分布式 Ada 语言系统的实现*

张冰 李赣生 王华民

(浙江大学计算机科学与工程系 杭州 310027)

E-mail: ligs@cs.zju.edu.cn

摘要 基于 Ada 95 参考手册附录 E"分布式系统"中的思想,提出了实现分布式 Ada 语言系统的一些概念和设计思想,并给出了实现分区通信子系统的具体方案.在此基础上,通过一些前置处理,并利用已实现的分区通信子系统的接口,实现了分布式 Ada 语言系统.最后通过一个实例,具体介绍了分布式 Ada 语言系统的程序设计方法.

关键词 分区,远程过程调用,分区通信子系统,Stub,打包,解包.

中图法分类号 TP312

Ada 程序设计语言及其相应的运行支撑环境能够用于开发大型软件工程应用项目,这些工程大多涉及嵌入式计算机系统.但是,对于涉及松散耦合方式的分布式群机系统的工程项目,特别是必须通过通信网络连接的嵌入式应用来说,现有的 Ada 语言环境不能提供有效的支持.分布式 Ada 语言系统就是针对这种具体的群机系统环境,使一个完整的 Ada 程序分布到群机系统上并行执行.

在国外,GNU 组织和美国 Colorado 大学计算机系都曾对分布式 Ada 进行开发,但是都无法做到既遵从 Ada 95 规范又使应用开发人员不涉及网络编程.国内一些大学和研究机构在分布式程序设计领域中,对 C 和 C++做的工作较多,在分布式 Ada 方面还没有同类的工作.

本文讲述的分布式 Ada 语言系统遵循 Ada 95 参考手册附录 E"分布式系统"的标准,并且做了大量的前置处理工作,这样对应用开发人员来说,开发分布式应用系统时可以快捷、方便,不用考虑网络通信之间的细节问题.同时,应用开发人员较少地接触底层通信设施,从而提高了整个分布式应用系统的安全性.

1 分布式 Ada 语言系统设计思想

Ada 语言是由美国国防部主持设计的大型公共基础语言.1983 年,形成国际标准 Ada 83.此后,为了将面向对象的特征与方法融合于 Ada,进一步完善 Ada 83 的功能,形成了新的标准 Ada 95.Ada 95 参考手册附录 E 描述了分布式 Ada 语言系统的设计思想[1]:将一个完整的 Ada 程序划分成若干个分区,每个分区都对应于分布式系统的一个节点,通过分区间分工协作来完成整个系统的功能.

分布式 Ada 程序的分区是库单元的集合.

在分布式 Ada 语言系统中,远程过程调用(remote procedure call,简称 RPC)是不可缺少的重要环节,通过远程过程调用,分布式 Ada 语言系统中各个分区将有机地结合起来,共同完成整个系统的任务.

Ada 95 参考手册附录 E 还设计了分区通信子系统(partition communication subsystem)这样一种接口,远程过程调用就是通过调用这个接口中提供的各种函数来完成的.

1.1 分布式 Ada 语言系统中远程过程调用基本框架

在分布式 Ada 语言系统中远程过程调用的基本框架如图 1 所示.

---

* 作者张冰,1974 年生,硕士生,主要研究领域为编译,软件工程,并行处理.李赣生,1940 年生,教授,博士生导师,主要研究领域为软件工程,编译,软件自动化.王华民,女,1944 年生,副教授,主要研究领域为程序设计语言与实现,软件工程.

本文通讯联系人:李赣生,杭州 310027,浙江大学计算机科学与工程系

本文 1998-10-06 收到原稿,1999-04-12 收到修改稿

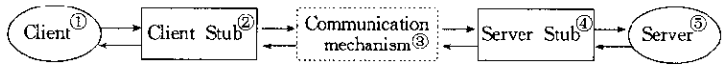


Fig. 1 Basic RPC frame for distributed Ada language system

图1 分布式Ada语言系统的RPC基本框架

在实现RPC的过程中,引入了Stub机制,客户方和服务方都是通过各自的Stub来进行远程过程调用,所谓的Stub就是分区间关于通信的一段程序代码.在分布式Ada语言系统中,Stub是一段对分区通信子系统中接口调用的程序代码.

1.2 分布式Ada语言系统的透明性问题

Andrews S. Tanenbaum曾给出关于分布式系统透明性的5种描述[2]:位置透明性、迁移透明性、副本透明性、并发透明性和并行透明性.

