



(12)发明专利

(10)授权公告号 CN 102467701 B

(45)授权公告日 2017.06.06

(21)申请号 201110355907.6

(51)Int.Cl.

G06F 9/54(2006.01)

(22)申请日 2011.11.11

(65)同一申请的已公布的文献号

申请公布号 CN 102467701 A

(43)申请公布日 2012.05.23

(56)对比文件

US 2006/0225075 A1, 2006.10.05,

US 6393458 B1, 2002.05.21,

US 2007/0033247 A1, 2007.02.08,

CN 101061471 A, 2007.10.24,

CN 1419675 A, 2003.05.21,

(30)优先权数据

12/945,084 2010.11.12 US

审查员 邱晓宁

(73)专利权人 甲骨文国际公司

地址 美国加利福尼亚

(72)发明人 R·阿达拉 A·辛格

S·赖伊辛加尼 Z·巴特

(74)专利代理机构 中国国际贸易促进委员会专

利商标事务所 11038

代理人 马景辉

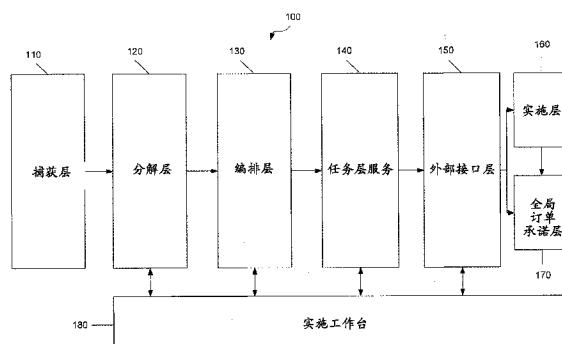
权利要求书3页 说明书31页 附图28页

(54)发明名称

分布式订单编排系统中的基于事件的编排

(57)摘要

本发明公开一种分布式订单编排系统，包括一个事件管理器，被配置为基于存储在数据库中的处理状态和元数据产生并发布一组事件。一组订购者可以消耗该组事件，并且每个订购者可以执行基于所消耗的事件的任务。



1. 一种编排系统,包括:

编排模块,被配置为启动事件管理器以使用编排处理来编排订单;以及

数据库,被配置为存储处理状态和元数据,

其中事件管理器被配置为:

定义编排处理的处理状态和元数据,并且把该处理状态和元数据存储在数据库的数据表中,所述处理状态包括指示编排处理的哪些步骤已经被执行的编排处理的状态以及表示订单的对象的每个首标、行和实施行的属性值,所述元数据包括编排处理的一个或多个步骤以及编排处理的该一个或多个步骤的次序;

基于由存储在数据库中的处理状态指示的已经执行的编排处理的步骤和由存储在数据库中的元数据指示的编排处理的该一个或多个步骤的次序来确定要执行的编排处理步骤;

基于所确定的编排处理步骤和元数据产生事件,其中该事件包括执行与编排处理步骤相对应的任务的指令和事件类型,所述任务包括多个子任务;以及

在事件消息发送系统上发布该事件,

其中任务层服务被配置为:

取回所发布的事件,

确定任务层服务是否订购所发布的事件的事件类型,以及

在任务层服务订购所述事件类型时:

执行所述多个子任务中的每一个;

生成发送给外部系统的至少一个任务消息;

处理来自所述外部系统的至少一个任务结果消息;以及

在执行每个子任务后更新数据库中存储的处理状态,

其中外部接口层被配置为:

经由网络把任务消息发送给所述外部系统;以及

经由网络接收来自所述外部系统的任务结果消息。

2. 如权利要求1所述的编排系统,其中事件管理器和任务层服务分别是所述编排系统的不同组成部分。

3. 如权利要求2所述的编排系统,

其中事件管理器包括由所述编排系统调用的无状态处理。

4. 如权利要求1所述的编排系统,

其中中介器取回所述事件;

