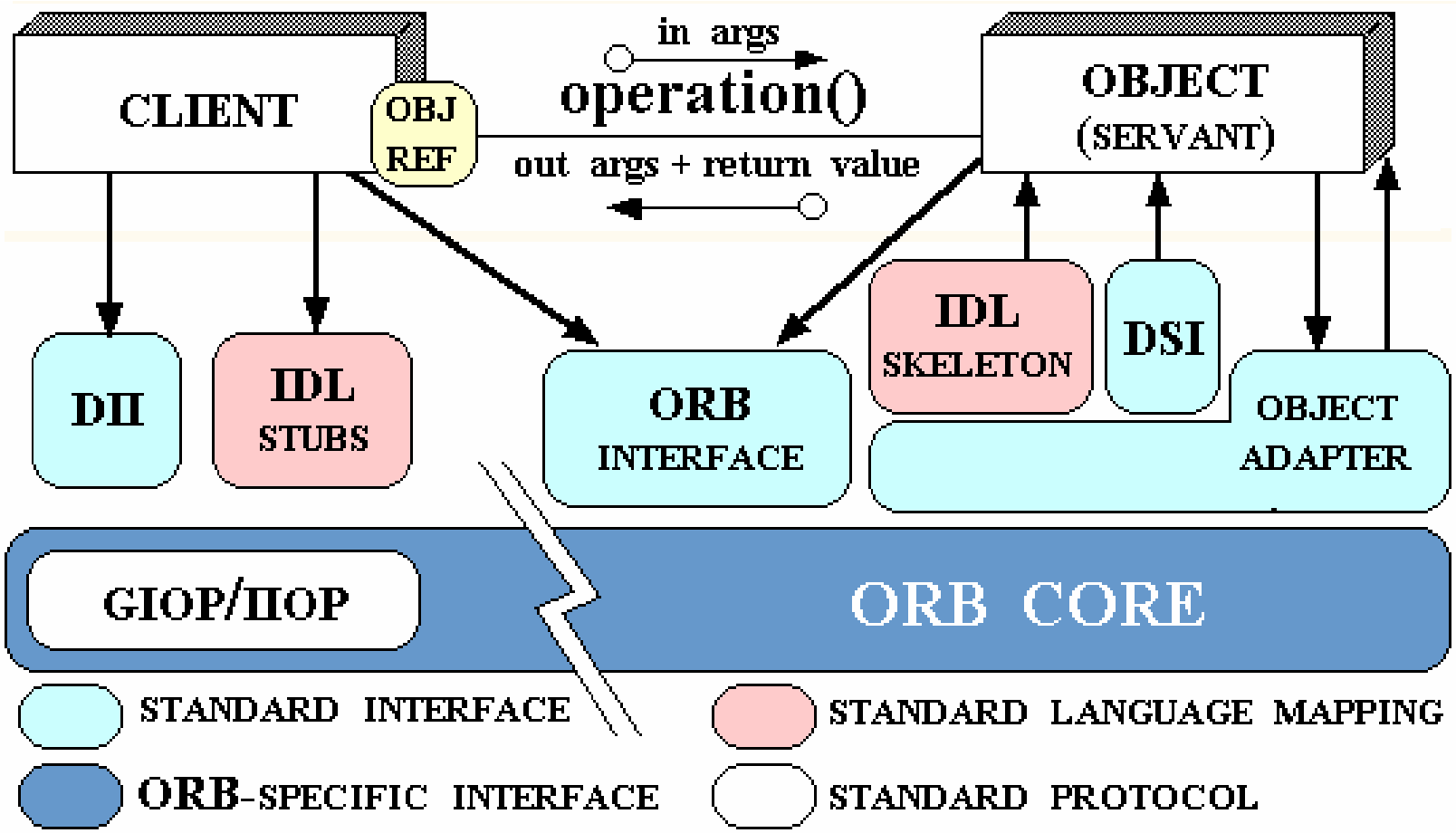
CORBA & IDL

- CORBA không phụ thuộc vào ngôn ngữ lập trình, platform
- CORBA sử dụng ngôn ngữ định nghĩa interface độc lập (IDL – Interface Definition Language)
  - Từ khóa *in* được sử dụng cho các biến vào
  - Từ khóa *out* được sử dụng cho các biến ra
- Dựa trên IDL interface, chúng ta phát sinh mã lệnh sử dụng ở client-side và server-side phù hợp với ngôn ngữ lập trình cụ thể

RMI vs. CORBA

- RMI cung cấp công cụ hỗ trợ và một cơ chế gọn nhẹ trên nền Java để các Java-Object tương tác với nhau qua mạng. Tất cả đều được tích hợp trong bộ JDK
- CORBA là một đặc tả kiến trúc tích hợp các đối tượng phân tán trên mạng. Các nhà sản xuất (CORBA) phải cung cấp công cụ, thư viện lập trình khác nhau tương ứng với các nền tảng ngôn ngữ khác nhau
- Sun cũng cung cấp cài đặt Java cho CORBA trong bộ JDK
- CORBA không tương thích với RMI

