〔19〕中华人民共和国专利局

〔11〕公开号 CN 1065942A

〔12〕发明专利申请公开说明书

〔21〕申请号 92101822.3

〔51〕Int.Cl[5]

G06F 9/06

〔43〕公开日 1992年11月4日

〔22〕申请日 92.3.18

〔30〕优先权

〔32〕91.4.17 〔33〕IL 〔31〕097894

〔71〕申请人 国际商业机器公司

地址 美国纽约

〔72〕发明人 尼塔·雅各布·阿米特

约翰·米歇尔·马伯格

尤瑞·沙尼

〔74〕专利代理机构 中国国际贸易促进委员会专利代理部

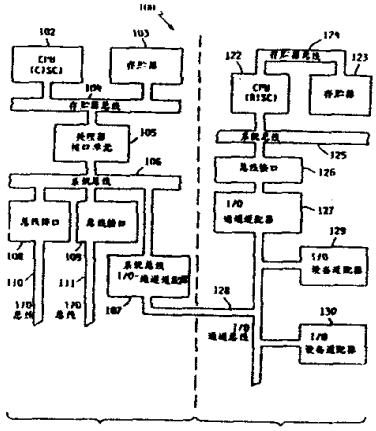
代理人 乔晓东

说明书页数: 24 附图页数: 9

〔54〕发明名称 分布式程序栈装置和方法

〔57〕摘要

一个多处理器计算机系统执行一个在其程序模块里具有多个可调用过程的单线程序。每个处理器的局部存贮器包含一个程序栈、在该处理器上执行的每个模块的目标代码、以及一个包含连接信息的代理目标和数据结构。另外，此局部存贮器还包括在另一处理器上可执行的每个过程的C—备忘模块，以及可被在另一处理器上执行的过程调用的此局部存贮器内每个过程的S—备忘模块。



权　利　要　求　书

1. 一个执行具有多个可调用过程的程序的分布式处理装置，其特征为包括：

连到第一局部存贮器上的第一处理器，用来执行含于所述第一局部存贮器内的过程；

连到第二局部存贮器上的第二处理器，用来执行含于所述第二局部存贮器内的过程；

用于所述第一和第二处理器之间数据通讯的装置；

用于包含在所述第一局部存贮器内所述程序的第一可调用过程调用包含在所述第二局部存贮器内所述程序的第二可调用过程的装置；

用于所述第二可调用过程调用一个含于所述第一局部存贮器内以所述第一可调用过程的名义执行的可调用过程的装置。

2. 权利要求 1 的分布式处理装置，其特征为：用于所述第二可调用过程调用含于所述第一局部存贮器内的可调用过程的装置包括用于所述第二可调用过程递归调用所述第一可调用过程的装置。

3. 权利要求 1 的分布式处理装置，其特征为，包含在所述第一局部存贮器内的第一程序栈，所述第一程序栈包括一个或多个启

动单元，每个启动单元包含所述第一局部存贮器内一过程的即时状态信息；

包含在所述第二局部存贮器内的第二程序栈，所述第二程序栈包括一个或多个启动单元，每个启动单元包含所述第二局部存贮器内一过程的即时状态信息。

4. 权利要求 1 的分布式处理装置，其进一步的特征为包括：

用于调用所述程序的外层过程以在所述第一处理器内开始执行所述程序的装置；

用于自动在所述第一过程和所述第二过程之间建立联系的装置，其中用于所述第一可调用过程调用所述第二可调用过程的装置是自动启动的；

用于自动在所述第二过程和一个包含在所述第一局部存贮器内的可调用过程之间建立联系的装置，其中用于所述第二可调用过程调用包含在所述第一局部存贮器内以所述第一可调用过程名义执行的可调用过程的装置是自动启动的。

5. 权利要求 1 的分布式处理装置，其特征为：

包含在所述第一局部存贮器内并代表所述第二可调用过程的第一 C—备忘模块；

包含在所述第二局部存贮器内并代表所述第一可调用过程的第一 S—备忘模块；

包含在所述第二局部存贮器内并代表包含在所述第一局部存

贮器内所述过程的第二C—备忘模块，所述第一局部存贮器内的所述过程在以所述第一可调用过程名义执行时可被所述第二可调用过程调用；

包含在所述第一局部存贮器内并代表所述第二可调用过程的第二S—备忘模块；

用于所述第一可调用过程向所述第一C—备忘模块发布第一局部调用命令的装置；

响应所述第一局部调用命令、用于所述第一C—备忘模块向所述第一S—备忘模块传送包含在所述第一局部调用命令中的数据的装置；

响应用于第一C—备忘模块向所述第一S—备忘模块传送包含在第一局部调用命令中数据的装置，用于所述第一S—备忘模块向代表所述第一可调用过程的所述第二可调用过程发布第二局部调用命令的装置；

用于所述第二可调用过程在所述第二可调用过程以所述第一可调用过程的名义执行时向所述第二C—备忘模块发布第三局部调用命令的装置；

响应所述第三局部调用命令，用于所述第二C—备忘模块向所述第二S—备忘模块发送包含在所述第三局部调用命令中的数据的装置；

响应用于所述第二C—备忘模块向所述第二S—备忘模块发

送包含在所述第三局部调用命令中数据的装置、用于所述第二S—备忘模块向包含在所述第一局部存贮器内的所述过程发布第四局部调用命令的装置，所述过程可被以所述第一可调用过程名义执行的所述第二可调用过程调用，其中，所述第四局部调用命令以所述第二可调用过程的名义发布。

6. 权利要求5的分布式处理装置，其特征为：用于所述第二可调用过程调用包含在所述第一局部存贮器的过程的所述装置包括用于所述第二可调用过程递归调用所述第一可调用过程的装置。

7. 权利要求5的分布式处理装置，其特征为：

包含在所述第一局部存贮器内的第一程序栈，所述第一程序栈包括一个或多个启动单元，其中包含在所述第一局部存贮器内的每例过程，包含在所述第一局部存贮器内的每例C—备忘模块，以及包含在所述第一局部存贮器内的每例S—备忘模块由包含所述例子状态信息的所述第一程序栈的单个启动单元代表；

包含在所述第二局部存贮器内的第二程序栈，所述第二程序栈包括一个或多个启动单元，其中包含在所述第二局部存贮器内的每例过程，包含在所述第二局部存贮器内的每例C—备忘模块，以及包含在所述第二局部存贮器内的每例S—备忘模块由包含所述例子状态信息的所述第二程序栈的单个启动单元代表。

8. 权利要求5的分布式处理装置，其特征为：

用于调用所述程序的外层过程以开始在所述第一处理器上执

行所述程序的装置；

用于自动建立所述第一过程和所述第二过程间的联系的装置，其中用于所述第一可调用过程调用所述第二可调用过程的装置自动地被启动；

用于自动建立所述第二过程和包含于所述第一局部存贮器内的可调用过程之间的联系的装置，其中用于所述第二可调用过程调用包含在所述第一局部存贮器内以所述第一可调用过程名义执行的可调用过程的装置自动地被启动。

9. 一个用于在多处理器系统上执行计算机程序的方法，其特征为以下步骤：

把包含在所述程序内的第一可调用过程集合分配到第一处理器上；

把包含在所述程序内的第二可调用过程集合分配到第二处理器上；

用所述第一处理器执行包含在所述第一可调用过程集合内的第一可调用过程；

在用所述第一处理器完成所述执行第一可调用过程的步骤时，从所述第一可调用过程调用包含在所述第二可调用过程集合中的一个可调用过程；

用所述第二处理器以所述第一可调用过程的名义执行第二可调用过程；

在用所述第二处理器完成所述执行所述第二可调用过程的步骤时，从所述第二可调用过程调用包含在所述第一可调用过程集合中的一个可调用过程。

10. 权利要求 9 的方法，其特征为：从所述第二可调用过程调用包含在所述第一集合中的可调用过程的所述步骤递归调用所述第一可调用过程。

11. 权利要求 9 的方法，其特征为以下步骤：

在所述第一处理器的第一局部存贮器内保存一个第一程序栈，所述第一程序栈包括一个或多个启动单元，每个启动单元包含所述第一局部存贮器内一例过程的状态信息；

在所述第二处理器的第二局部存贮器内保存一个第二程序栈，所述第二程序栈包括一个或多个启动单元，每个启动单元包含所述第二局部存贮器内一例过程的状态信息。

12. 权利要求 9 的方法，其特征为：从所述第一可调用过程调用所述第二集合中的一个可调用过程的步骤包括；

(*a*)从所述第一可调用过程向包含在所述第一处理器的第一局部存贮器内的第一 *C*—备忘模块发布第一局部调用命令；

(*b*)把所述第一局部调用命令送到包含在所述第二处理器的第二局部存贮器内的第一 *S*—备忘模块；

(*c*)从所述第一 *S*—备忘模块向所述第二集合中的代表所述第一可调用过程的一个可调用模块发布第二局部调用命令；以及

从所述第二可调用过程调用包含在所述第一可调用过程集合中的一个可调用过程的所述步骤包括：

(*d*)从所述第二可调用过程向包含在所述第二局部存贮器内的第二*C*—备忘模块发布第三局部调用命令；

(*e*)把所述第三局部调用命令送到包含在所述第一局部存贮器内的第二*S*—备忘模块；

(*f*)从所述第二*S*—备忘模块向所述第一集合中的代表所述第二可调用过程的一个可调用过程发布第四局部调用命令。

13. 权利要求 11 的方法，其特征为以下步骤：

在所述第一处理器的所述第一局部存贮器内保存第一程序栈，所述第一程序栈包括一个或多个启动单元，其中所述第一局部存贮器包含的每一例过程，所述第一局部存贮器内包含的每一例*C*—备忘模块，以及所述第一局部存贮器内包含的每一例*S*—备忘模块由包含所述例子状态信息的所述第一程序栈的单个启动单元代表；

在所述第二处理器的第二局部存贮器中保存一个第二程序栈，所述第二程序栈包括一个或多个启动单元，其中所述第二局部存贮器包含的每例过程，所述第二局部存贮器包含的每一例*C*—备忘模块，以及所述第二局部存贮器包含的每一例*S*—备忘模块由包含所述例子状态信息的所述第二程序栈的单个启动单元代表。

14. 权利要求 9 的方法，其特征为以下步骤；

调用所述程序的外层过程以开始在所述第一处理内执行所述程序；

自动建立所述第一过程和所述第二集合中的一个可调用过程之间的联系，其中从所述第一可调用过程调用包含在所述第二可调用过程集合中的一个可调用过程的步骤被自动地启动；

自动建立所述第二过程和包含在所述第一可调用过程集合中的一个可调用过程之间的联系，其中从所述第二可调用过程调用包含在第一可调用过程集合中的一个可调用过程的步骤被自动地启动。

15. 一个用于执行具有多个可调用过程的单线计算机程序的方法，其特征为以下步骤：

把所述程序中的所述多个可调用过程的每一个分配到多个可调用过程的集合之一；

在多处理器计算机系统的第一处理器的第一局部存贮器中存入所述多个可调用过程集合的第一集合的可调用过程；

在所述多个处理器计算机系统的第二处理器的第二局部存贮器中存入所述多个可调用过程集合的第二集合的可调用过程；

在所述多处理器系统上执行所述程序，其中所述执行步骤包括以下步骤；

(a)在所述第一处理器上执行包含在所述第一可调用过程集

合内的可调用过程，其中至少有一个所述第一集合的过程调用所述第二集合的过程；

(b)在所述第二处理器上执行包含在所述第二可调用过程集合内的可调用集合，其中至少有一个所述第二集合的过程调用所述第一集合的过程。

16. 权利要求 15 的方法，其中把所述程序中的所述多个可调用过程的每一个分配到多个可调用过程集合之一的步骤包括以下步骤；

对每个所述可调用过程，确定所述多处理器系统的哪个处理应当执行这个可调用过程，其中所述确定步骤的完成不用参考所述程序的调用历史。

17. 权利要求 16 的方法，其特征为所述第一处理器比所述第二处理器更有效地执行第一类型的可调用过程，所述第二处理器比所述第一处理器更有效地执行第二类型的可调用过程，其中所述确定步骤确定所述第一类型的可调用过程应在所述第一处理器上执行，所述第二类型的可调用过程应在所述第二处理器上执行。