其中中介器基于包含在所述事件中的参数调用任务层服务;

其中任务层服务基于所发布的事件执行任务;以及

其中任务层服务在执行任务之后更新存储在数据库中的处理状态。

5. 如权利要求1所述的编排系统,其中所述事件消息发送系统包括一个或多个消息发送通道,每个消息发送通道被配置为把事件从事件管理器发送到任务层服务。

6. 如权利要求1所述的编排系统,

其中所述事件管理器还被如下配置以确定要产生的事件:

确定是否有任何的编排处理步骤剩余;

选择处理步骤；  
确定所选择的处理步骤是否依赖于其他的处理步骤；以及  
当确定出所选择的处理步骤依赖于其他的处理步骤时，确定是否所述其他的处理步骤已完成。

7. 如权利要求1所述的编排系统，其中所述元数据用于定义订单内的任务以及用于调用与任务相关的服务。
8. 如权利要求7所述的编排系统，其中所述与任务相关的服务包括任务层服务。
9. 如权利要求1所述的编排系统，其中重复所述启动直到处理状态达到退出状态。
10. 如权利要求1所述的编排系统，其中所述启动还包括以下至少一个：发布用于编排处理的并行步骤的事件、分割编排处理、以及处理订单的更改请求。
11. 如权利要求1所述的编排系统，其中所述启动还包括：发布用于编排处理的条件步骤的事件。
12. 一种编排系统的计算机实现的方法，所述编排系统用于启动事件管理器以使用编排处理来编排订单，该计算机实现的方法包括：

在事件管理器处：  
经由网络接收来自客户计算机的网络订单；  
定义编排处理的处理状态和元数据，并且把该处理状态和元数据存储在数据库的数据表中，所述处理状态包括指示编排处理的哪些步骤已经被执行的编排处理的状态和表示订单的对象的每个首标、行和实施行的属性值，所述元数据包括编排处理的一个或多个步骤以及编排处理的该一个或多个步骤的次序；  
基于由存储在数据库中的处理状态指示的已经执行的编排处理的步骤和由存储在数据库中的元数据指示的编排处理的该一个或多个步骤的次序来确定要执行的编排处理步骤；

基于所确定的编排处理步骤和元数据产生事件，其中该事件包括执行与编排处理步骤相对应的任务的指令和事件类型，所述任务包括多个子任务；以及

在事件消息发送系统上发布该事件；

在任务层服务处：

取回所发布的事件，

确定任务层服务是否订购所发布的事件的事件类型，以及

在任务层服务订购所述事件类型时：

执行所述多个子任务中的每一个；

生成发送给外部系统的至少一个任务消息；

处理来自所述外部系统的至少一个任务结果消息；以及

在执行每个子任务后更新数据库中存储的处理状态，在外部接口层处：

经由网络把任务消息发送给所述外部系统；以及

经由网络接收来自所述外部系统的任务结果消息。

13. 如权利要求12所述的计算机实现的方法，其中所述事件管理器和任务层服务分别是所述编排系统的不同组成部分。

14. 如权利要求13所述的计算机实现的方法，

其中所述事件管理器包括由所述编排系统调用的无状态处理。

## 分布式订单编排系统中的基于事件的编排

### 技术领域

[0001] 本发明一般涉及一种计算机系统,特别是涉及一种用于业务处理编排的计算机系统。

### 背景技术

[0002] 订单管理系统是一种计算机软件和/或硬件系统,其由多种产业来实施从而有助于订单的输入和处理。公司,例如库存公司和那些使用电子商务的公司,使用订单管理系统来接收、处理以及履行客户的订单。订单管理系统可以实现通过网站购物服务或数据输入系统来输入订单。该系统通常为每个订单捕获客户专属信息和/或账户级别信息。然后执行信用验证或支付处理来检查资金的可用性并确认交易。对有效订单执行仓储履行,包括挑选、包装并发出所订货物或服务。

