面向对象的分布计算

- 分布式对象（面向对象编程：抽象、封装、继承、多态）

  一个对象由两部分组成：状态（属性） + 行为（方法）

  分布式对象是一些独立代码的封装体（可以说是对象，也可以是说组件）。它**向外**提供了一个包括一组属性和方法的**接口**，远程客户程序通过**远程方法**调用来访问它。

  特点:

  > 位置、实现的透明性
  > 语言、平台独立
  > 通过预先定义好的接口进行访问
  > 相互发送消息实现请求服务
  > 客户与服务器的角色划分是相对和或多层次的
  > 分布式对象具有可在网络上移动的动态性

- 远程对象方法调用RMI

  1. 远程对象引用(Remote Object Reference)

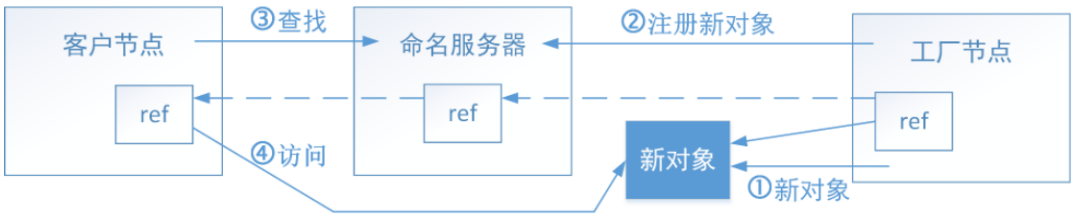
     访问**远程对象**的一种机制

     这个对象引用可以用**{网络地址，端口号，内部ID}**三元组，唯一标识

     *网络地址：对象实际驻留节点的网络地址

     *端口号：管理该对象的服务器进程所在的端口号

     *内部ID：服务器为该对象提供的内部ID

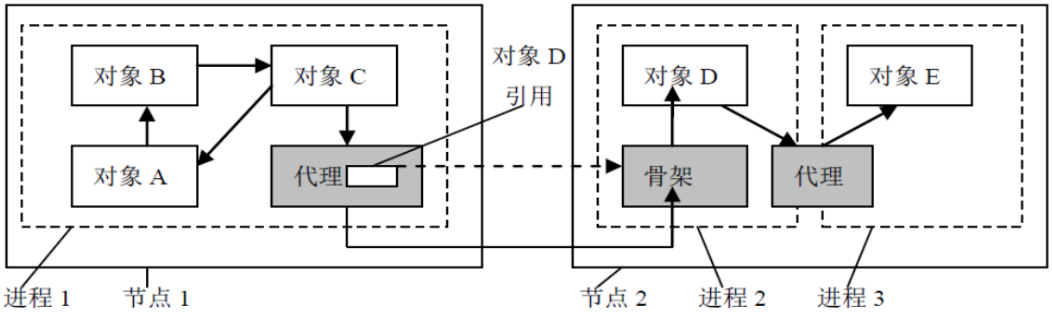
     

     这里的虚线不是很明白

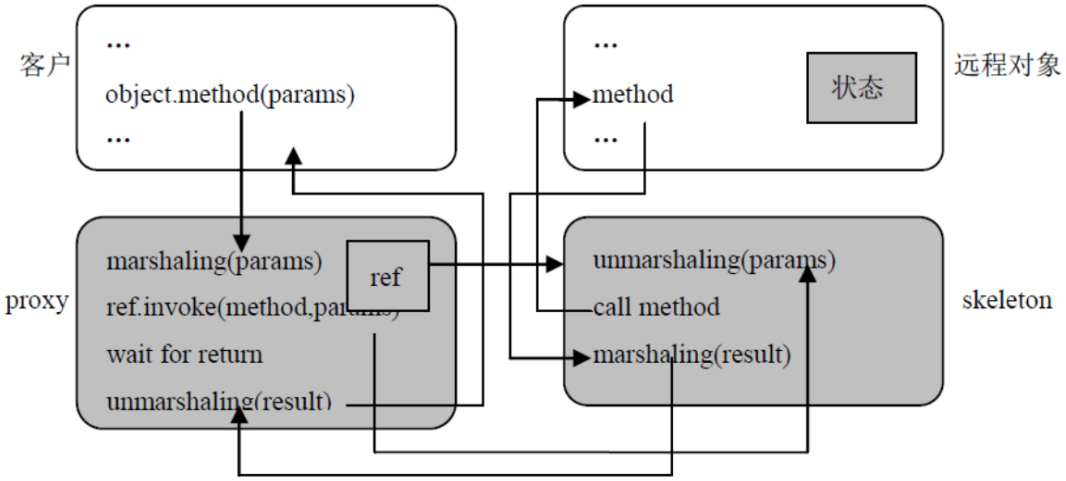
  2. 对象通信

     > 调用和被调对象在同一个进程中，称为**本地调用**；
     > 调用和被调对象在同一节点但不在同一进程中，称为**进程外调用**；
     > 调用和被调对象在不同的节点上，称为**远程调用**。