18. 权利要求 17 的方法，其特征为所述第一处理器为通用商务处理器并且所述第二处理器为面向数字的处理器。

19. 权利要求 15 的方法，其特征为在所述多处理器系统上执行所述程序的步骤包括；

调用所述程序的外层过程以开始在所述第一处理器内执行所

述程序；

自动建立所述第一可调用过程集合的一个过程与所述第二集合的一个过程之间的联系，其中所述第一集合的过程自动启动对所述第二集合的过程的调用；

自动建立所述第二可调用过程集合的一个过程与所述第一集合的一个过程之间的联系，其中所述第一集合的过程自动启动对所述第二集合的过程的调用。

20. 一个用于调整具有多个可调用过程并且原为在单个处理器系统上执行而编写的单线计算机程序以在一个多处理器系统上执行的方法，所述方法的特征为以下步骤：

建立所述计算机程序的第一分区，所述第一分区包含第一可调用过程集合；

建立所述计算机程序的第二分区，所述第二分区包含第二可调用过程集合，其中第一和第二集合不相交；

其中所述第一可调用过程集合的一个可调用过程包含一个对所述第二可调用过程集合的一个可调用过程的调用；

其中所述第二可调用过程集合的一个可调用过程包含一个对所述第一可调用过程集合的一个可调用过程的调用；

接入用于接收来自所述第一集合的过程发往所述第二集合的过程的局部调用命令的所述第一分区装置；

接入用于代表所述第一集合的过程向第二集合的过程发布局

部调用命令的装置；

接入用于接收来自所述第二集合的过程发往所述第二集合的过程的局部调用命令的所述第二分区装置；并在此中断接入用于代表所述第二集合的过程向所述第一集合的过程发布 局部调用命令的所述第一分区装置。

21. 权利要求 20 的方法，其特征为所述多处理器系统包括第一处理器和第二处理器，其中所述第一处理器比所述第二处理器更有效地执行第一类型的可调用过程，所述第二处理器比所述第一处理器更有效地执行第二类型的可调用过程，其中所述建立第一分区和建立第二分区的步骤把所述第一类型的过程分配到所述第一分区并且把所述第二类型的过程分配到所述第二分区。

22. 权利要求 21 的方法，其特征为所述第一处理是通用商务处理器并且所述第二处理器是面向数字的处理器。

23．一个用于在一个多处理器计算机系统上执行的程序产品，其特征为：

一个以单线方式执行的计算机程序；

一个用以在所述多处理器系统的第一处理器上执行的所述计算机程序的第一分区，所述第一分区包括多个可调用的过程；

一个用以在所述多处理器系统的第二处理器上执行的所述计算机程序的第二分区，所述第二分区包括多个可调用的过程；

一个包含在所述具有用于向所述第二分区内的一个可调用过

程发布调用命令装置的第一分区内的第一可调用过程；

一个包含在所述第二分区内的第二可调用过程，所述第二分区具有用于向包含于所述第一分区内以所述第一可调用过程名义执行的一个可调用过程发布调用命令的装置。

24. 权利要求 23 的程序产品，其特征为所述第二可调用过程包括用于递归调用所述第一可调用过程的装置。

25. 权利要求 23 的程序产品，其特征为：

一个包含在所述第一分区内代表包含在所述第二分区内的一个可调用过程的第一 *C*—备忘模块；

一个包含在所述第二分区内代表所述第一可调用过程的第一 *S*—备忘模块；

一个包含在所述第二分区内代表包含在所述第一分区内的一个可调用过程的第二 *C*—备忘模块；

一个包含在所述第一分区内代表所述第二可调用过程的第二 *S*—备忘模块；

其中包含在所述第一可调用过程中用以向所述第二分区内的一个过程发布调用命令的所述装置包括用于向所述第一 *C*—备忘模块发布第一局部调用命令的装置；

其中所述第一 *C*—备忘模块包括用于向所述第一 *S*—备忘模块发送数据以响应第一局部调用命令的装置；

其中所述第一 *S*—备忘模块包括用于向包含于所述第二分区

内代表所述第一可调用过程的一个可调用过程发布第二局部调用命令的装置；

其中包含在所述第二可调用过程用于向包含于所述第一分区内代表所述第一可调用过程的一个可调用过程发布一个调用命令的所述装置包括用以向所述第二 C—备忘模块发布第三局部调用命令的装置；

其中所述第二 C—备忘模块包括用于把数据送往所述第二 S—备忘模块以响应所述第三局部调用命令的装置；

其中所述第二 S—备忘模块包括用于向包含在所述第一分区内代表所述第二可调用过程的一个可调用过程发布第四局部调用命令的装置。

说　　明　　书

分布式程序栈装置和方法

本发明涉及数据处理软件的用法。更具体地，它涉及在多个处理器上有效地执行单线计算机程序的方法。

一个现代计算机系统一般包括单个中央处理单元(CPU)，和其它支撑硬件，如系统存贮器，通讯总线，输入/输出控制器，存贮设备等。CPU是系统的心脏。它执行包括计算机程序的指令并指导其它系统部件的工作。

在计算机发展初期，CPU是系统中最贵的部分。因此，人们围绕着CPU建系统以优化其用法。能同时为多个用户服务执行各种任务的多任务系统就是这么产生的。多任务系统允许多个用户和任务共享CPU，虽然这种系统能够同时为多个用户服务执行各种任务，但是，在任何瞬间，CPU中正执行的任务只有一个。如果一特定任务需要CPU而CPU正忙，此任务必须等待。因此，多任务系统提高了CPU的使用效率，同时这也意味着CPU是整个系统运行的瓶颈。

随着集成电路的出现，处理器的成本相对其它系统部件下降了。因此，计算机设计为具有多个处理器。例如，数年来，在从属处理器((如磁盘驱动控制器处理器，工作站控制器处理器等)上执行特定

的低级外围功能已成为标准。当这种外围处理器的相对成本下降了时，系统设计者扩充了它们的用途，减轻了 CPU 的工作负担。

最近几年，各种便宜的处理器导致并行和分布式处理系统，包括多处理器执行原来由单个 CPU 所执行的功能的发展。这种多处理器系统中的处理器具有独立的地址空间，可拥有自己的存贮器和内部数据总线以及 I/O。这些处理器可通过共享的总线和共享的存贮器相连，或更松散地，经通讯网络或其它 I/O 控制器相连。

这种多处理器的一个特殊情况是采用了一个通用主处理器和一个面向数字的协处理器。这种面向数字的协处理器体系结构是为执行需大量计算（一般为浮点运算）的应用程序而优化的，而主处理器是为处理数据转移、比较、I/O 等典型的混合指令而优化的。

这种多处理系统有一个问题：为在一个计算机系统上执行而设计的大多数程序本质上是单线的。这时所用的单线"意指程序只包含单个控制流，从而在任意瞬间，都在执行单指令序列。这种序列可以循环或跳到编码中的另一点，但永远是单路径的。这种单线程序和多线控制不同。在多线控制中，程序流可像岔口处的道路一样分开，同时在两条路径上继续进行。一个单线程序不太适合在多处理器上执行。

当单线程序在一个包含不同类型处理器的多处理器系统中执行时，必须把此程序的各个部分分配到不同的处理器上执行。一种方法是重写单线代码以支持不同的控制流，实现多处理器优化。某些计算

机语言支持这种多道处理，尽管只有小部分现有计算机程序是用这些语言写的。例如，*SIMULA* 语言支持使用联立程序，可执行联立的多线程序。但是，这种方法不是永远可行；即使可行，重写现有编码也太贵。

另一种向多处理器分配程序各部分的方法是顾客——服务员模型(*the client—server model*)，它被广泛地用于分布式处理系统。每个程序部分在某个处理器(顾客)上执行。当它需要具有与顾客处理器不同能力的另一处理器(服务员)时，它发布一个请服务员为其服务的请求，服务员在工作结束时把控制连同工作结果一起(如果需要)返回顾客。顾客——服务员模型使不同的处理器在执行程序时相互合作，但合作程度有限。顾客必须在执行开始以前，通常是在知道需要何种信息以前，向服务员提供所有需要的信息。现有顾客——服务员模型本质上是单向的；服务员没有能力发布一个调用顾客的命令。

在一个多处理器系统中向不同处理器分配程序的不同部分时对编码的改变最好不要很大。具体地，在具有通用主处理器和面向数字的协处理器的系统情形里，最好在协处理器上执行面向数字的过程，并在主处理器上执行其它过程。不幸的是，先有技术机制限制了系统以一种优化方式分配过程的能力。

因此本发明的一个目的是提供一种在多处理器计算机系统上执行程序的改进方法和装置。

本发明的另一目的是提供向多处理器计算机系统中的不同处理器分配程序各部分的改进方法和装置。

本发明的另一目的是增加向多处理器计算机系统中的不同处理器分配程序各部分的灵活性。

本发明的另一目的是增加在多处理器计算机系统上执行过程的效率。

本发明的另一目的是减少单线程序所需改变量使其在一个多处理器计算机系统上有效地运行。

本发明的另一目的是减少有多处理器计算机系统上执行程序的成本。

本发明的另一目的是提供在具有通用处理器和面向数字的协处理器计算机系统上执行程序的改进方法和装置。

一个包括多个程序模块的计算机程序(每个模块具有一个或多个可调用过程)在一个多处理器系统上执行。每个程序模块在此系统的一个处理器上执行,而任一处理器都可以执行多个模块。每个处理器局部可寻址存贮器包含程序栈、在此处理器上执行的每个模块的目标代码、以及含 有用来处理与其它处理器通讯的链接信息的代理栏(*agent object*)和数据结构。另外,局部存贮器还包括可在另一处理器(能被局部处理器中某个过程调用)上执行的每个过程的 *C*—备忘模块(*c—stub module*),以及局部存贮器中可被一个在另一处理上执行的过程调用的每个过程的 *S*—备忘模块。一个处理器的程序

模块，备忘模块栈和代理程序的集合称为一个分区。

当在处理器 A 上执行的过程 $P1$ 希望调用在处理器 B 中执行的过程 $P2$ 时，它向对应于处理器 A 的局部可寻址存贮器中 $P2$ 的 C—备忘程序发布一个局部调用命令。$P2$ 的 C—备忘程序调用处理器 A 中的代理处理程序，此代理处理程序与处理器 B 中对应的代理处理程序通讯，处理器 B 中的代理处理程序使得对应于过程 $P2$ 的处理器 B 中的 S—备忘程序发出一个调用过程 $P2$ 的局部调用命令。从一个过程返回按同一路线反过来即可。每个处理器独立地维护其自身的程序栈，其栈入口参照局部可执行过程，局部备忘程序或局部代理程序随着每次局部调用或返回，所述处理器的程序栈被适当地修改。我们称这种机制为“分布式栈”。

因为每个处理器参照局部过程，备忘程序和代理程序独立地维护其自身的栈，所以远程过程调用不像在典型的先有技术顾客——服务员模型(单向的)的情形中那样，会受到一个处理程序过去调用历程的限制。处理器 A 中过程 $P1$ 或以调用处理器 B 中的过程 $P2$，处理器 B 中的过程 $P2$ 又可调用处理器 A 中的过程 $P3$。另一方面，过程 $P2$ 可递归调用处理器 A 中的过程 $P1$。

根据本发明，为在单处理器系统上执行而编写的常规单线程序可转到在多处理器系统上操作，几乎或完全不需修改源代码。对每个程序模块都要作出应在系统的哪个处理器执行的决定。这个决定与对其它模块作出的决定无关。如果需要，备忘程序将自动生成。所有

过程调用都局限于调用过程，因而对源程序不需作任何修改，而将事实上的局部过程调用与远程过程调用区别开来。如果需要，装载处理程序将自动启动远程处理器的分区，并返回必要的链接信息以建立不同处理器间的通讯。

在最佳实施例中，多处理器系统包含了一个连到 *IBM RISC System/6000* 系统处理器上的 *IBM Application System/400* 系统处理器。后面的处理器是为通用商务处理而设计的；前面的处理器是减少的指令集（*RISC*）处理器，是为科学/工程应用而设计的。*RISC* 处理器作为包含大量数字操作，特别是基本上为浮点操作的过程的加速器。在本最佳实施例中，支持通用商务的过程在通用处理器上执行；而包含大量数字操作的过程在 *RISC* 处理器上执行。