在Ada 95附录E中,并没有对透明性进行详细的描述.但是,透明性是分布式系统的关键特征之一.在分布式Ada语言系统应做到:首先,利用分布式Ada语言系统开发分布式应用程序时应方便、快捷,基本上不必涉及网络编程;其次,以上5种透明性在利用分布式Ada语言系统开发的分布式应用系统中都能实现,即用户在对远程过程进行调用的时候,不必考虑调用发生的位置,不必了解具体的调用过程所在的节点,不必知道访问的对象是否有增加可靠性的副本,同时,多个用户可共享资源,系统中对象的迁移也不受改名的干扰.例如:

Ada的远程过程调用形式为

```
package A is                                    && 在一个分区中
    pragma Remoter_Call_Interface (A);
    procedure P (.....);
    .....
end A;
```

```
with A;                                         && 在另一个分区中
package B is
    .....
    A.P (.....);                                && 一个远程过程调用
    .....
end B.
```

通过上面的程序可以看到,远程过程调用的形式与本地调用(包括数据结构)是一致的.因此对于发生本地过程调用或是远程过程调用,系统应能自动识别,这就要求系统做到位置透明性.在底层的传输实现中,采用了TCP/IP协议,因此,在分布式Ada语言系统的分区设计中,分区与相应机器的IP地址对应,就解决了分布式系统的迁移透明性.其余的透明性问题,可以通过后面的例子来解释.

2 分区通信子系统的实现

分区通信子系统(PCS)为一个分布式程序的分区之间的程序提供了进行通信的一组接口.

2.1 打包和解包过程的实现

针对打包和解包,PCS中提供了两个函数:

```
procedure write(stream:in out string; item:in implementation-defined);
procedure read(stream:in out string; item:out implementation-defined; last:out integer).
```

过程write提供打包的功能,其中参数stream为打包后的参数流,作为远程过程调用参数的item可以是整型、字符型、浮点型、字符串型、枚举型、记录类型等类型之一.设计过程中利用Ada语言中提供的重载机制,将write和read过程中的参数Item类型多样化.用户调用时,对于write不必考虑所要写入流中的是何种类型,对于read不必考虑从流中读出下一个类型的转化问题.

在分区通信子系统中,将参数流设计为字符串的形式,这种形式有两个优点:(1) 字符串的操作及字符串与其他类型之间的相互转换便于实现,系统也提供了一部分函数.(2) 在网络传递参数流的底层,由Linux下的SUN RPC软件包实现了一些基于C语言的RPC的函数,这些函数都是建立在字符串类型的基础上的,而Ada

与C的字符串之间没有相互转化的问题.

在对客户方的输入流设计中,规定一种特殊符号为信息之间的分隔符;流的前端必须加入特定字符,用来识别同步或异步调用,其后是要调用的过程名及其参数.在对服务方的返回流设计中,规定流的前端加入特定字符,用来区别调用是否成功.

2.2 传输过程的实现

利用Linux下的SUN RPC软件包,可以进行基于C语言的RPC程序设计.这样的一个RPC的实现,最基本的要求是:在服务方和客户方要求登记双方约定的远程过程的程序号、程序版本号和过程号,提供输入参数和输出参数的类型以及网络的类型;利用外部数据表示标准(XDR)来进行参数传递.这个软件包提供的函数也有不全面的地方,例如,无法进行多参数传递,处理复杂的数据结构时应用开发人员除了事先需要做大量的工作——利用XDR语言编写复杂的数据结构外,还要参与网络编程,同时操作上也很繁琐.因此,纯粹依赖Linux下的SUN RPC软件包是无法完成分布式Ada语言系统的要求的.

为此,在设计中先实现了一个传递字符串的RPC过程c_stub,这个过程构成了传输过程的主体,它可以完成同步字符串远程传递和异步字符串远程传递,前者发送一个字符串并等待接收一个字符串,后者发送一个字符串后立即返回.

PCS在客户方提供了以下过程:

procedure do_rpc(partition:in partition_id; params:in string; result:out string);

procedure do_apc(partition:in partition_id; params:in string).

过程do_rpc提供了同步远程过程调用的功能.它的3个参数分别为分区号、输入参数流和输出参数流.过程do_apc提供了异步远程过程调用的功能.它的两个参数分别为分区号和输入参数流.