[0003] 业务处理通常由业务架构设计师/分析师进行建模。业务处理可以模拟在网络服务环境下与不同的系统进行交换的消息。然后业务架构设计师/分析师将该模型提供给信息工程(“IT”)设计师。IT设计师使用编排语言,例如业务处理执行语言(“BPEL”),对业务处理进行编码。BPEL处理是在BPEL编辑器中创建的,并且调用所部署的BPEL处理。因为IT设计师和业务架构设计师/分析师通常具有不同的技能组合(即业务架构设计师/分析师熟悉所建模的业务处理,而IT设计师熟悉编排语言而不是业务处理),结果IT设计师所开发的BPEL处理可能不像业务架构设计师/分析师想象的那样工作。结果,在原始构想的业务处理模型和所实施的模型之间存在着较大分歧。

[0004] 业务处理还存储着所有相关的数据直至业务处理被关闭。当业务处理长时间运行时,所有相关数据的存储就成为试图提高吞吐量时的瓶颈。另外,网络服务环境下利用不同系统来使用业务处理会导致不同系统的紧密耦合。

### 发明内容

[0005] 本发明的一个实施例涉及一种计算机可读介质,其具有存储于其上的指令,当被处理器所执行时,使处理器启动事件管理器,以使用处理来处理订单。指令包括:基于存储在数据库中的处理状态和元数据来确定要执行的处理步骤。指令还包括:基于所确定的处理步骤产生一个事件,该事件包括执行与该处理步骤相对应的任务的指令。指令还包括:在一个事件消息发送系统上发布该事件。

### 附图说明

[0006] 在下面结合附图对优选实施例的详细描述中,本发明的更进一步的实施例、细节、优点和改变将会变得清晰。

[0007] 图1示出了根据一个实施例的分布式订单编排系统的一个实例。

[0008] 图2示出了根据一个实施例的用于处理订单的流程图。

[0009] 图3示出了根据一个实施例的在订单实施的上下文中,用于提供编排处理设计和

编写环境的系统的一个实例。

- [0010] 图4示出了根据一个实施例的接口的实例。
- [0011] 图5示出了根据一个实施例的运行时间操作。
- [0012] 图6示出了根据一个实施例的使用流程序列器调用服务的一个实例。
- [0013] 图7示出了根据一个实施例的在不同的层中用于编排数据流的处理。
- [0014] 图8示出了根据一个实施例的用于修改一个可执行处理的方法的流程图。
- [0015] 图9示出了可实现本发明一个实施例的系统的方框图。
- [0016] 图10示出了根据一个实施例的分布式订单编排系统。
- [0017] 图11示出了根据一个实施例的用于处理一个修改请求的方法的流程图。
- [0018] 图12示出了根据一个实施例的修改请求流程的一个实例。
- [0019] 图13示出了根据一个实施例的可执行处理的一个实例。
- [0020] 图14示出了根据一个实施例的用于调整原始的可执行处理的各步骤的新的可执行处理的一个实例。
- [0021] 图15示出了根据一个实施例的原始的可执行处理和根据所接收的修改请求的新的可执行处理的流程图。
- [0022] 图16示出了根据一个实施例的依据使用事件管理器的处理来发布一组事件的方法的流程图。
- [0023] 图17示出了根据一个实施例的使用事件管理器和任务层服务来编排订单的方法的流程图。
- [0024] 图18示出了根据一个实施例的使用事件管理器、任务层服务和中介器来编排订单的方法的流程图。
- [0025] 图19示出了根据一个实施例的在订单实现的上下文中使用事件管理器来提供编排的系统的一个实例。
- [0026] 图20示出了根据一个实施例的采用事件管理器的分布式订单编排系统。
- [0027] 图21示出了根据一个实施例的事件消息发送系统的一个实例。
- [0028] 图22示出了根据一个实施例的事件消息发送系统的发布订购消息通道的一个实例。
- [0029] 图23示出了根据一个实施例的在分布式订单编排系统中管理事件的方法的流程图。
- [0030] 图24示出了根据一个实施例的确定待发布的一组事件的方法的流程图。
- [0031] 图25示出了根据一个实施例的事件管理器发布用于并行处理步骤的事件的一个实例。
- [0032] 图26示出了根据一个实施例的事件管理器分割处理的一个实例。
- [0033] 图27示出了根据一个实施例的事件管理器使用事件来处理修改请求的一个实例。
- [0034] 图28示出了根据一个实施例的分布式订单编排模块的功能性的流程图。