图 1 图示了根据本发明的最佳实施例执行单线程序的多处理器的主要硬部件；

图 2 图示了一个常规程序是如何在单处理器计算机体系结构中执行一系列嵌套过程调用的；

图 3 图示了图 2 的常规程序是如何在根据最佳实施例的多处理器系统上执行的；

图 4 图示了根据最佳实施例的分区中参加一个远程过程调用的不同成分；

图 5 用图示了根据最佳实施例的分布式栈；

图 6 图示了根据最佳实施例的一个代理状况栈的结构；

图 7 图示了根据最佳实施例的代理程序之间信息的结构；

图 8 图示了单个远程过程调用和返回的步骤中的控制转移；

图 9 是根据最佳实施例的单个远程过程调用和返回步骤中由 *C*—备忘程序和 *S*—备忘程序所执行步骤的流程图；

图 10 是根据最佳实施例的代理程序所执行步骤的流程图；

图 11 图示了根据最佳实施例，为远程过程调用保存路径信息的表格结构；

图 12 图示了根据最佳实施例的装载/启动程序机制。

图 1 图示了根据本发明最佳实施例的多处理器系统的主要硬部件示意图。多处理器系统 100 包含一个通用商务系统 101，它连在专用数字加速器系统 121 上。通用系统 101 由常规复杂指令集(*CISC*)*CPU*102 驱动。*CPU*102 经存贮器总线 104 与局部存贮器 103 和处理器接口单元 105 通讯。处理器接口单元 105 处理存贮器总线 104 和系统总线 106 之间的通讯。系统总线/*I/O* 通道适配器 107 接到系统总线 106 上。额外总线接口设备 108，109 可接到系统总线 106 上，允许 *I/O* 设备(未示出)经 *I/O* 总线 110，111 与系统总线 106 通讯。数字加速器系统 121 由减少的指令集(*RISC*)的 *CPU*122 驱动，这种 *CPU* 是为如科学/工程应用等需要大量数字计算的应用而设计的。*RISC* *CPU*122 经存贮器总线 124 与局部存贮器 123 通讯。*CPU*122 也连到系统总线 125 上，使它能经总线接口 126 与 *I/O* 通道适配器 127 通讯。*I/O* 通道 128 连在 *I/O* 通道适配

器 127 和系统总线/I/O 通道适配器 107 之间，因而连在 通用系统 101 和数字加速器系统 121 之间 。I/O 通道 128 还 能 把许多 I/O 设备（未示出 ）经 I/O 设备适配器 129，130 连到系统 121 上。尽管在图中系统 101 只接有两条 I/O 总线 110 ，111 以及两个总线接口设备 108，109 ；系 统 121 只 接有两个 I/O 设备适配器 129，130，但是应当理解，接到系统上的这种设备的实际数字可多可少。

局部存贮器 103 在 CPU102 的地址空间里；局部存贮器 123 在 CPU122 的地址空间里。尽管在图中这些存贮器是单个部件，但是应当理解，每个处理器的局部存贮器实际上可以是分组存贮器系统，如小型的相对快的高速缓冲存贮器和慢而大型的主存贮器。还应当理解，每个系统一般包括一个或多个如磁盘驱动器的局部大容量存贮设备，并具有把数据从这种大容量存贮设备装进局部存贮器（如果需要）的机制。这种存贮设备的使用同行皆知。对本发明来说，局部存贮器可概念性地视为在相应处理器的地址空间中的单个部件。

在最佳实放例中，系统 101 是 IBM Application System/400 系统，而系统 121 是 IBM RISC System / 6000 系 统 。I/O 通道 128 是 IBM Micro—Charmel 总线。系统 121 主要作为从属系统以改进主系统 101 的性能。在最佳实施例中，系统 101 和系统 121 物理地装在同一个单元里，接收来自公用电源的电能。把系统装在一起使得系统间的数据可高速传送。但是，应当理解，本发明可以用于其它结构的其它计算机系统，并且系统不一定要一起装在同一物理单

元里。

一个常规单线计算机程序包括多个可以嵌套或递归形式相调用的过程。图2图示了在常规单处理器计算机体系结构中这种程序的一第列嵌套过程的调用。在图2的例子中，过程A210向过程B202发布一个把控制转给过程B的调用信念211。过程B202然后向过程C203(在返回过程A之前)发布一个把控制转给过程C的调用命令212。过程C203向过程D204(在返回过程B之前)发布一个调用命令213。过程D在D执行完毕时把控制214返回给过程C。后来过程C在C执行完毕时把控制215返回给过程B,过程B在B执行完毕时把控制216返回给A。在这种常规系统中,过程调用一个先前已调用的过程是可能的,这种调用称为递归。例如,过程C可以调用过程B而不是过程D,或者可以调用它自身。在一个常规单处理器体系结构中,一种称为栈的数据结构保存一个过程返回到调用它的过程时在正确的位置恢复程序的执行时所必需的状态信息。一个栈包含一个或多个过程启动单元。每个单元含有某个过程的状态信息,如下一个要被执行的指令,局部变量状态等。当调用一个过程时,栈中新加一个启动单元,并且存入被调用过程的状态信息。当一个过程返回时,从此启动单元读出状态信息并取消栈中顶层启动单元。

根据本发明,组成程序的过程分在几个分区内,每个独立处理器有一个分区。一个程序的外层过程在其中执行的分区(即程序开始在其中执行的分区)被称为主分区,所有其它分区被称为次分区。图3

概念性地图示了图 2 的程序如何在具有两个分区的计算机系统 100 上执行的。在这个例子中，分区 I 301 是主分区，分区 II 302 是次分区。过程 *A*303 和过程 *C*305 已被分到分区 I，而过程 *B*304 和过程 *D*306 已被分到分区 II。当过程 *A* 向过程 *B* 发布调用命令 311 时，调用经过分区界线，因而被称为“远程过程调用”，过程 *B* 然后向过程 *C* 发布远程过程调用命令 312，而过程 *C* 又向过程 *D* 发布远程过程调用命令 313。过程 *D* 在 *D* 执行完毕时把控制 314 返回到过程 *C*，*C* 返回 315 到 *B*，*B* 返回 316 到 *A*。和单处理器系统的情形一样，经分区界线递归调用过程是可能的。例如，过程 *C* 可以调用过程 *B* 而不是过程 *D*。

可根据各种性能或其它考虑把过程分到各个分区。在图 3 的例子中，过程 *B* 和 *D* 可能是面向数字的过程，这种过程被分配到包含 2 一个 *RISC* 处理器的分区里以运用此处理器的能力进行复杂的计算。过程 *A* 和 *C* 可能是具有大量 *I/O* 或其它一般混合指令的过程。但是在分配过程时也可能有其它考虑。例如，在另一实施例中，一个分布式处理系统可包含多个相同的处理器，在各自的局部存贮器中有不同的数据库。在这种情形里，可根据其存取的数据库把过程分到各处理器。

发布远程过程调用的过程起顾客作用。被调用的过程起服务员作用。单个过程相对某些调用的顾客，相对另外的调用是服务员。在图 3 的例子中，过程 *A* 在发布远程过程调用 311 时是顾客，过程 *B*

相对此调用是服务员。但是，过程 B 在发布远程过程调用 312 时是顾客，而过程 C 是服务员。

图 4 图示了一个分区的不同单元以及在最佳实施例的远程过程调用中它们是如何相互作用的。每个分区包含一个或多个过程代码单元 402，412，一个或多个 C—备忘程序 403，413，一个或多个 S—备忘程序 404，414，以及一个代理程序 405，415。过程代码单元 402，412 包含运行此过程的机器可执行指令。每个过程代码单元可以是一个独立的编译模块，或者在分区的单个模块里编译多个单元。一个编译模块永远包含在单个分区内，不管它具有一个还是多个过程。

参见图 4，如果分区 401 内的过程 X402 发布一个调用分区 411 内的过程 Y412 的命令，分区 401 内过程 Y 的 C—备忘程序 403 收到此调用命令。C—备忘程序 403 调用它的局部代理程序 405，请求它启动远程服务员。代理程序 405 发布一个经分区界线调用代理程序 415 的命令。代理程序 415 然后调用过程 Y 的 S—备忘和谐 414，S—备忘程序则向过程 Y 发布一个好像来自另一局部过程（从过程 Y 的角度来看）的局部调用命令。过程 Y 可经过 C—备忘程序 413，代理程序 415，代理程序 405 和 S—备忘程序 404 调用过程 X。

备忘程序使得远程调用对调用过程来说好像是局部的。因此发布调用命令的过程不必知道被调用的过程是否在同一分区。它只须简单地发布一个标准的局部调用命令不用考虑其它因素。一个 C—

备忘程序接收来自同一分区内顾客过程的局部过程调用与另一分区内代表服务员的C—备忘程序一样。一个S—备忘程序向同一分区内服务员过程发布的局部过程调用与另一分区内代表顾客过程的S—备忘程序一样。对每个作为服务员的过程来说，在服务员所在分区内有一个独立的S—备忘程序；在每个其它分区内有一个独立的C—备忘程序。只有作为服务员的过程才需要备忘程序。

每个分区有一个代理程序，它处理与其它分区的连接。更精确地，代理程序负责经分区界线转移程序控制。不同分区的代理程序经一种通讯媒介相互通讯。在最佳实施例中，这种通讯媒介是系统101和系统121之间经I/O通道128的路径。在一给定的分区内，同一代理程序既处理发出的远程过程调用，又处理传来的远程过程调用。因此，代理程序是单线控制转走和返回到这个局部分区的接点。每个代理程序具有有关局部调度哪个服务员以响应传来的远程调用命令的信息，以及有关怎样向其它分区的服务员发出远程调用命令的信息。

每个分区包含一个独立的局部栈，它具有此分区内过程，备忘程序和代理程序的启动单元。总的说来，分区内的独立局部栈包含所有维护程序流所需的程序状态信息。发明者称这种结构为“分布式栈”。

图5图示了这种分布式栈的例子以说明图3所示远程过程调用如何使用这种分布式栈。在此例中，分布式栈500包含两个局部栈501,531，分别位于图3的分区Ⅰ301和分区Ⅱ302内。因为分区Ⅰ是主分区，首先进入栈501底的是过程A的启动单元。当过程A执行

时，且在它发布调用命令以前，栈 501 中只有这一个启动单元。栈 531 可能包含也可能不包含分区 Ⅱ 内代理程序 Ⅱ 的启动单元 532，这取决于分区 Ⅱ 是否被启动(以后再讨论)。当过程 *A* 向过程 *B* 发布远程过程调用命令 311 时，它实际上向分区 Ⅰ 内过程 *B* 的 *C*—备忘程序发布局部调用命令 511。使得 *C*—备忘程序启动单元 503 进入栈 501。过程 *B* 的 *C*—备忘程序然后调用(512)分区 Ⅰ 的代理程序(代理程序 Ⅰ)，使得代理程序 Ⅰ 的启动记录 504 进入栈 501。代理程序 Ⅰ 然后与代理程序 Ⅱ 通讯(513)。如果分区 Ⅱ 还未启动，则装载/启动实用程序被启动以启动分区 Ⅱ (以后详述)。装载/启动实用程序初始化栈 531 并在此栈的底部放入代理程序 Ⅱ 的启动单元。代理程序 Ⅱ 然后调用(541)过程 *B* 的 *S*—备忘程序，使得 *S*—备忘程序的 启动 单元 533 进入栈 531 。过程 *B* 的 *S*—备忘程序调用(542)过程 *B*，使得过程 *B* 的启动单元 534 放入栈 531。因此，在图 3 中示为单个概念性远程调用的远程过程调用 311 事实上包含一系列局部调用 511，512，541 和 542，以及代理程序之间的远程调用 513。当过程 *B* 正在执行时，且在它调用其它过程以前，栈 501 包含过程 *A*502，*B* 的 *S*—备忘程序 503 和代理程序 Ⅰ 504 的启动单元，而栈 531 包含代理程序 532，*B* 的 *S*—备忘程序 533 和过程 *B*534 的启动单元。

分布式栈 500 对图 3 中所剩远程过程调用按同样方式增长。过程 *B* 为了向过程 *C* 发布远程过程调用命令 312 ，过程 *B* 调用(543)分区 Ⅱ 内 *C* 的 *C*—备忘程序，在栈 531 上加入启动单元 535 ，*C* 的 *C*