分区号是指提供远程服务的节点名称.当调用发生后,do_rpc把分区号、相关参数打包后传递给过程c_stub,由c_stub向分区号规定的服务方发送输入参数流.当c_stub接收到返回参数流后,再将此参数流返回给do_rpc来处理解包工作.

PCS在服务方提供了过程:

procedure Establish_RPC_Receiver.

过程Establish_RPC_Receiver是个无参数的过程,它的主要作用是初始化网络,接收和发送参数流.这个过程,只有在服务方前置处理生成的另一个Ada程序的配合下,才能完成服务方的stub工作.

3 前置处理的有关工作

在客户方,前置处理的目的是为了使应用开发人员在使用本系统时不必考虑网络编程的细节.

前置处理所做的工作是对标记了pragma(Remote_Call_Interface)的程序包规格说明进行处理,生成一个相同名字的程序包体.在用户发出类似本地调用的远程调用时,即激发了此包体中相同名字的过程,此过程通过调用PCS中的接口完成了打包、发送、接收、解包等stub的任务.

例如,某远程程序包规格说明如下:

```
package testfunc is
    pragma Remote_Call_Interface(testfunc);    && 表示 testfunc 是一个远程程序包
    function testfunc1(i:in integer; j:in integer) return integer;
    ...
end testfunc.
```

上面这个程序包规格说明经前置处理后自动生成的程序包体如下:

```
with ada_rpc;use ada_rpc;
package body testfunc is
    function testfunc1(i:in integer;j:in integer) return integer is
            inparams,result:string(1..implementation_defined);  && 可根据用户要求定义长度
            resultparams:integer;                               && 返回的参数
            last:integer;                                       && 指示当前参数流中的位置
```

```
begin
        write(inparams,"testfunc1");                    && 打包过程
        write(inparams,i);                              && 打包过程
        write(inparams,j);                              && 打包过程
        do_rpc(partition_id_1,inparams,result);         && 分区号在程序包初始化中定义
        last:=1;
        read(result,resultparams,last);                 && 解包过程
        return(resultparams);
end testfunc1;
...
end testfunc.
```

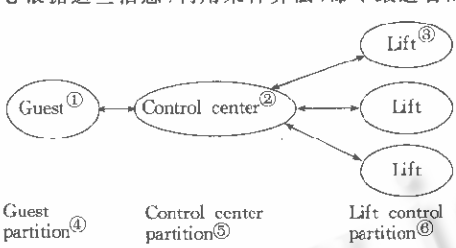
可以看到,用户在调用远程过程 testfunc.testfunc1时却激发了本地新创建 testfunc 程序包中的函数 testfunc1,这个 testfunc1函数通过调用 PCS 的接口完成了客户方 stub 的所有任务.

在服务方,前置处理也对标记了 pragma(remote_call_interface)的程序包规格说明进行了处理,生成一个 Ada 程序 ada_call_func,它可以完成打包、解包、信息分析和相关过程调用的工作.当服务方启动了过程 Ada_rpc.Establish_RPC_Receiver,接收到客户方的参数流后,将控制权交给 ada_call_func 进程,并把参数流传递给 ada_call_func,由 ada_call_func 进程对参数流进行解包,并根据解包后的参数执行相关过程,随后将应回传的参数打包并交给 Establish_RPC_Receiver,由 Establish_RPC_Receiver 将参数流回传给客户方.至此,完成了服务方的 Stub 工作.下面将通过一个分布式问题的解决方法,介绍如何利用 Ada_rpc 程序包和一些相关的前置处理来设计一个分布式 Ada 应用系统.

4 一个分布式问题:模拟电梯控制

这个例子取自文献[3],但本文对这个问题的处理与其不同.

设电梯的运行控制模式如下:当客人要求电梯服务时,控制中心收到客人所在楼层及要求的方向等信息.控制中心向每个电梯发出查询信号,并接收到每个电梯返回的当前所在楼层及运行情况(向上、向下或静止),控制中心根据这些信息,利用某种算法,命令最适合的电梯在客人要求的楼层进行服务.



①客人,②控制中心,③电梯,④客人分区,⑤控制中心分区,⑥电梯分区.

Fig. 2 Partitions for simulating lift control
图2 模拟电梯控制的分区划分

根据分布式 Ada 程序设计思想,将这样的一个电梯控制程序划分为3个分区(如图2所示):控制中心分区、电梯分区和客人分区,并编写了相关的分布式 Ada 程序.