## 具体实施方式

- [0035] 本发明的一个实施例涉及一种分布式订单编排系统(“D0O”)。分布式订单编排提供了一种灵活的、可配置的基础架构，其可被容易地定制以便符合企业的商务实践以及现

有的订单实施系统的体系结构。分解是指把从一个或多个订单捕获模块接收的数据转换为内部规范格式以便处理该数据的过程。在一个示范实施例中，分布式订单编排系统包括：捕获层，用于在多个通道上捕获客户订单；分解层，用于帮助确定编排处理；编排层，用于执行并编排订单行处理；任务层服务，用于执行与任务相关的逻辑；外部接口层，用于连接外部系统；实施层；以及全局订单承诺层，用于为用户提供界面进行调度和提供资源。分布式订单编排系统可以进一步地包括一个实施工作台层，其连接系统的其它层并管理资源、任务和分配。对以上描述的分布式订单编排系统的各个层进行组合，从而提供一个降低了实施和维护成本的完整的订单管理解决方案。然而，在一个可选的实施例中，捕获层不是分布式订单编排系统的一部分。在该可选的实施例中，捕获层是一个单独的系统的一部分，使用一个连接器服务来作为分布式订单编排系统和该捕获层之间的桥。

[0036] 图1示出了根据一个实施例的分布式订单编排系统100的一个实例。在该实施例中，分布式订单编排系统100包括捕获层110，其能够在多个通道上接收并捕获与客户对货物和/或服务的订单有关的信息。该订单可以通过图形用户界面来接收，例如网站购物车，或者能够通过任意数据输入系统来接收。捕获层110捕获并向分解层120发送订单信息。然而，在一个可选实施例中，捕获层110与分布式订单编排系统100是分离的。在该可选的实施例中，使用一个连接器服务来作为分布式订单编排系统100和该捕获层110之间的桥。

[0037] 订单捕获系统捕获的订单具有任意必要的用于处理订单的功能属性，例如定价、验证、合格等。销售订单以源订单对象的形式被馈送到分解层120。源订单对象是从不同的捕获系统所提交的销售订单对象中产生的。源订单对象具有馈送到分解层120的通用的格式。

[0038] 分解层120接收订单信息并将所接收的订单拆分为单个的购买定单，以被发送到实施系统以及供应商来执行。分解层120可以包括分解规则工作台，用于建立规则、规则集合以及用于订单捕获层110的转换处理，它们可以从销售角度捕获订单。例如在全球范围内销售一种膝上电脑。该膝上电脑包括一根电源线，但是客户只购买了膝上电脑（订单上的一行）。也就是说，销售网站可能会只销售该膝上电脑而不会再让客户单独订购电源线。然而，从实施角度看，膝上电脑和电源线需要从订单中取回。在分解层120中，存在着这样的业务规则，即膝上电脑必须具有电源线并且电源线上的插头必须与膝上电脑的运输目的地国家相匹配。因此当分解模块120完成时，订单就具有了两行，一行是膝上电脑，另一行是合适的电源线。这样订单就被分解为多个待实施的项目。