—备忘程序调用 (544)代理程序Ⅱ，在栈 531 上 加 进 启动单元 536；代理程序Ⅱ与代理程序 I 通讯(545)；代理程序 I 调用(514)分区 I 内 C 的 S—备忘程序，在栈 501 上加入启动单元 505；且 C 的 S—备忘程序调用(515)过程 C，在栈 501 上加入启动单元 506。当过程 C 向过程 D 发布远程过程调用命令 313 时，过程 C 调用(516)分区 I 内 D 的 C—备忘程序在栈 501 上加入启动单元 507；D 的 S—备忘程序调用(517)代理程序 I，在栈 501 上加入启动单元 508；代理程序 I 调用(518)代理程序(Ⅱ)；代理程序Ⅱ调用(546)分区Ⅱ内 D 的 S—备忘程序在栈 531 上加入启动单元 537；D 的 S—备忘程序调用(547)过程 D，在栈 531 上 加 入启动单元 538 。在过程 D 执行时，分布式栈 500 包含了所有图 5 所示的启动单元。

从各过程返回，按同一序列返过来即可 。当过程 D 执行完毕时，它是这样返回到过程 C 的(图 3 中示为单个返回 314)：返回(548)到 D 的 S—备忘程序，从栈 531 上除掉 D 的启动单元 538；D 的 S—备忘程序返回(549)到代理程序Ⅱ且从栈 531 上 除 掉 启动单元 537；代理程序Ⅱ与代理程序 I 通讯(550)；代理程序 I 返回(519)到 D 的 C—备忘程序且从栈 501 上 除 掉 启 动 单 元 508；D 的 C—备忘程序返回(520)到过程 C，从栈 501 上除掉单元 507。注意在返回经过分区界线时，代理程序Ⅱ的启动单元 536 并未从栈中除掉，因为程序Ⅱ在控制重新经过分区界线时还要作为入口点。过程 C 通过循序返回到 C 的 S—备忘程序 521、代理程序 I 522、代理程序Ⅱ 523、C

的 *C*—备忘程序 551 和过程 *B*552，返回到过程 *B*，其间启动单元 506，505，536 和 535 从它们各自的栈中被除去。过程 *B* 通过循序返回到 *B* 的 *S*—备忘程序 553、代理程序Ⅱ 554、代理程序Ⅰ 555、*B* 的 *C*—备忘程序 524 和过程 *A*525，返回到过程 *A*，其间启动单元 534，533，504 和 503 从它们各自的栈中被除去。

每个代理程序维持着一个内部代理状况栈。图 6 图示了代理状况栈的结构。代理状况栈 601 记录了代理状况，使得在调用或返回导致控制转移到代理程序时，代理程序能在合适的位置恢复执行。栈 601 包含一个或多个代理状况记录 602，每个记录包含一个状态域 611，特征域 612，模块识别符 613，过程识别符 614 和 分区识别符 615。状态域包含此代理程序的当前状态，如 *AGT—IN—SERVICE*，*AGT—CALL*，*AGT—RETURN* 等。特征域 612 是一位表示各种运行时间条件的向量。模块识别符 613，过程识别符 614 和分区识别符 615 分别包含被此代理程序调用的过程的模块、过程和分区标识数，不管这个调用是代表远程过程的局部调用还是代表局部过程的远程调用。在调用一个远程服务员（代表局部顾客）时，当前的状况被下推进栈 601 并被修改以反应代理程序的新状态。当远程调用返回时，旧的状况被从栈的顶层复原（上托）掉。在收到一个远程请求（代表局部服务员）时，当前的状况在局部服务启动以前被下推进栈 601，并在服务完成时被从栈的顶层复原（上托）掉。

图 7 图示了分区间信息的格式。动词域 701 包含信息所请求的

动作，可以是 *AGT－CALL*，*AGT－REPLY*，*AGT－RETURN*，*AGT－ERROR*，*AGT－DEBUG* 或 *AGT－TERMINATE*。前三条信息将在下面解释；其余信息从字面上可以理解并且对本发明的意义不大。特征域702包含的信息与代理状况记录的特征域602相同。目标模块识别符域703和目标过程识别符域704分别包含信息目的地的模块和过程。原分区识别符705包含发送分区的识别符。编组参数域706包含被传递参数的编码。

在最佳实施例中，参数编码采用对分级数据结构进行递归下降测量直到识别出原始简单数据元素的方法，然后把这些数据以与机器无关的格式传到另一处理器上。这种操作被称为参数编组。参数解码时发生逆操作，这时把从与机器无关格式复原的原始数据元素建成分级数据结构。这种操作称为参数的非编组。参数编组/非编组技术同行皆知，其细节不在这里重复。同行所知任何其它参数编码/解码方案也可以采用。

现在详细描述远程过程调用步骤。图8概念性地图示了在单个远程过程调用和返回的步骤中顾客、备忘程序、代理程序和服务员间的控制转移。图9图示了作为单个远程过程调用的一部分，顾客和服务员备忘程序所执行步骤流程图。图10是代理程序所执行的流程图。

在开始一个远程过程调用之前，程序的状态如下。将要发出远程过程调用命令的分区正在执行发布此命令的顾客过程。其它所有分

区正在执行各自的代理程序，此时代理程序的状态是 *AGT—WAIT*，等待远程分区出现的事件。这种状态在图 8 中由 801 表示。为响应远程调用，代理程序也可以是 *AGT—GET—REPLY* 或 *AGT—ACCEPT RETURN* 状态。但是，在下面解释中，*AGT—WAIT* 状态代表所有这三种状态。

当顾客过程准备启动远程服务员时，它向所述服务员的局部顾客备忘程序发出一个局部过程调用(802)，这个调用对此顾客来说和其它任何局部过程调用一样。这个调用启动了运行 901 的 *C*—备忘程序，此备忘程序然后用 *AGT—CALL* 信号向局部代理程序发出一个调用命令(803,902)，请求它启动远程服务员。顾客代理程序建立一个远程联系并把 *AGT—CALL* 信息传给正在 *AGT—WAIT* 状态下等待的服务员代理程序(804)。

图 10*A* 图示了顾客代理程序在收到 *AGT—CALL* 信息以后采取的行动。顾客代理程序首先检验状况（在现有状态下接收 *AGT—CALL* 是否合法)(1001)。如果状况合法，当前代理状况在步骤 1002 存进代理状况栈 601。代理程序然后在步骤 1003 采用动词 *AGT—CALL*、目标模块和过程识别符、原局部分区识别符构造一条信息。在步骤 1004 设置新代理状况，在步骤 1005 把信息送到目标分区。当服务员代理收到这条信息，顾客代理将控制返回到 *C*—备忘程序。

图 10*B* 图示了服务员代理收到来自顾客代理的信息后所采取的行动。服务员代理在 *AGT—WAIT* 状态下收到任何信息以前初步

设置它的状态为 *AGT—NULL—CODE*(1011)。代理程序然后在其缓冲区收到信息(1012)并检查动词。如果动词 不是 *AGT—CALL*(1013)并且不是 *AGT—TERMINATE*(1014),发出错误信号。如果动词是 *AGT—TERMINATE*,代理处理程序结束其分区。如果动词是 *AGT—CALL*,接收的代理程序把其当前状况下推入代理状况栈(1015),设置来自信息头的新状况特征(1016),设置其状态为 *AGT—IN—SERVICE*(1017),通过调用局部服务员备忘程序分派服务(1018,805),使得 *S* 备忘程序启动(911)。以后当服务员完成工作并返回时从代理状况栈复原其状况(1019)。

随着控制返回到顾客分区的 *C*—备忘程序和在服务员分区启动的 *S*— 备忘程序,*C*—备忘程序把输入 参数送到 *S*—备忘程序(903,806),*S*—备忘程序收到这些参数(12,807)。在所有输入参数被送出后,*C*—备忘程序调用顾客代理,发生了 *AGT— GET—REPLY* 事件(904,808),示意它等待远程服务返回。顾客代理然后进入 *AGT—WAIT* 状态(809)。在这种状态下,它可接收来自某个远程代理的两种信号:*AGT—REPLY* 或 *AGT—CALL*。*AGT—REPLY* 信号来自于服务员过程完成后当前正在进行的远程过程调用的服务员代理程序。*AGT--CALL* 信号将作为新远程过程调用某个局部服务的一部分出现,它被嵌套于原远程过程调用里。这种调用可来自任何其它代理。

在服务员备忘程序收到所有输入参数后,它向服务员过程发出

一个局部调用命令(810,913),以与任何局部调用相同的形式向它传送参数。服务员过程然后得以执行。服务员过程可发出调用其它局部或远程过程的嵌套调用命令,所用过程如上所述。但在此例中,只描述单个调用和返回。当服务员过程完成时,它返回到*S*—备忘程序(811)。

S—备忘程序用 *AGT—REPLY* 信号启动服务员代理程序(812,914),示意它提醒顾客服务已经完成。图 10c 图示了服务员代理程序在发出 *AGT—REPLY* 信号以后(图 8 中 813 所示)所采取的行动。服务员代理程序首先检验状况(在当前状态下发出 *AGT—REPLY* 是否合法)(1021)。如果状况合法,代理程序在步骤 1022 用动词 *AGT—REPLY*,目标模块和过程识别符,来自当前代理状况的原分区识别符构造一条信息。服务员代理程序然后把此信息送往顾客分区(1023)设置其状况为 *AGT—IN—SERVICE*(1024)。当信息收到时,服务员代理返回到*S*—备忘程序。

图 10*D* 图示了顾客代理在接收来自服务员代理的 *AGT—REPLY* 信息以后所采取的行动(图 8 中 814 所示)。当顾客代理完成启动服务员的工作时,它清除其输出数据(1031)并进入等待状态,等待前述的来自另一分区的信息。当信息到达时,它被放进代理程序的缓冲区内(1032)并检查动词。如果动词是 *AGT—REPLY*(1033),它检验状况(1034)。如果 *AGT—REPLY* 信号在当前状况下是合法的,则代理程序返回到局部*C*—备忘程序(1035,904),如果动词是

AGT—CALL（1036），表明有一个嵌套的远程过程调用。代理程序将把当前状况下推进代理状况栈，重置来自信号信头数据的状态域，设置其状态为 *AGT—IN—SERVICE*，通过启动适当的 *S*—备忘程序分派服务(1037)。当局部服务完成时，它复原代理状况栈的先有状况(1038)并重新进入等待状态。

当控制从顾客代理返回到局部 *C*—备忘程序和从服务员代理返回到 *S*— 备 忘程序时 ，*S*—备忘程序把输出参数送到接收它们(816,905)的 *C*—备忘程序(815,915)，在所有输出 参 数 被接收后 ，*C*—备忘程序用 *AGT—ACCEPT—RETURN* 信号调用顾客代理(817,906)，示意它期望来自服务员代理的远程过程调用返回。几乎在同时，*S*—备忘程序用 *AGT—RETURN* 信号把控制返回到服务员代理(818,916)，示意它把远程过程调用返回到顾客代理。*S*—备忘程序完成了它的作用。

图 10*E* 图示了在发出示意自 *S*—备忘程序返回的 *AGT—RETURN* 信号后，服务员代理所采取的行动。服务员代理检验状况(1041)。如果在当前状态下发布 *AGT—RETURN* 是合法的 ，此代理程序构造了一条发往顾客代理的信息 ，把动词 *AGT—RETURN* 放进信息头，还采用了来自当前状况的原模块，过程和分区识别符(1042)。代理程序然后把信息送到顾客代理(819,1043)。当信息传送完毕，服务员代理清洗它的输出缓冲区(1044)并把控制返回到 *S*—备忘程序。*S*—备忘 程 序 也执行完毕并把控制返回到其分配程序

，即 *AGT—WAIT* 状态（正在执行 *AGT—WAIT*，*AGT—GET—REPLY*，或 *AGT—ACCEPT—RETURN*）下的代理程序。在这一时刻，代理程序正在等待另一调用。