Kiến trúc CORBA



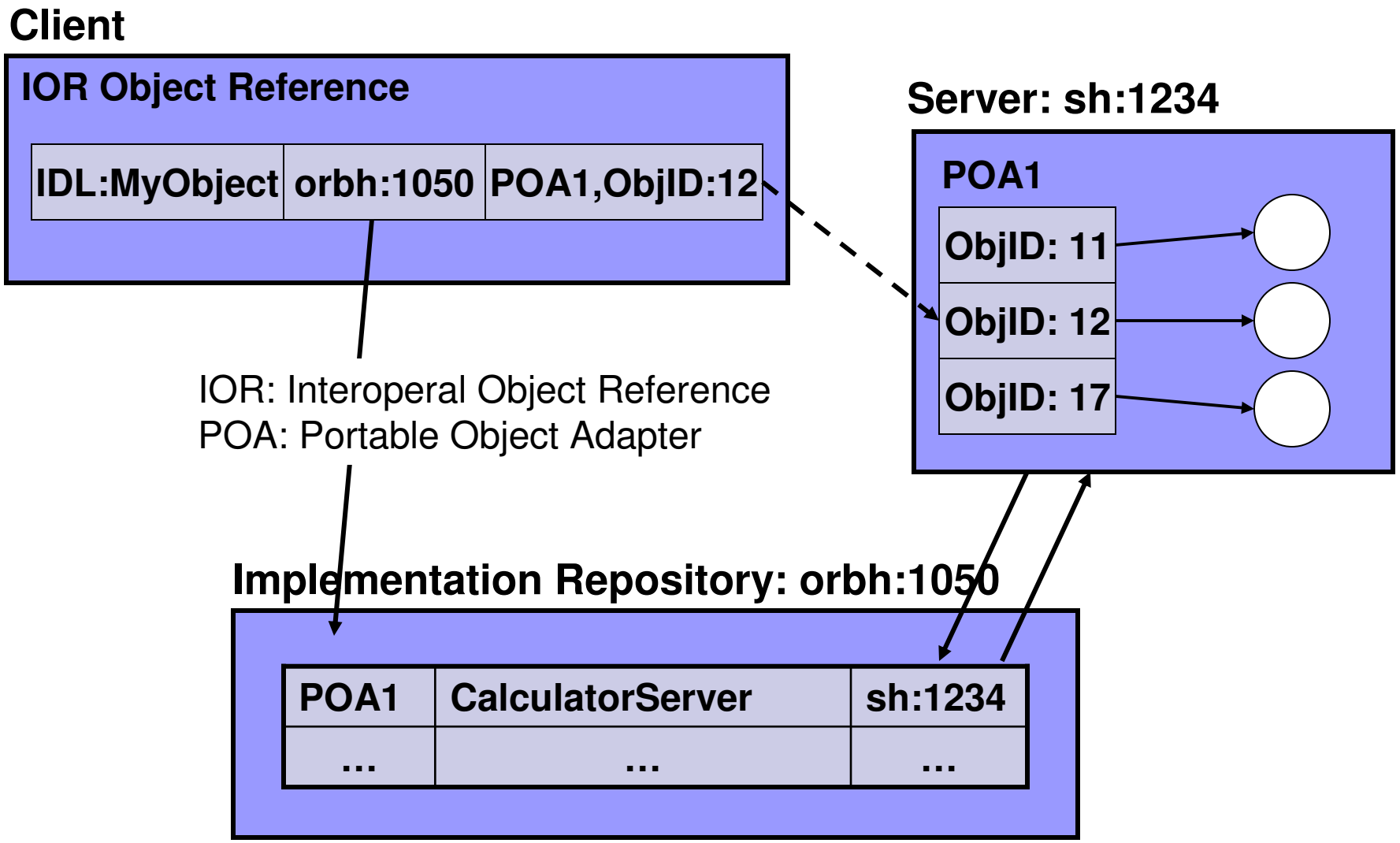
GIOP: General Inter-ORB Protocol
IIOP: Internet Inter-ORB Protocol