     那我们的实验不是只能算是进程外调用？？

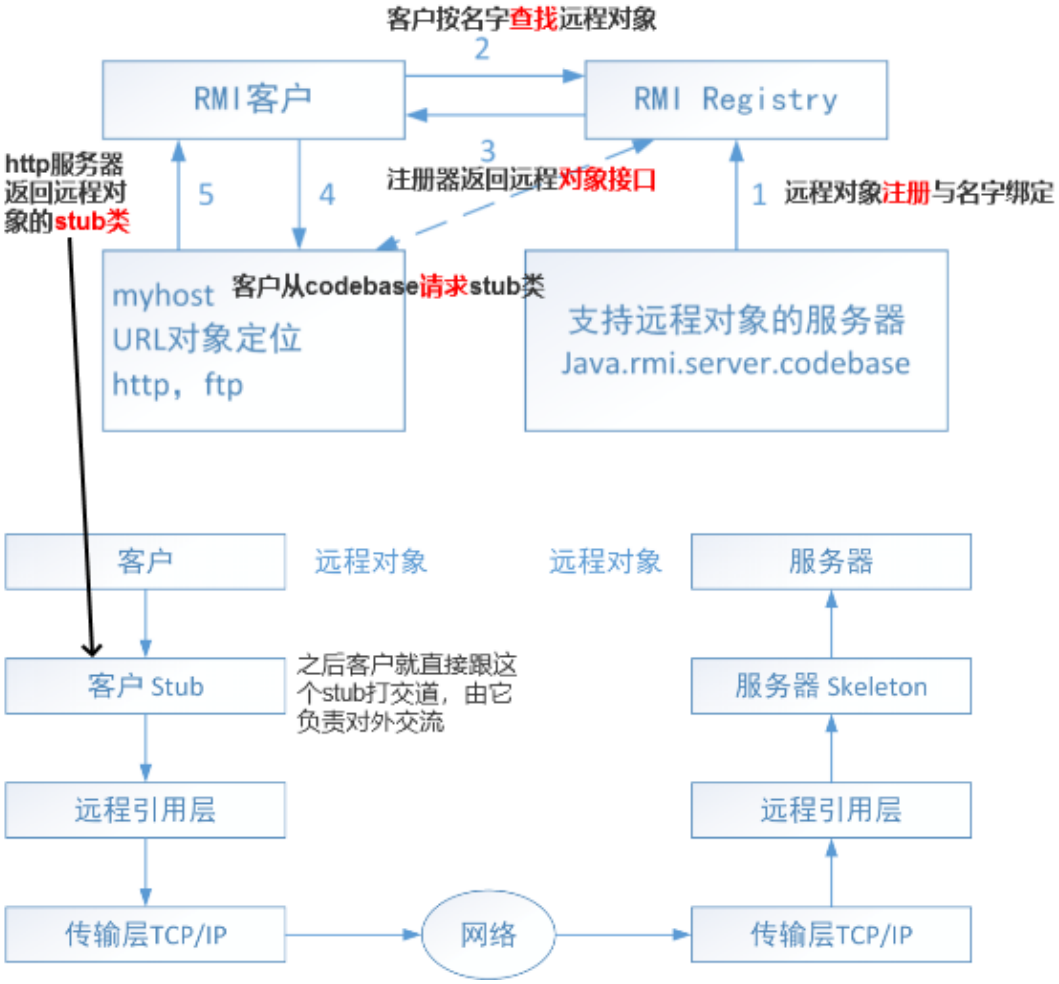
  3. 方法调用执行

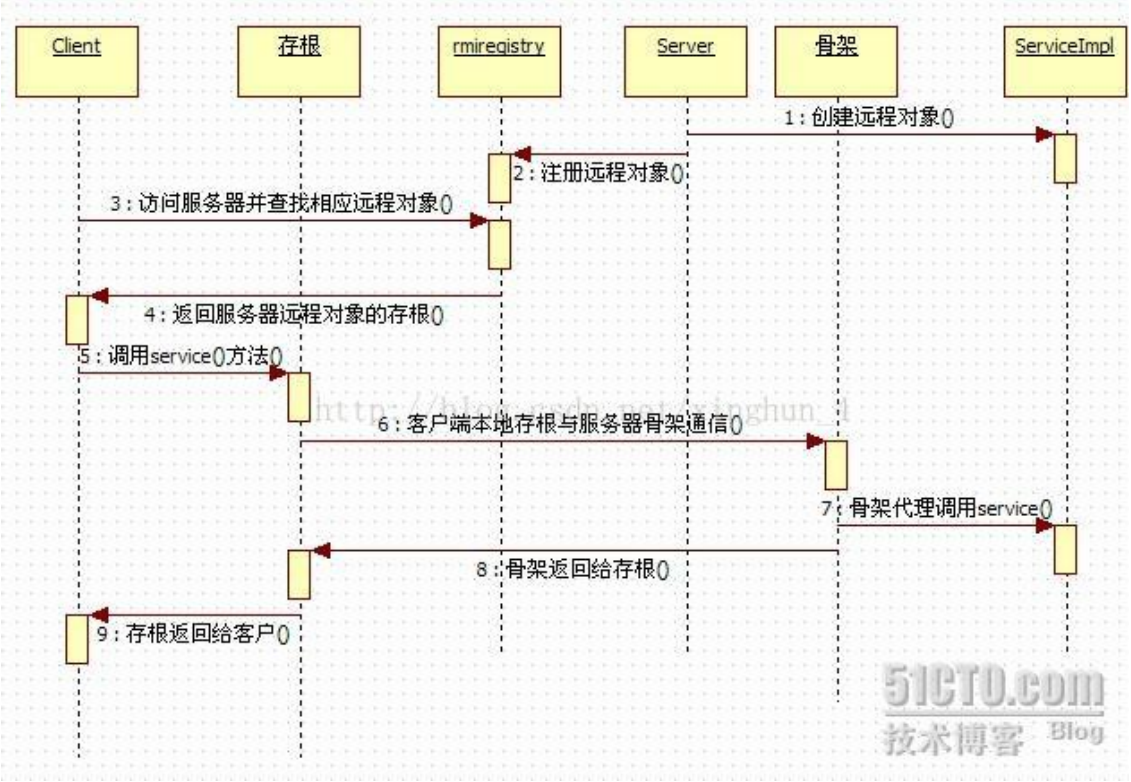


其中marshaling主要是一个封装作用，因为需要在网络中传输嘛，所以必须要进行封装。能实现封装的话，在java中这个类必须implements serializable。