图 10*F* 图示了在收到 *AGT—RETURN* 信息（图 8 中 820 所示）后顾客代理所采取的行动。在顾客代理收到来自 *C*—备忘程序的示意它期望远程过程调用返回的调用命令后，顾客代理处于等待状态。当信息收到时，信息被读进缓冲区（1015）且动词被检查。如果动词是 *AGT—RETURN*，代理程序检验状况（1052）。如果接收这条信息是合法的，代理程序复原来自代理状况栈的状况（1053），并返回到 *C*—备忘程序。在等待 *AGT—RETURN* 时，也有可能收到另一条（嵌套）*AGT—CALL* 信息，如 1054 处所示在这种情形里，代理程序将把当前状况下推进代理状况栈，重置来自信息头数据的状况域，设置其状态为 *AGT—IN—SERVICE*，并通过启动适当的 *S*—备忘程序分派服务（1055）。当局部服务完成时，它复原其来自代理状况栈的先有状况（1056）并重新进入等待状态。当局部 *C*—备忘程序接收到代理程序的返回时，它把参数返回到（局部返回）顾客过程（821）。

每个分区维护着几张表，它们包含执行远程调用的路径信息，如图 11 所示。它们包括模块到分区的映射表 1101，模块分派表 1111，和过程分派表 1121。分区被标以数字 0 到 n，其中共有 $n+1$ 个分区。主分区被标为 0。模块被标以数字 0 到 m，共有 $m+1$ 个模块。模块到分区表 1101 的入口数等于模块数，每个入口包含具有此模块的分

区识别符，索引此入口。在每个分区都复制了表 1101，使得代理程序把远程过程调用送到正确的分区。模块分派表 1111 的入口数等于模块数，每个入口包含一特殊模块——备忘程序的局部地址。由于任一分区不包含所有的模块，模块分派表中包含了一个错误处理过程的局部地址以处理那些不在这局部分区内的模块。尽管每个分区有一个模块分派表 1111，但是，每个分区的入口将不同。每个模块——备忘程序包含一个过程分派表 1121。过程分派表的入口数等于此模块中服务过程之数，每个入口包含此过程 S—备忘程序的局部地址。当代理程序收到一个调用远程分区服务的命令时，它从表 1111 中获得模块——备忘程序的地址，然后启动此模块——备忘程序，由此从表 1121 中获得 S—备忘程序的地址，然后启动这个 S—备忘程序。

根据本发明，一个程序的不同部分将放于不同处理器的局部存贮器里。当程序开始执行(在其主分区)时，必须具有某种机制确保在次分区的程序模块在需要时被装载。在本最佳实施例中，由单独的装载/启动处理程序完成此任务。

图 12 图示了此装载/启动程序机制。装载程序 1205，启动程序 1206，1207，主分区 1203 和次分区 1208，1209 每个都有一个主入口，在它执行时分配给它。主入口对分区间建立低级联系很有必要。每个启动程序 1206，1207 是操作系统处理程序，每个处理一个。装载程序知道它的主入口名。启动程序 1206，1207 在接到来自装载程序

1205 的命令后启动各自的分区。装载程序可含于主分区所在局部地址空间内，也可在另一处理器的地址空间内。当一个母处理程序 1202 启动了此程序 。母处理程序把装载程序的主入口数（图 12 中的“*M*2011”）经环境 1204 传给主分区。“环境”意指任何用来传送入口数的机制，如一个环境变量，外部文件或其它装置。装载 程序 1205 调用每个启动程序 ，传给它装载程序的主入口名。启动程序然后启动对应的次分区。启动分区需要为次分区建立程序栈，在此栈的底部有一个代理处理程序的启动单元。次分区的代理处理程序将在 *AGT—WAIT* 状态下被启动。主分区和次分区用先前提供的装载程序主入口名建立了与装载程序的联系，并向装载程序提供相应分区的主入口名。装载程序然后把入口名分给其它所有分区。

尽管在最佳实施例中，一个程序的所有分区在主分区被启动时都被装载并链接起来。但在另一实施例中，也可推迟装载直到一个远程过程调用启动包含在未装在分区的过程（要求装载）。在这个可选实施例中，每个分区将保存一个含以前被装载分区的主入口的向量。当开始一个远程过程调用时，顾客分区首先检查此向量以检验目标分区是否已装载。如果目标分区设有装载，目标处理器的启动程 序在来自调用分区内装载功能的命令发出后启动这个目标分区。一次只启动一个分区，但在别的方面，要求装载功能和最佳实施例中的预装载所有分区一样。

在最佳实施例中，基于性能或其它考虑把每个独立地编译代码

模 块分到一个处理器上 。分配是在模块级而不是在过程级进行的，以避免修改源代码。如果希望在独立的过程级分配模块，则可以对每个过程独立地进行编译(在大部分计算机语言里)。在另一实施例中，编译程序指令可包含在源代码内，它允许单个模块内的不同过程可编译成用于不同分区的不同模块。

在最佳实施例中，本发明被用来帮助一个连在通用计算机系统上的面向数字的加速器系统，其中加速器主要作一个从属系统。但是，还有许多可能的系统构造和使用实施例。例如，所有处理器都相同，但每个处理器存取不同的局部数据库。通讯链接可比最佳实施例中所用的更远或更近。例如，不同的处理器可通过各处局域网技术链接。

虽然公开了一个特殊的本发明实施例和某些可选实施例，但是同行专家将认识到在下述权利要求范围内可做一些形式上和细节上的修改。

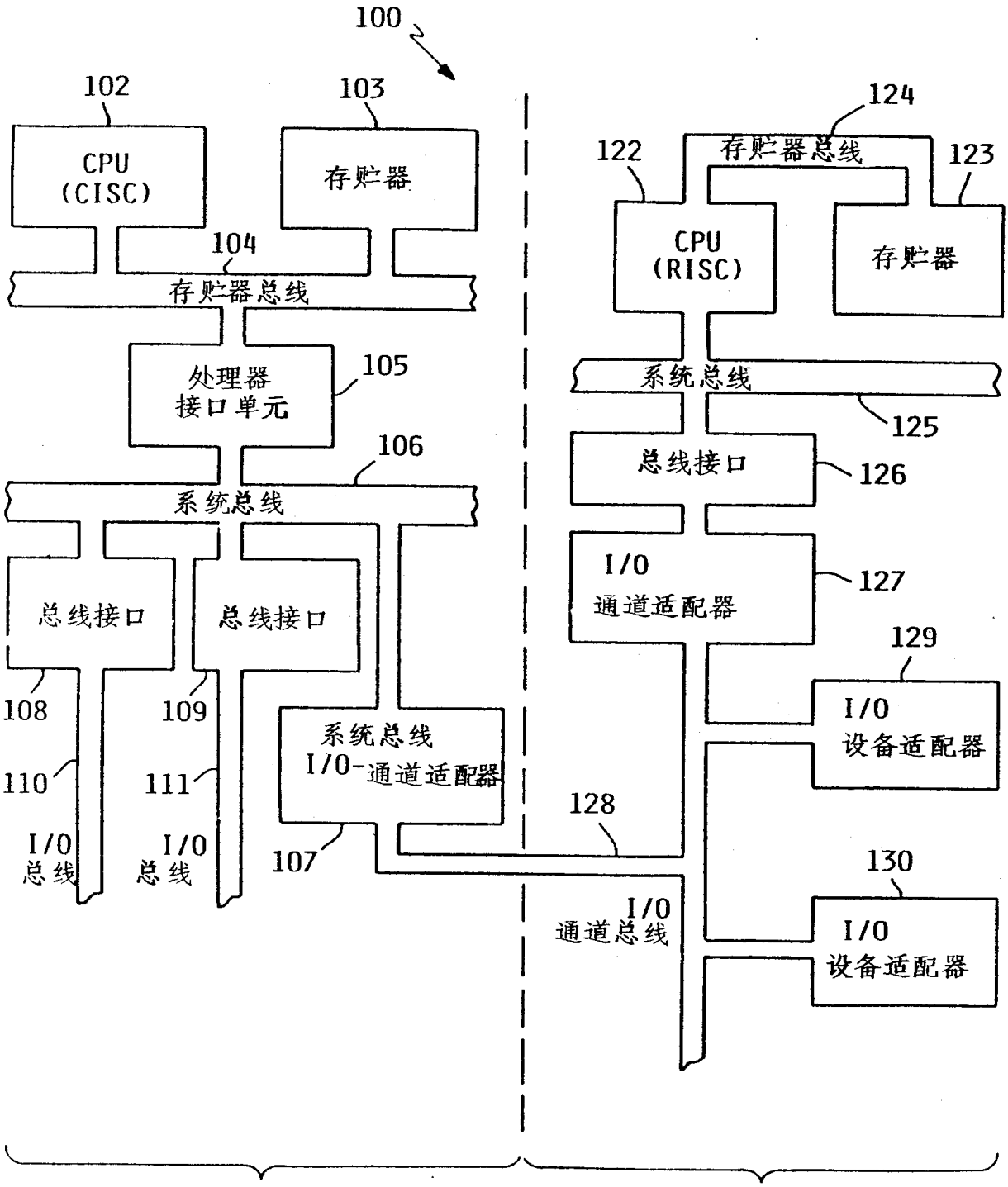

100
102
CPU
(CISC)
103
存贮器
104
存贮器总线
处理器
接口单元
105
106
系统总线
总线接口
总线接口
108
109
110
111
I/O
总线
I/O
总线
系统总线
I/O-通道适配器
107
101
124
存贮器总线
122
123
CPU
(RISC)
存贮器
系统总线
125
总线接口
126
I/O
通道适配器
127
129
I/O
设备适配器
128
130
I/O
通道总线
I/O
设备适配器
121