DII: Dynamic Invocation Interface
DSI: Dynamic Skeleton Interface

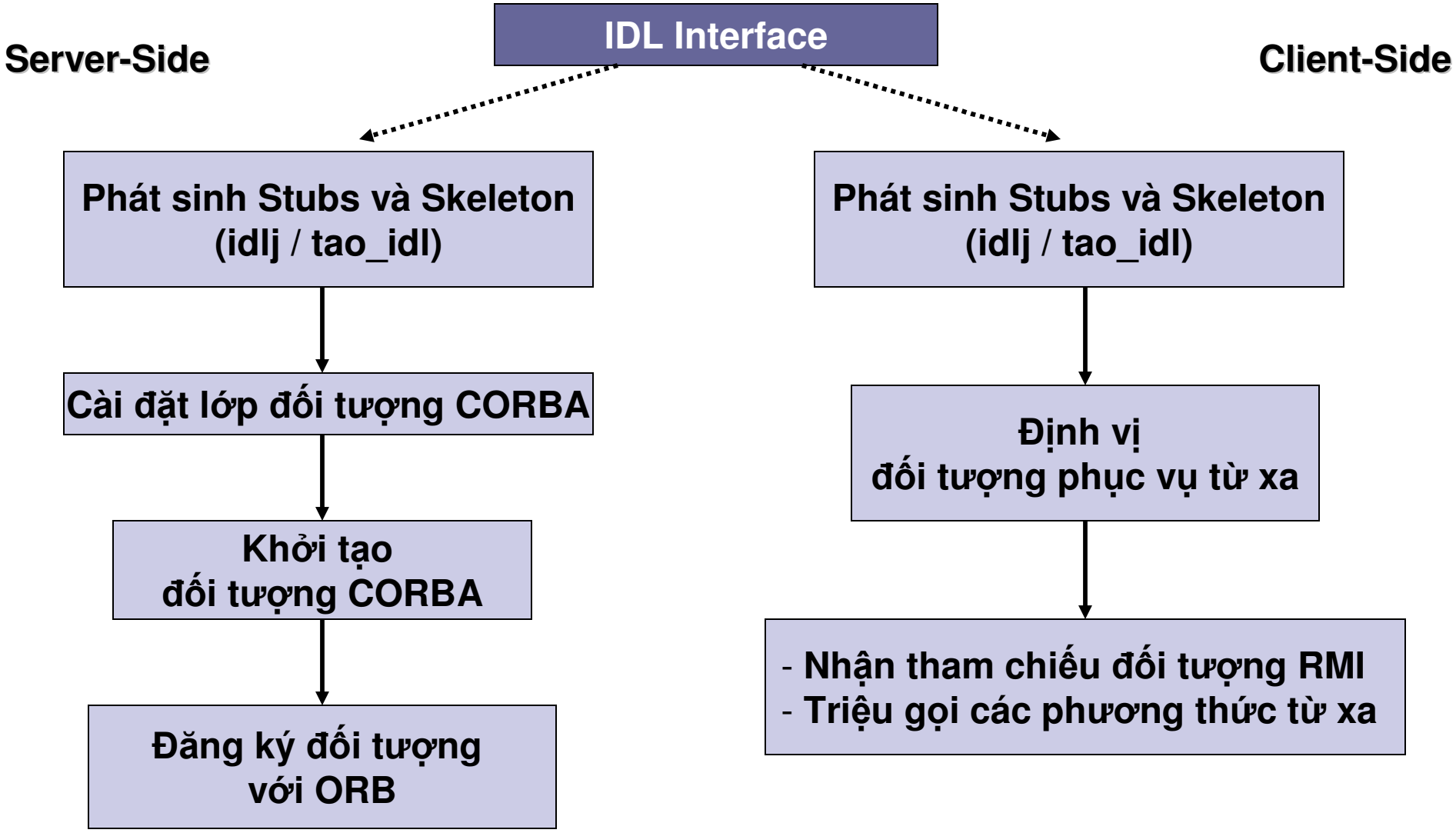
Kiến trúc CORBA

- Các đối tượng CORBA trên các hệ thống khác nhau được triệu gọi thông qua các ORB
- ORB hoạt động dựa vào IDL; dựa trên IDL, ORB biết được cách thức gọi các phương thức của đối tượng: tên phương thức, tham số, giá trị trả về.
- ORB hoàn toàn trong suốt đối với người sử dụng (lập trình viên)

Tham chiếu đối tượng



Qui trình CORBA



CORBA và Java

- IDL-to-Java compiler
  - idlj.exe
- Implementation Reposistory
- Naming Service
  - orbd.exe (default port: 900)
  - java com.sun.corba.se.impl.activation.ORBD
- IDL Server support tool
  - servertool.exe

Data Type Mapping to Java

IDL	Java
`module`	`package`
`interface`	`interface`
`string, wstring`	`java.lang.String`
`void`	`void`
`boolean`	`boolean`
`char, wchar`	`char`
`octet`	`byte`
`short / unsigned short`	`short`
`long / unsigned long`	`int`
`long long / unsigned long long`	`long`
`float`	`float`
`double`	`double`
`fixed`	`java.math.BigDecimal`

- Tham khảo "*IDL to Java™ Language Mapping Specification*"