[0039] 而且，分解层120可以取得所接收的订单并将其翻译成分布式订单编排系统100的其它层，例如实施层160，所需的订单格式以及订单内容。因为捕获层110出于兼容性目的，能够在不同类型的系统上接收任意格式的订单，因此分解层120就需要将订单翻译成与该分布式订单编排系统100所有的其它层和/或处理相兼容并能够被它们识别的格式。另外，分解层120可以提供一种处理启动能力，用来基于适当的分解规则分配并启动订单的编排处理。例如，基于订单中的参数分配不同的编排处理。例如，公司可以按照实施或运输的速度对某些客户给予特殊待遇。例如，金卡客户可以给予加快运输。而且，也可以提供与客户沟通的附加步骤。当接收到这些客户的订单时，它们就被分配给具有这些参数和步骤的编排处理，而其它客户的订单就被分配给标准处理。

[0040] 分解可以使用规范对象模型来完成与订单捕获系统的数据格式的解耦合。分解整

合处理工作在称作企业业务对象 (EBO's) 的一组通用数据结构上。它们是基于规范数据模型的。这种方法允许 D00 不不必知晓所参与的应用程序并且独立于源或目标应用程序。该模型消除了使来自不同的应用程序的数据互相直接映射的需要。

[0041] 如图1所示,分布式订单编排系统100进一步包括一个编排层130。编排层130提供了单独的编排处理用于管理订单和/或服务行项目。例如,编排层130可以提供业务处理管理功能,以便支持处理中的步骤的调度,包括步骤的持续时间和完成日期的计算或重新计算。编排层130还可以提供外部任务执行功能,以便支持外部任务的创建、更新、释放和监控。外部任务是由实施系统来执行的任务。任务层服务进行特殊处理然后将数据发送到那些集成的实施系统。编排是任务层服务的调用的顺序。

[0042] 编排层130还可以提供风险管理,以把订单的承诺的交货日期与当前的完成订单的估计日期进行核对,映射到用户定义的阈值,以及处理并逐步提高条件。编排层130可以进一步提供用于修改订单的处理,包括一个支持处理,用以退回到提供修改订单自动操作的阶段并对订单捕获阶段所修改的订单更改实时编排处理。而且,可以通过实例化该编排处理来提供预计的订单完成日期。编排层130还提供一种机制用于自动地或基于用户请求来更新订单状态。

[0043] 一个实施例提供了一种工具,用来在订单实施业务处理中为业务处理的模型化提供高度抽象。业务处理可以由用户例如业务分析师来模型化,并且不需要任何IT设计师来编写代码以执行业务处理。在中心位置定义业务处理为用户提供了灵活性,其中该中心位置被配置为输入并捕获编排并实施订单所需的所有信息。这种中心位置的一个实例可以是基于网络的管理用户界面。同样地,编排和订单实施所需的所有信息的一个实例可以是业务调度、核心编排和修改管理所需的信息。业务处理可以确认定义了要在订单实施处理中执行的步骤的一个或多个服务。运行时间引擎然后使用该定义并基于业务处理的定义来动态地调用这些服务。

[0044] 在业务环境下,业务用户通常会面对模型设计师而不是IT人员。通过提供一种基于网络的管理环境,业务用户就可以设计业务处理了。处理的定义可以用业务术语而不是IT术语来定义。特定的实施例允许管理环境脱离代码编辑器例如BPEL编辑器之外,用来使用相关服务来定义这些处理。用户可以将这些能够在运行时间上执行的处理配置为可执行处理,而不需要IT的介入。这就减少了每次部署这些处理就要修改业务处理的需要。用户在数据表上设置服务的序列。然后模型化的业务处理被用于执行一个可执行处理(也被识别为编排处理),其可以在运行时间上被安装及执行。在一个实施例中,“运行时间”可被定义为当接收订单以便进行处理的时间。元数据被安装在数据运行时间表中,并用于定义业务处理的可执行处理。元数据用于在可执行处理中调用服务。