图1

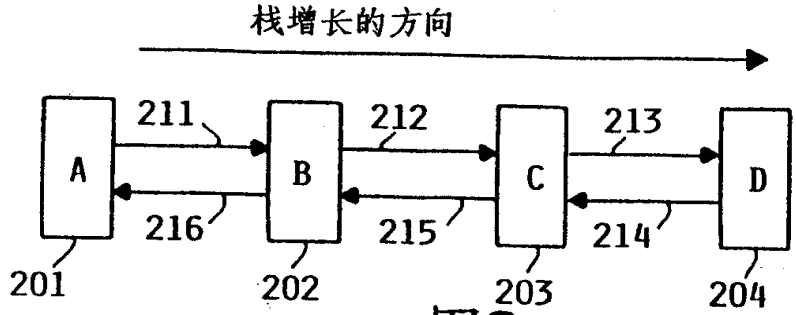

栈增长的方向
A
B
C
D
211
212
213
216
215
214
201
202
203
204

图2

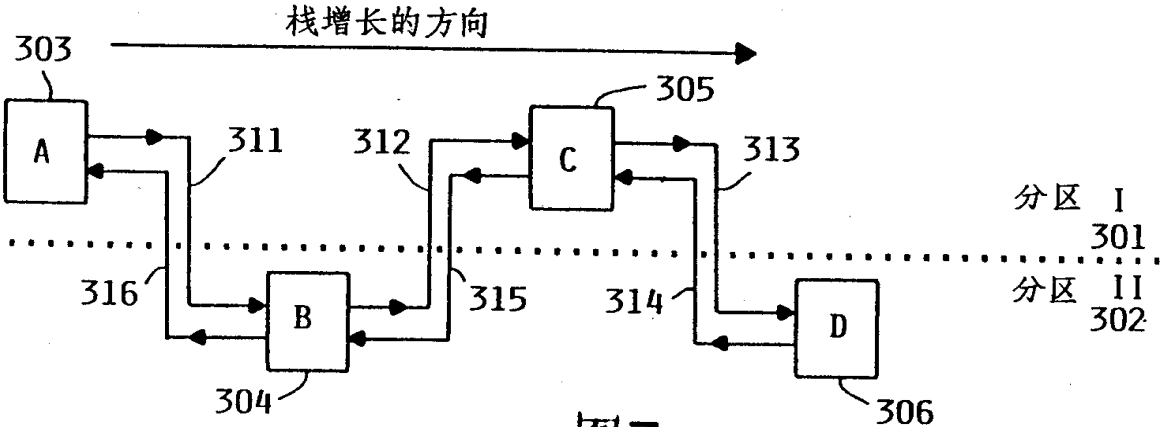

栈增长的方向
303
A
311
312
C
305
313
分区 I
301
分区 II
302
316
B
315
314
D
304
306

图3

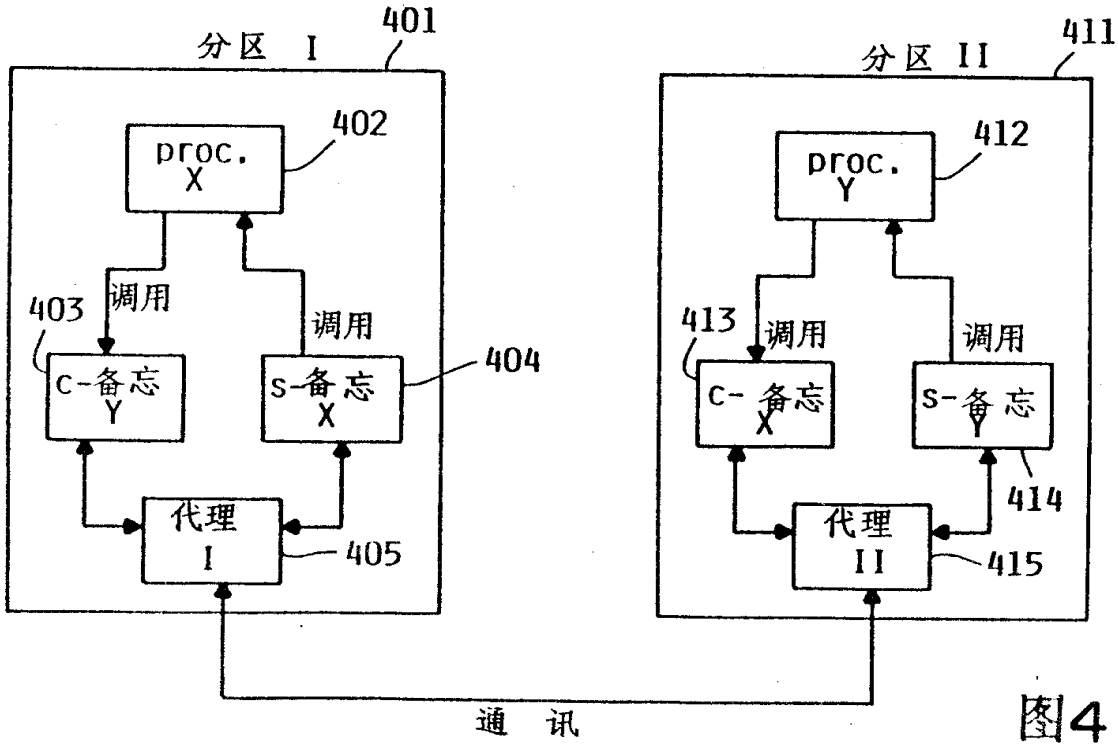

分区 I
401
proc.
X
402
调用
403
调用
C-备忘
Y
S-备忘
X
404
代理
I
405
分区 II
411
proc.
Y
412
调用
413
调用
C-备忘
X
S-备忘
Y
414
代理
II
415
通 讯

图4

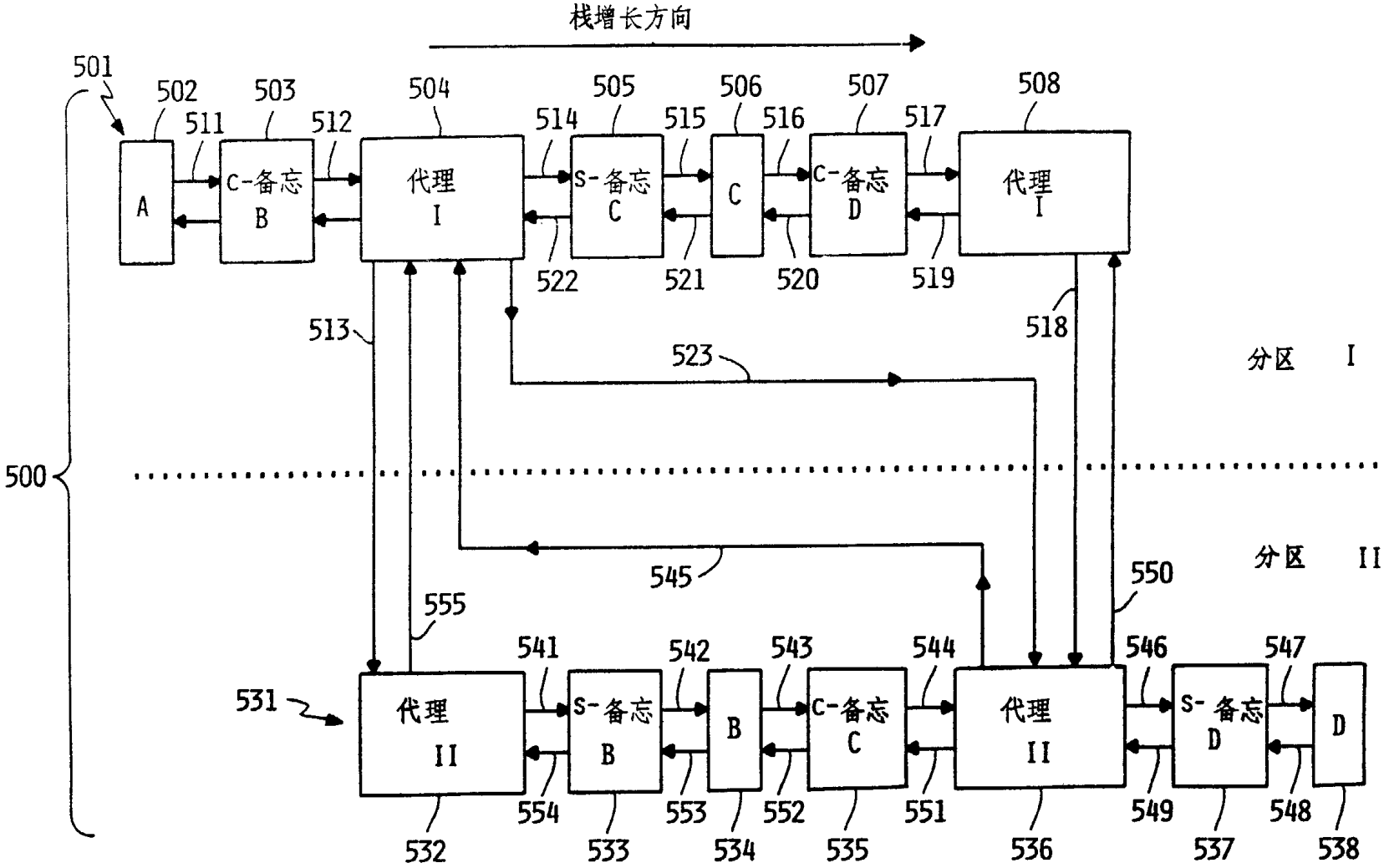

栈增长方向
501
502
503
504
505
506
507
508
511
512
514
515
516
517
A
C-备忘
B
代理
I
S-备忘
C
C
C-备忘
D
代理
I
522
521
520
519
518
513
523
分区 I
500
545
分区 II
550
555
541
542
543
544
546
547
531
代理
II
S-备忘
B
B
C-备忘
C
代理
II
S-备忘
D
D
554
553
552
551
549
548
532
533
534
535
536
537
538