Ứng dụng CORBA

- Cài đặt một ứng dụng thực hiện các phép toán cơ bản (cộng, trừ, nhân, chia) theo kiến trúc CORBA
- Các thành phần:
  - IDL interface
  - Đối tượng phục vụ CORBA
  - Chương trình Server
  - Chương trình Client

IDL Interface

- Khai báo CORBA IDL module
- Khai báo Interface
- Khai báo các phương thức

Ví dụ

- **Calculations.idl**

```
module Calculations {
   interface iCalculator {
       long long add(in long x, in long y);
       long long sub(in long x, in long y);
       long long mul(in long x, in long y);
       long long div(in long x, in long y);
   };
};
```

Biên dịch IDL

- Thực thi chương trình tiện ích *idlj.exe*

```
idlj -fall Calculations.idl
```

- Sau khi biên dịch IDL, ta có:
  - *iCalculatorPOA.java*: skeleton của đối tượng CORBA
  - *iCalculator.java*: Java-interface được sử dụng trong các chương trình Client và Server
  - *iCalculatorOperations.java*: chứa khai báo cho các phương thức đã được đặc tả trong IDL interface
  - *iCalculatorHelper.java*, *iCalculatorHolder.java*: các lớp tiện ích sử dụng trong ứng dụng CORBA
  - *_iCalculatorStub.java*: stub, client sử dụng để giao tiếp với server-side skeleton

Cài đặt lớp đối tượng CORBA

- Lớp đối tượng phải kế thừa lớp *iCalculatorPOA* vừa được tạo ra ở trên.
- Cài đặt tất cả các phương thức có trong interface *iCalculatorOperations*

Ví dụ

- CalculatorImpl.java

```
import calculations.iCalculatorPOA;

public class CalculatorImpl
                    extends iCalculatorPOA {
  private ORB orb;

  public long add(int x, int x) {
      return x + y;
  }
  ...
}
```

Chương trình Server

- Khởi tạo đối tượng ORB
- Định vị tham chiếu đến đối tượng rootPOA và kích hoạt POAManager
- Khởi tạo đối tượng phục vụ CORBA và chuẩn hóa kiểu đối tượng
- Định vị *naming service* để đăng ký đối tượng CORBA
- Đăng ký đối tượng CORBA với *naming service*
- Chờ phục vụ các yêu cầu CORBA từ client

Ví dụ [1]

- Server.java

```
ORB orb = ORB.init(args, null);

POA rootpoa = POAHelper.narrow(
      orb.resolve_initial_references("RootPOA"));
rootpoa.the_POAManager().activate();

CalculatorImpl calImpl = new CalculatorImpl();

org.omg.CORBA.Object ref =
      rootpoa.servant_to_reference(calImpl);
iCalculator href = iCalculatorHelper.narrow(ref);
```

Ví dụ [2]

```
org.omg.CORBA.Object objRef =
      orb.resolve_initial_references(
                                        “NameService");
NamingContextExt ncRef =
      NamingContextExtHelper.narrow(objRef);

String name = "Calculator";
NameComponent path[] = ncRef.to_name( name );
ncRef.rebind(path, href);

System.out.println(“CORBA Object ready.");

orb.run();
```

Chương trình Client

- Khởi tạo đối tượng ORB
- Định vị *naming service* để tìm kiếm đối tượng CORBA
- Tìm đối tượng CORBA
- Triệu gọi phương thức trên đối tượng CORBA tìm được

Ví dụ

- Client.java

```
ORB orb = ORB.init(args, null);

org.omg.CORBA.Object objRef =
      orb.resolve_initial_references(
                                        "NameService");

NamingContextExt ncRef =
      NamingContextExtHelper.narrow(objRef);

String name = "Calculator";
iCalculator calImpl = iCalculatorHelper.narrow(
                          ncRef.resolve_str(name));

System.out.println(calImpl.add(1, 7));
```

Thực thi ứng dụng

- Khởi động ORB

```
orbd
    [-ORBInitialPort port]
    [-ORBInitialHost host]
```

- Thực thi chương trình server

```
java <server-class>
    [-ORBInitialPort port]
    [-ORBInitialHost host]
```

- Thực thi chương trình client

```
java <client-class>
    [-ORBInitialPort port]
    [-ORBInitialHost host]
```

Đọc thêm

- CORBA services & Java API
- DII & DSI
- CORBA và C/C++
  - ORBacus: C++ and Java ORB
    URL: http://www.orbacus.com
  - omniORB: C++ and Python ORB
    URL: http://omniorb.sourceforge.net
  - Orbix: Enterprise CORBA ORB
    http://www.iona.com/products/orbix/welcome.htm
  - VisiBroker: Enterprise CORBA ORB
    URL: http://www.borland.com/us/products/visibroker/index.html
  - The ACE ORB (TAO): C++ ORB [Open Source]
    URL: http://www.theaceorb.com

Q&A