[0045] 在一个实施例中,所调用的服务被封装并且是可重复使用的。元数据用于确定怎样和何时调用服务。而且,依靠元数据,产生输入变量并发送给各个服务以便调用被封装的服务。使用公共签名来发送数据从而调用服务。不同的输入变量可以针对不同的可执行处理中的不同的服务来公式化。输入变量以相同方式来格式化,以便使服务能够读取不同的数据组并调用服务。这样,服务就可以在不同的业务处理中重复使用,而不需重新编码及重新部署。服务的部署表示该处理准备好发布以进行测试或生产。

[0046] 分布式订单编排系统100可以进一步包括任务层服务140,以提供用于为每个编排

处理阶段控制逻辑的封装服务。具体地说，任务层服务140可以提供任务专用业务逻辑，使该逻辑围绕某个请求，以便使系统100知晓何种逻辑任务与特定请求相关。需要在编排中的可执行处理中执行的步骤可以请求要被执行的任务。例如，任务层服务140能够提供并控制处理逻辑，以安排运输、请求运输、更新安装平台、创建活动等等。任务层服务140的输出是标准的货物和/或一个或多个服务请求，它们可被提供给系统100的其它层，例如外部接口层150或实施层160。另外，任务层服务140可以接收用于更新处理逻辑或状态的输入。

[0047] 任务层服务140启动该任务，产生用于外部系统的消息并发送该消息。产生执行该任务所需的数据结构。某些任务可以在任务层服务中预先定义。而且，客户可是使用定义了如何创建一项任务的模板来增加其它任务。所产生的消息指示了哪个任务应当由外部系统来执行。待执行的任务是已被模型化的订单处理的体现。例如，任务可以是为订单开账单。还可以包括用于执行任务的参数。向外部接口层150的外部接口发送该消息。任务层服务140转换该消息并将其发送给外部接口层。

[0048] 分布式订单编排系统100还包括外部接口层150，用于翻译标准请求并将请求路由至外部系统以便处理。更特别地，外部接口层150可以接收从任务层服务140输出的标准货物和/或服务的一个或多个请求，并且如果需要的话就就提供请求的单层转换以匹配实施系统的格式。外部接口层150所执行的转换将数据映射到集成实施系统所需的内容和格式。分解层120的转换是将数据转化为系统100使用的内部格式。外部接口层150可以将来自任务层服务140的数据结构映射到外部格式。外部接口层150提供灵活的路由以便使这些请求基于业务规则路由到具体的实施系统。例如，如果多于一个运输系统是可用于实施的，那么业务规则将确定各个运输请求要被发送到哪个运输系统。外部接口层150还可以包括转换规则工作台，该转换规则工作台能够用于建立规则、规则集合以及用于外部路由该请求的转换数据。

[0049] 任务层所产生的消息可以是通用格式的。然而不同的外部系统可以使用其他的格式来通信。外部接口层确定外部系统所使用的格式并转换该消息。例如，用户定义的元数据可用于确定要使用哪种格式。在一个实例中，对哪些外部系统调用了所订购的产品的映射，被用来翻译该消息。

[0050] 外部系统可以是执行与处理订单相关的任务的系统，例如可以是调度系统、运输系统等等。当执行任务时，确定任务的结果。结果可以是调度运输时的日期，运输货物的日期等等。然后结果被送回外部接口层150。

[0051] 分布式订单编排系统100可以进一步包括全局订单承诺层170，该全局订单承诺层170提供用于调度订单并为订单提供资源的接口，例如图形用户接口。特别是在一个实施例中，全局订单承诺层170包括一个来源经纪人，用于基于客户概要、订单和供应商定义等确定与订单相关的产品和服务的最佳来源。而且，全局订单承诺层170在分布式内部系统上提供实时的库存储备和未储备以及库存的检查。全局订单承诺层170的接口允许浏览可用性和来源信息，以便使用户能够浏览其可用性并手动修改实施货物或服务订单的来源。然而，在一个可选实施例中，全局订单承诺层170是与分布式订单编排系统100分离的。在该可选实施例中，使用一个连接器服务来作为分布式订单编排系统100和全局订单承诺层170之间的桥。