图5

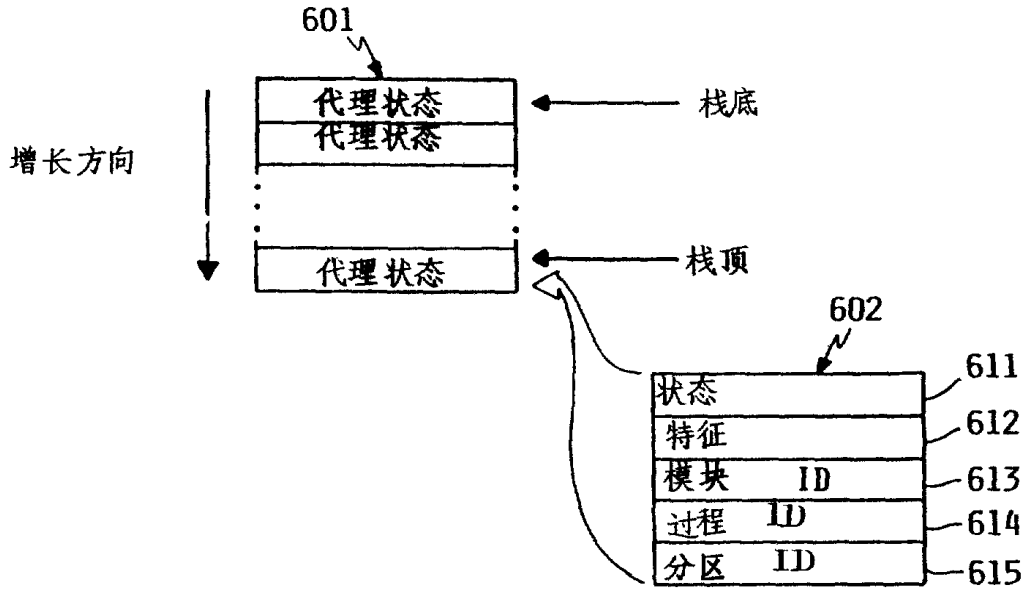

601
代理状态
代理状态
栈底
增长方向
代理状态
栈顶
602
状态
特征
模块 ID
过程 ID
分区 ID
611
612
613
614
615

图6

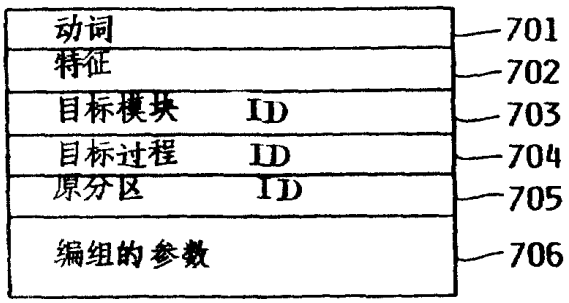

动词
特征
目标模块 ID
目标过程 ID
原分区 ID
编组的参数
701
702
703
704
705
706

图7

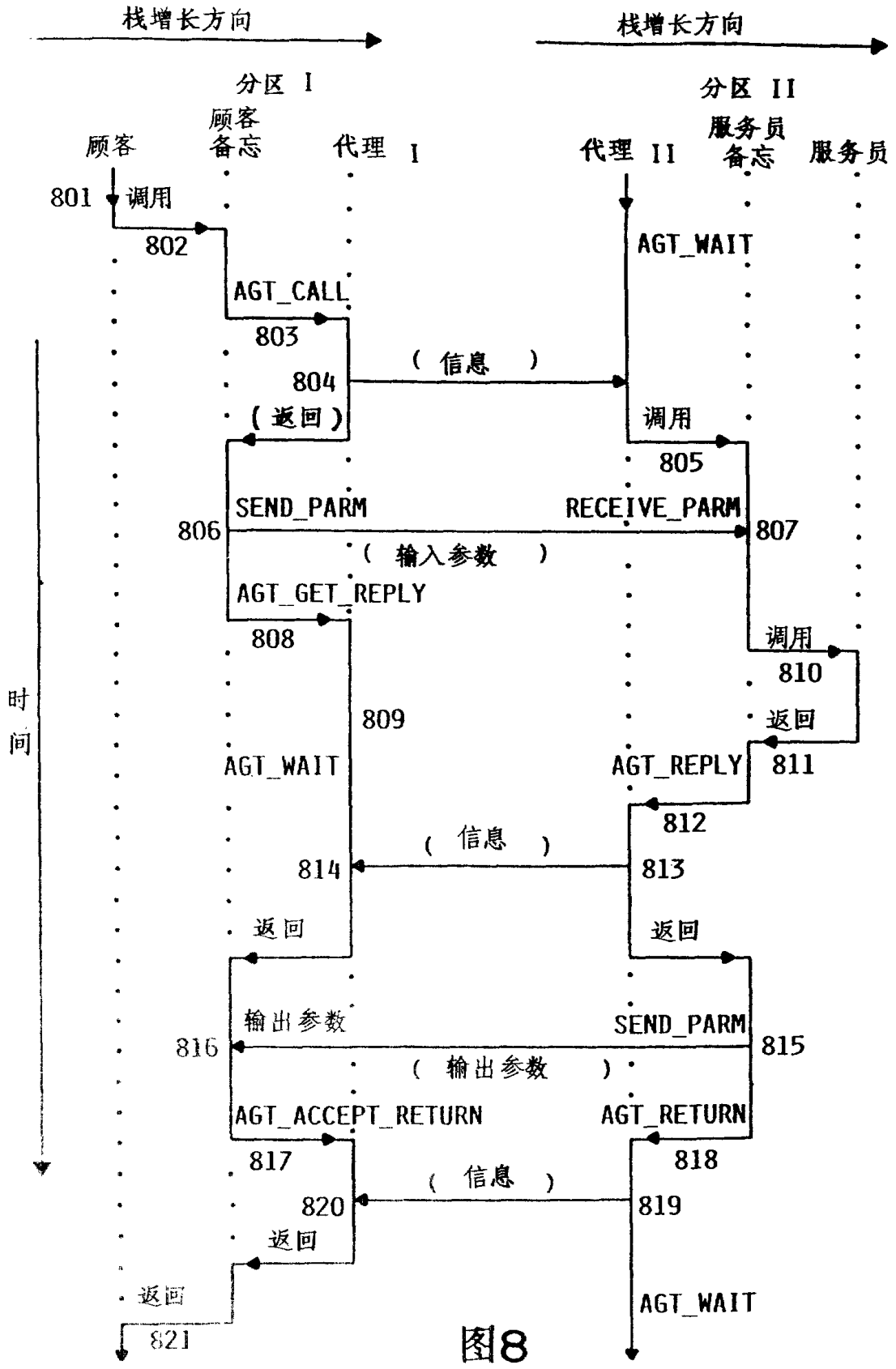

栈增长方向
栈增长方向
分区 I
分区 II
顾客
顾客备忘
代理 I
代理 II
服务员备忘
服务员
801
调用
802
AGT_CALL
803
(信息)
804
(返回)
AGT_WAIT
调用
805
SEND_PARM
RECEIVE_PARM
806
807
(输入参数)
AGT_GET_REPLY
808
调用
810
返回
809
AGT_WAIT
AGT_REPLY
811
812
(信息)
814
813
返回
返回
输出参数
SEND_PARM
816
815
(输出参数)
AGT_ACCEPT_RETURN
AGT_RETURN
817
818
(信息)
820
819
返回
返回
AGT_WAIT
821
时间

图8

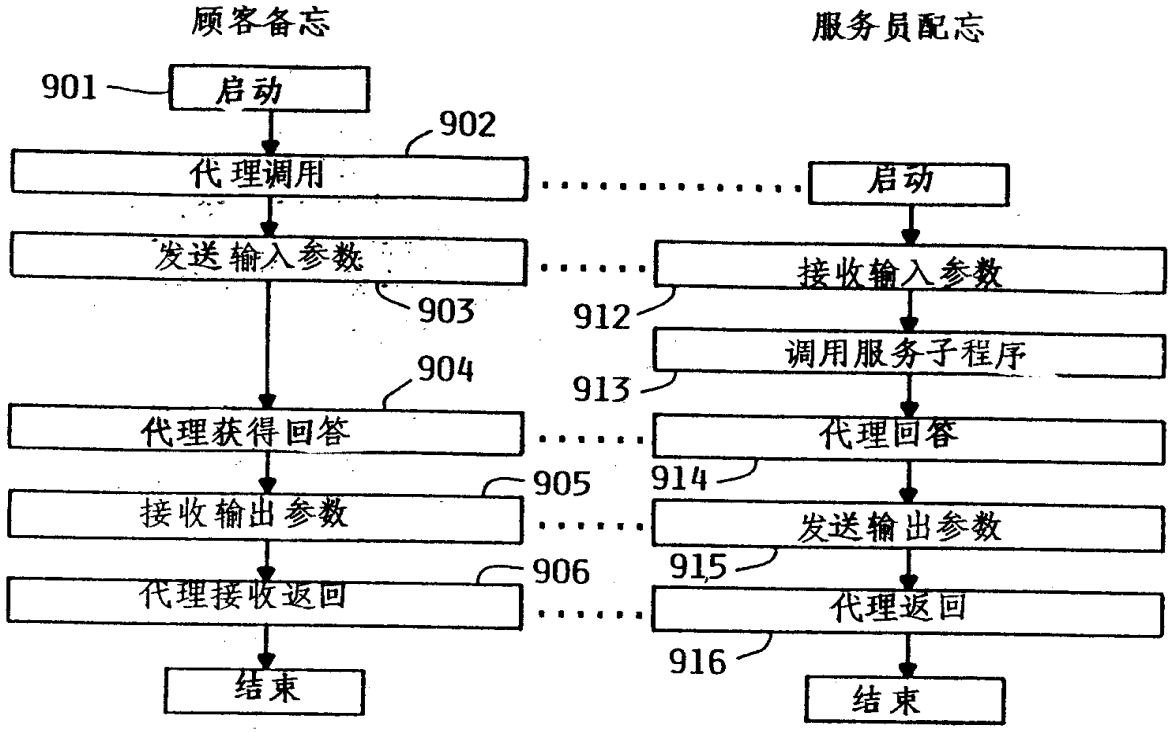

顾客备忘
服务员配忘
901
启动
902
代理调用
发送输入参数
903
904
代理获得回答
905
接收输出参数
906
代理接收返回
结束
启动
接收输入参数
912
调用服务子程序
913
代理回答
914
发送输出参数
915
代理返回
916
结束

图9

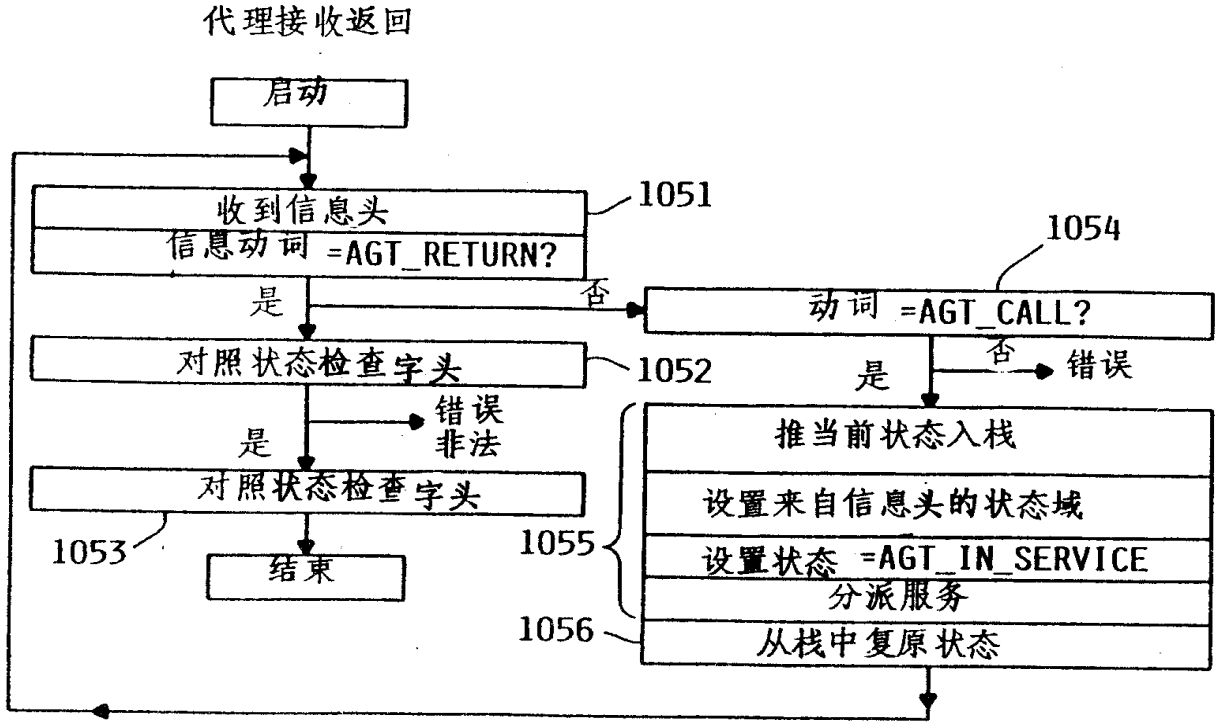

代理接收返回
启动
收到信息头
信息动词 =AGT_RETURN?
1051
是
否
对照状态检查字头
1052
是
错误
非法
对照状态检查字头
1053
结束
动词 =AGT_CALL?
1054
否
错误
是
推当前状态入栈
设置来自信息头的状态域
设置状态 =AGT_IN_SERVICE
分派服务
从栈中复原状态
1055
1056