这3个分区分别分布在3台微机上,完成的功能是:客人分区向控制中心分区发送所在楼层和要上或要下的请求,控制中心接到请求后立即向电梯分区发出查询信息,当每个电梯的所在楼层和运行情况返回控制中心分区后,控制中心分区将结合客人的情况,指示最合适的电梯到达客人所在的楼层,并向客人分区返回为其服务的电梯的序号.

作为服务方的电梯分区提供的主要程序(这里省略了程序包的具体实现)有:

```
filename:liftcontrol.ads                    && 电梯分区提供远程程序包的规格说明
package liftcontrol is
    pragma Remote_Call_Interface;
    procedure quest_lift(liftnum:in integer;direction:out integer;layer:out in teger);
    && 接受控制中心的查询,返回电梯的运动方向和所在楼层
    function control_lift(liftnum:in integer; layer:in integer) return integer;
    && 接受控制中心的指令,命令指定的电梯前往指定的楼层,返回值表示指令是否有效
```

```
end liftcontrol.
作为既是服务方又是客户方的控制中心分区提供的主要程序有:
filename:controlfunc.ads                    && 这是控制中心提供的远程程序包的规格说明
package controlfunc is
    pragma Remote_Call_Interface;
    function requestfunc(layer:in integer;updordown:in integer) return integer;
    && 接受客户的请求后,对电梯分区进行查询,并通过一个最佳电梯算法来决定某电梯进行服务,
    && 然后发出相关指令,最后返回给客户将要进行服务的电梯号.
end controlfunc.
```

以上是应用开发人员应写的程序(作为客户方的客人分区的程序较为简单,从略).在分布式 Ada 语言系统中所有的前置工作都由预处理器 PP 来完成,同时 PP 还根据应用开发人员编写的系统配置来完成分区工作.在这个例子中,PP 分别对 controlfunc.ads 和 liftcontrol.ads 两个文件进行处理,在每个分区都生成一个具有 stub 功能的程序,并在每个提供服务的分区都生成一个侦听程序,用于侦听客户方发来的请求.这些前置处理生成的程序和用户编写的程序经编译生成可执行文件后,再分布到各自的节点上即可运行.

从上面这个例子看,应用开发人员设计的分布式程序和普通的程序几乎没有区别,不必考虑远程过程的位置和发生调用的具体细节.由于底层实现的传输实体是基于 Linux 下的 SUN RPC 完成的,而 SUN RPC 支持并发和并行的透明性,因而利用本系统开发的应用系统也支持并发和并行的透明性,在这个例子中,如果存在多个客人分区,它们之间将互不干扰,可以并行运行.对于副本透明性,由应用开发人员决定是否需要增加副本.

5 结束语

Ada 语言是大型软件工程语言.同时,在嵌入式系统中得到广泛的应用.但是,嵌入式系统大都是分布式系统,如果应用系统的程序是用没有分布式特征的 Ada 语言书写的,为了实现分布,应用开发人员就不得不自己进行大量的网络编程来实现远程过程调用和不同节点上的程序之间的通信.这些工作应用开发人员都是可以做到的,但是在这些细节问题上必须花费大量的时间.使用本系统提供的功能之后,分布式的程序设计就与普通的程序设计没有什么差别了,可以忽略网络编程和远程通信与调用的大量细节.这就是本项目的主要目的.

参考文献

1 ISO/IEC 9652: 1995(E). Ada reference manual——language and standard libraries. 1995
2 Tanenbaum A S. Distributed operating system. Englewood Cliffs, NJ: Prentice Hall, Inc., 1995
3 Wellings A J. Issues in distributed processing-session summary. In: Proceedings of the 1st International Workshop on Real. Time Ada Issues. ACM Ada Letters, 1987,7(6):57~60

Implementation for Distributed Ada Language System

ZHANG Bing LI Gan-sheng WANG Hua-min

(*Department of Computer Science and Engineering Zhejiang University Hangzhou* 310027)

Abstract According to Ada 95 reference manual(E), some concepts and design ideas implemented in distributed Ada Language system are introduced in the paper, and the detailed implementation of the partition communication subsystem is given. With the interfaces of partition communication subsystem and some pre-processings, the remote procedure calls in distributed Ada language system are implemented. Finally, an example is given.

Key words Partition, remote procedure call, partition communication subsystem, Stub, marshelling, unmarshelling.