[0052] 实施工作台180可以作为订单实施管理员、用户和监管员的用户接口,用来监控并管理订单在系统100内的流动。在一个实施例中,实施工作台180向用户(例如监管员)提供监控订单实施任务的机制,包括允许监管员监控活动的加载并生成报告。实施工作台180还提供了实施处理分析,其允许业务处理设计者分析处理的度量标准,例如手动干预的数量、发生错误的数量和种类、超期订单的数量以及期望处理周期与实际周期的对比。在某些实施例中,实施系统性能的分析能力也包含在实施工作台180中,用以提供报告和仪表板使订单管理者能浏览每个系统的订单并分析性能。实施工作台可以使用图形表示法(例如图表和曲线)来向用户清晰地传达系统状态/订单信息。由于D00系统100具有数据参考数据,因此可以画出合计图表/曲线用于趋势分析。如下所述,用户可以从实施工作台采取动作,例如替换所订购的项目、将数量分割为多个订单行、保持订单行以防止继续发展等。

[0053] 根据一个实施例,实施工作台180允许用户做出与实施相关的大量订单信息的修改,包括对实施信息(例如日期等)进行一行或多行的修改。实施工作台180进一步允许对编排处理的监控,例如回顾编排处理的状态,包括整个处理的进展,以及各个任务的状态和相应的实施行和人物行。在一个实施例中,实施工作台180包括维护订单实施处理的机制,并允许订单处理用户控制与订单相关的处理,包括暂停、编辑、取消等等。

[0054] 在某些实施例中,实施工作台180还提供用于订单和任务分配的功能。例如,实施工作台180可以使用分配引擎来将订单和活动分配给合适的实施资源。实施工作台180可以包括批量对订单进行再分配的机制,从而允许监管员从一个实施系统到另一个实施系统重新为一组订单提供资源。实施工作台180还提供供应率的分配和延期交货的规则,它们能够自动确定如何控制短缺状况。通用收件箱可以包含在实施工作台180中,用以允许用户浏览分配给他们的活动并对任务作出响应。

[0055] 实施工作台180允许用户浏览正在系统100的不同层中处理的订单。对订单状态的浏览可以从任何一个处理该订单的层中产生。这是因为已经提供了一个端到端集成系统因。传统的订单系统已经将解决方案定制成不允许浏览不同层。通过以一个允许通用通信的格式集成各个层,就能提供一个可以从所有层中产生浏览的用户接口。

[0056] 分布式订单编排系统100的实例还可以包括一个实施层160。在一个实施例中,实施层160可以是面向外部实施系统的接口,并能够用于将订单信息和实施需求传输给与订单相关的货物和/或服务的供应商或来源。

[0057] 分布式订单编排系统100的某些实施例包括一个管理用户接口。管理用户接口提供了对终端用户隐藏实施执行环境的复杂性的管理服务。例如,该管理用户接口通过一个管理环境提供产品映射,该管理环境定义了一些转换方式,以用于在销售视图和供应商系统定义之间进行产品结构的映射。在该实施例中,销售视图指的是在下购买订单时提供给客户的简单视图。供应商系统定义指的是由货物和/或服务供应商所使用的更具体和详细的信息。管理用户接口还可以提供一个编排处理工作台用于建立用于订单编排的处理、规则集合和参数。管理用户接口具有一个集成的设置,包括处理顺序、计划、风险、修改管理和工作台显示。管理用户接口还允许用于任务、处理和实施行的用户定义状态的转变,以及用于配置约束、转换规则和路由规则的业务规则配置。

[0058] 图2描述了根据一个实施例的用于处理订单的简化流程图200。在步骤202中,分解层120接收一个订单。在步骤204中,分解层120确定用于实施订单的一个或多个编排处理。