图10F

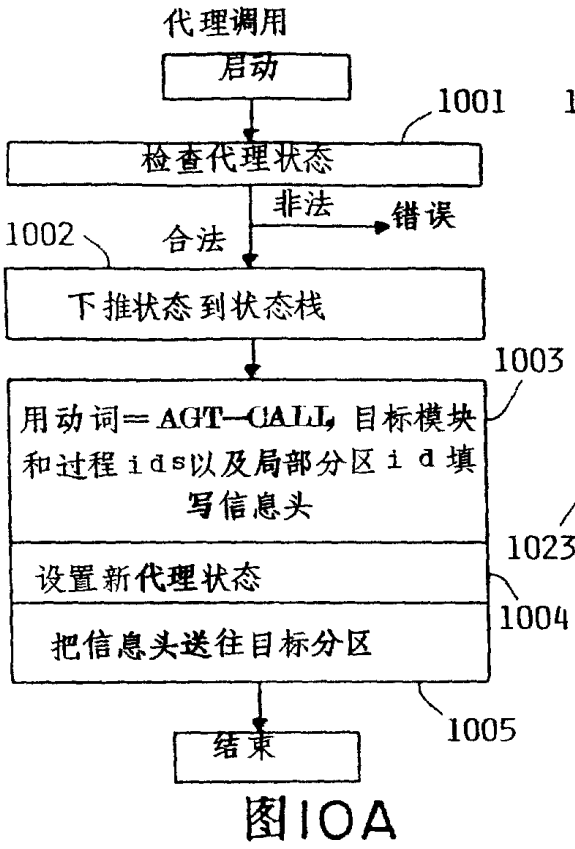


图10A

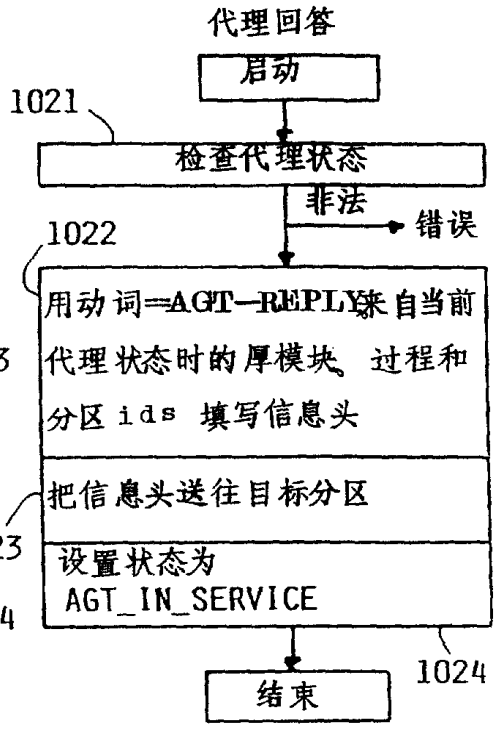


图10C

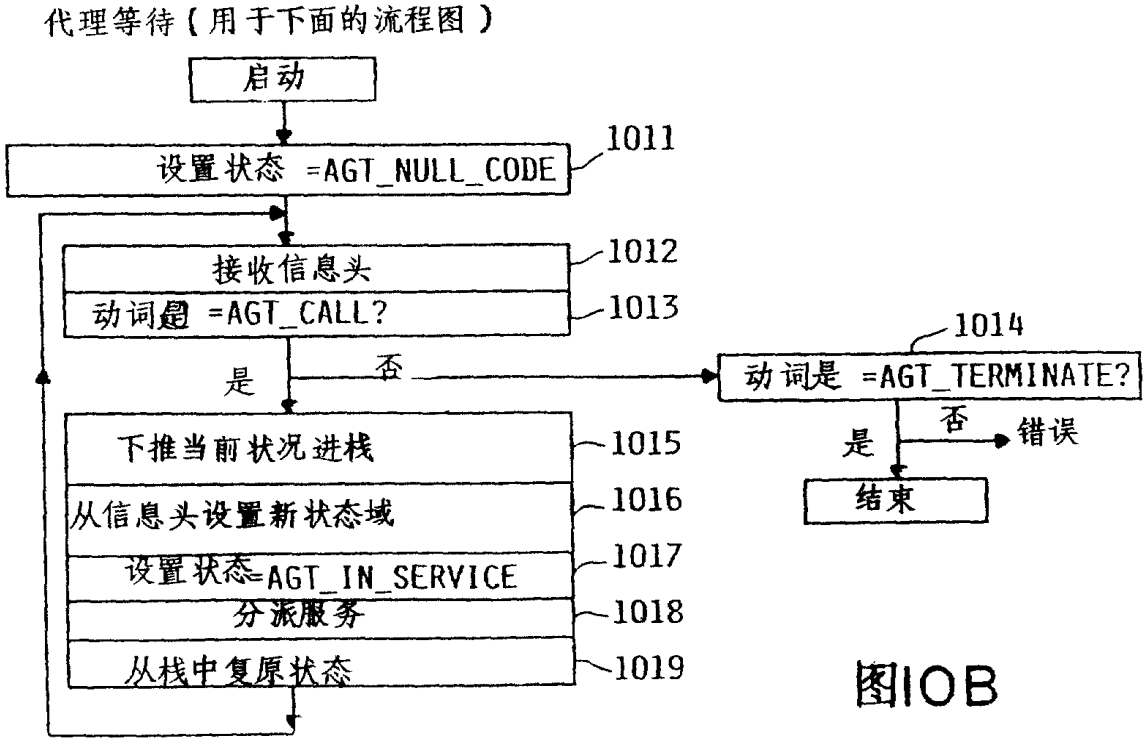


图10B

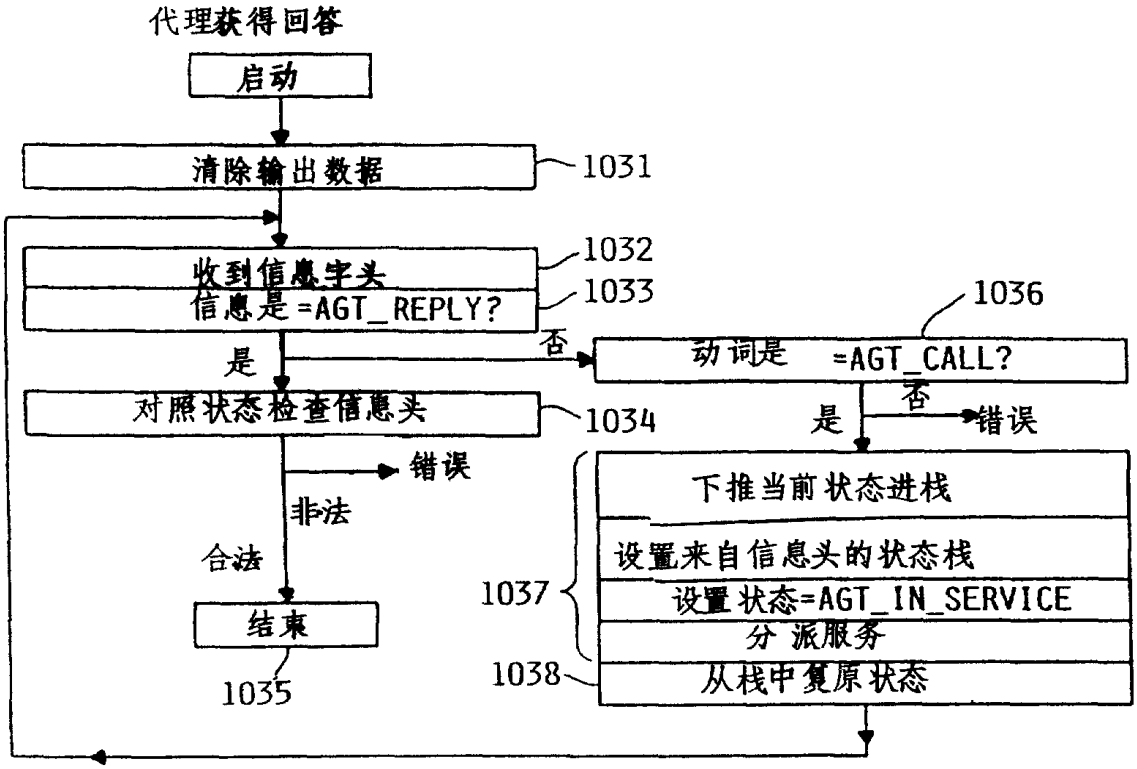

代理获得回答
启动
清除输出数据
1031
收到信息字头
信息是=AGT_REPLY?
1032
1033
是
否
对照状态检查信息头
1034
错误
非法
合法
结束
1035
动词是 =AGT_CALL?
1036
否
是
错误
下推当前状态进栈
设置来自信息头的状态栈
设置状态=AGT_IN_SERVICE
分派服务
从栈中复原状态
1037
1038

图10D

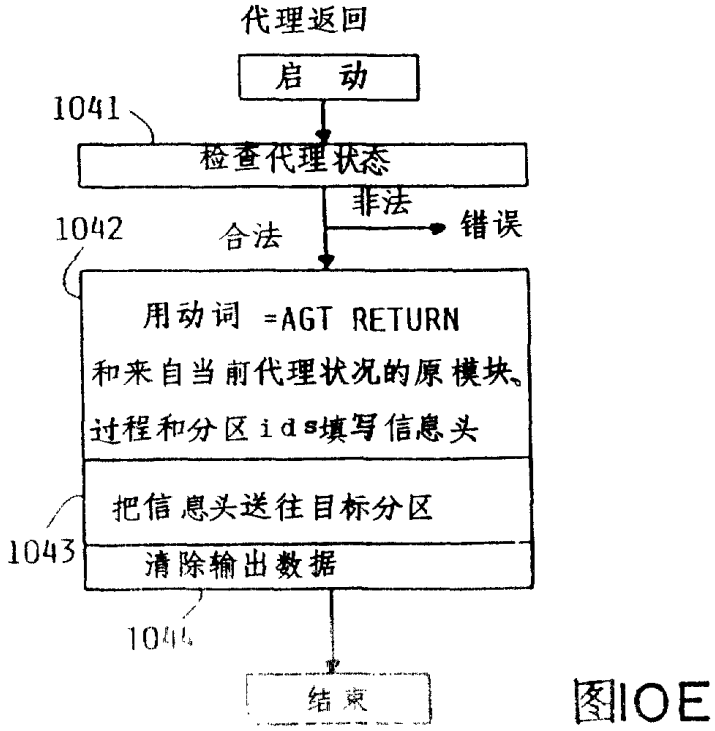

代理返回
启 动
1041
检查代理状态
非法
错误
1042
合法
用动词 =AGT RETURN
和来自当前代理状况的原模块、
过程和分区ids填写信息头
把信息头送往目标分区
1043
清除输出数据
1044
结束

图10E

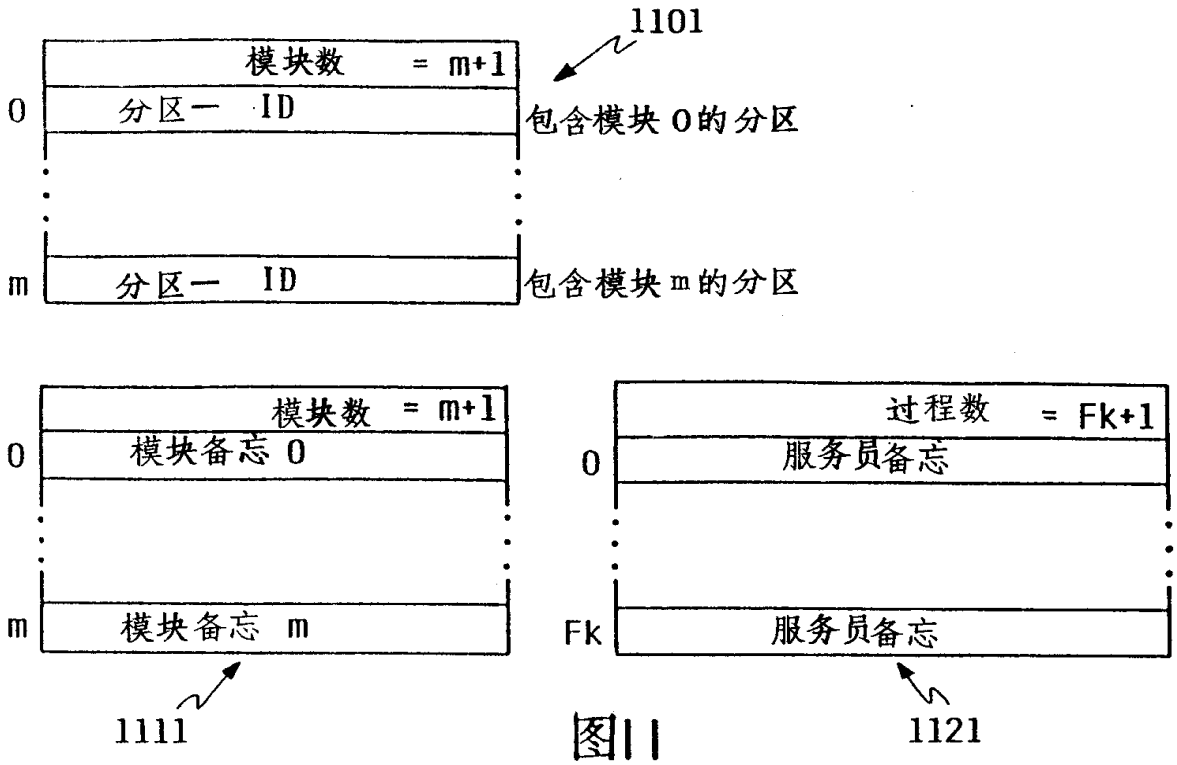
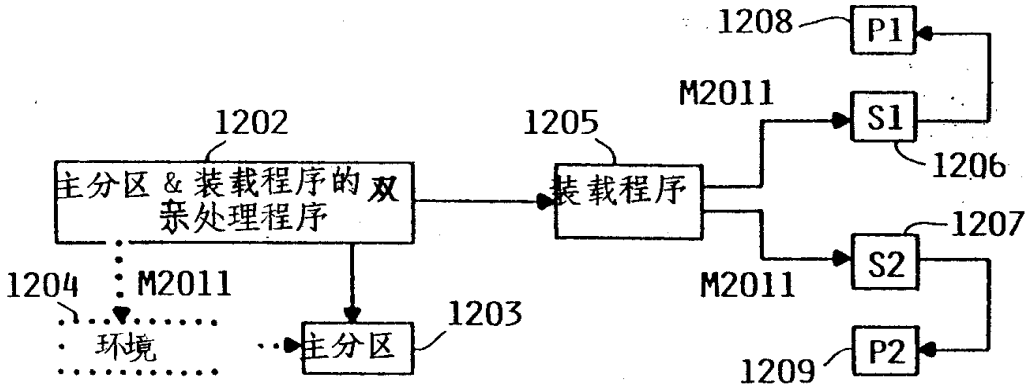
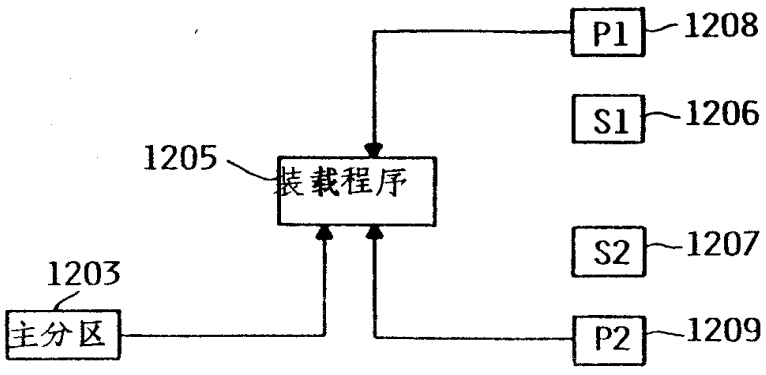


图11



(a) 开始收集周期



(b)结束收集周期

图12