4. 远程方法调用语义
    - 可能语义：调用者不知道执行过没
    - 至少一次语义：
    - 至多一次语义：

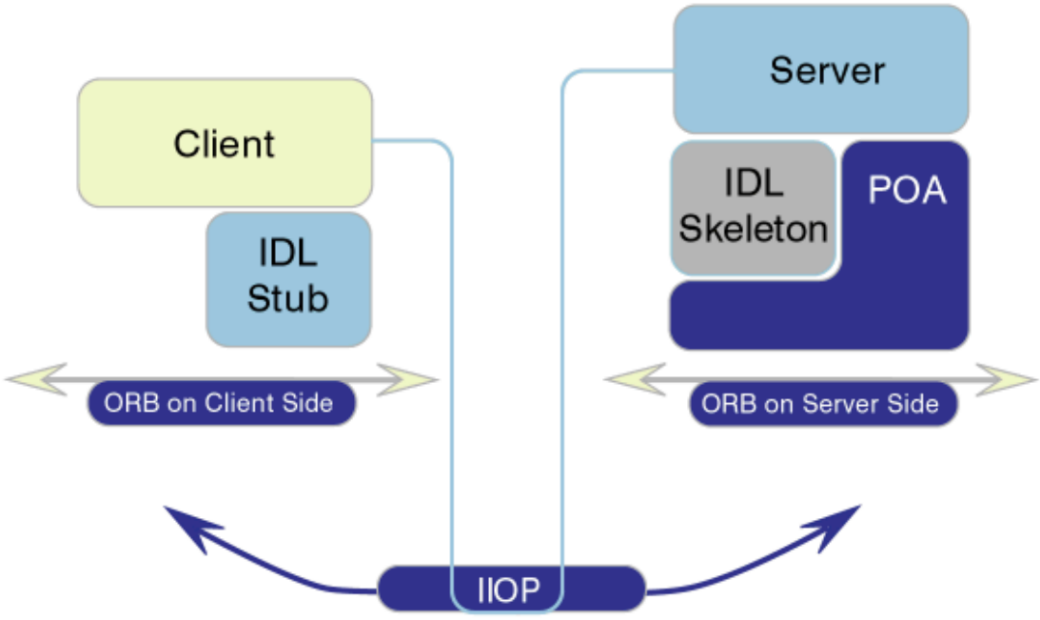
- Java RMI





- 公共对象请求代理结构CORBA

> CORBA的核心是对象请求代理 (ORB，Object Request Broker)，它提供了网络环境无关性、操作系统无关性和开发语言无关性的公共平台。
> 在面向对象的应用环境中，CORBA对象的请求者不必知道它所请求的对象是在哪里，是如何实现的，而是由ORB来负责跨平台的运作管理，无须应用系统的开发者干预。
> 具有的跨平台、分布式、面向对象等优点。
> CORBA是一个中间件规范并不是一个实体软件。软件开发者通过使用第三方的ORB工具或IDL语言来定义CORBA对象，实现ORB功能。



ORB：**对象请求代理**，作为"软件总线"来连接网络上的不同对象。提供对象的**定位**和**方法**调用。（类似于JAVA rmi中的远程引用层）

OA：**对象适配器**，用于构造对象实现与ORB之间的接口。给框架发送方法调用。支持服务器对象的生命周期(对象的创建和删除)。

> IDL
>
> - 数据类型

Java**数据类型**	IDL**数据类型**
void	void
boolean	boolean
char	char
byte	byte
short	short
int	long
Long	long long
Float	Float
Double	Double
Java.lang.String	String / wstring

其他的基本都是一样的，只有这三个有所区别，加强记忆。

- 接口

```
interface UserAccount {
    float getBalance();
    void setBalance(in float amount);
};
```

这个跟java也很类似，开头由关键字interface描述，不能在接口中定义变量。

CORBA支持接口的继承，例如：Interface Book： Product{ /*... */};
可以用冒号（：）表示继承性，一个接口可以**继承多个接口**的属性。

P46？？不是说接口中不定义变量嘛？P49咋又有变量了？

- 模块（module）

对应于java中的package的概念，把逻辑上相关接口组合在一起的一种便捷方式。

```
Module AccountTypes     {
        interface UserAccount{…};
        interface UserID{…};
        interface SubScription{…};
    }
```

- 属性

IDL中属性的定义：

```
attribute string isbn;
```

这**一个属性**在java中等价为**两个方法**，访问器和变换器：

```
String isbn(){} //accessor,means getter
Void isbn(String _isbn){}   //mutator,means setter
```

- 操作

  可以远程调用这些操作，执行你所能想到的简单的或者复杂的任务。
  操作是一个**需要接受参数**并在结束时有**返回值**（可以void）的函数。

  三种参数：

  - in： 只能用于输入的参数，是不可变的。
  - out： 可修改值的参数，是可变的。out怎么体现?
  - inout： 参数既可以被用作输入，又可以被修改。

```
interface UserBalance {
float getBalance();
void setBalance();
float addBalnace(in float amount);
float subtractBalnace(in float amount);
};
```

  IDL不支持方法的重载，必须为每一个方法配备一个不同的名字。

- 异常处理

```
module BankingSystem {
    exception AccountInactive {
    String reason;
    };
    exception AccountOverdrawn {
        string reason;
    };
    interface BankAccount {
        double withdrawMoney(in double amout) raises
(AccountInactive， AccountOverdrawn );
        //raises相当于java中的throws
    };
};
```

  在CORBA中，异常不可以为子类，即无法自定义异常。
  可以在一个模式中定义几个异常，并可以在一个操作中抛出多种异常。

- 分布式组件对象模型DCOM