例如,订单可以被分解成必须要获得的项目或必须要执行的服务。每个项目或服务可以具有其自身的编排服务。

[0059] 在步骤206中,编排层130产生可执行处理用以编排这些编排服务的实施。可执行处理可以具有多个必须对每个编排服务执行的步骤。

[0060] 在步骤208中,任务层服务140对执行该可执行处理的各步骤所需的业务功能进行控制。为可执行处理来执行的任务可以包括建立具有信息和参数的数据结构,这些信息和参数是使外部系统执行该任务所必须的。该数据结构可以按照系统100的内部格式。例如,该任务可以是为一个订单开账单。并且还包括用于执行该任务的参数。

[0061] 在步骤210中,外部接口层对任务进行翻译并将其路由到外部系统。然而,不同的外部系统可以使用其他的格式进行通信。外部接口层确定外部系统所使用的格式并对消息进行转换。例如,用户定义的元数据可以用于确定要使用的格式。在一个实例中,对哪些外部系统调用了所订购的产品的映射,被用来翻译该消息。

[0062] 在步骤212中,外部接口层150接收来自外部系统的、关于任务处理的结果。当执行任务时,就确定该任务的结果。结果可以是计划发货日期,发货日期等等。

[0063] 在步骤214中,外部接口层150转换消息并将其发送到任务层服务140。在步骤216中,编排层基于该结果更新关于任务的信息。例如,结果可以存储在表或数据库中。然后该处理前进到下一个要被调用的服务。

[0064] 下面将参照图3-8以及根据使用流程序列器进行编排的实施例,来描述更多实现编排的细节。然而,本领域技术人员将会容易地理解,更多的细节只是编排的一个实例,这种编排还可以采用许多不同的实施方式来实现,包括不使用流程序列器的可选实施例。例如,可以根据美国专利申请号12/697,756、名为“使用模板的业务处理的编排”中描述的细节来实现编排。

[0065] 图3示出了根据一个实施例的在订单实施的上下文中,用于提供编排处理设计和编写环境的系统300的一个实例。在该实施例中,系统300包括一个编排系统302和客户机304。尽管只提供了编排系统302和客户机304的单个实例,但可以理解,还可以使用多个编排系统和客户机的多个实例。而且,编排系统302和客户机304可以是分布式计算系统的一部分。也就是,所描述的功能可以分布于各种计算设备中。

[0066] 客户机304可以是一个计算设备或一组计算设备,它们被配置为允许业务处理模型化。编排系统302编排用于业务处理的可执行处理310的服务的调用和运行。如所描述的那样,编排是要在业务处理中执行的服务的协调和调用。

[0067] 如所使用的,业务处理可以由用户来模型化。业务处理是对要执行的步骤的定义。这些步骤定义在接口308中。可执行处理是由运行时间引擎312执行的处理。该可执行处理包括要被执行以协调服务的代码。

[0068] 服务库306包括多个可包含在业务处理中的服务。在一个实施例中,服务库306包括可在订单实施业务处理中执行的服务。订单实施包括被执行用以实施订单的处理。例如,可以从订单捕获模块中接收订单。订单可以是用于货物、服务等。可以执行不同的服务来实施订单,例如运输、安装、货品计价等。订单实施处理可以根据这些不同的服务来特性化。期望对于任一个订单,这些处理中的某些或全部要被执行用以实施订单。因此,特定的实施例针对那些期望在订单实施处理中执行的服务而创建服务。

[0069] 服务可以是非可配置单元和可配置单元。非可配置单元是建立并提供给客户的服务。非可配置单元是可被用在订单实施处理中的单元。例如,如所期望的,必须要在订单实施处理中执行不同的服务,例如应收帐款。因此,这些服务可以用语言来模型化,例如BPEL。尽管描述的是BPEL,但本领域技术人员可以容易地理解,也可以使用其他的语言。

[0070] 可配置单元是由客户建立并定义的服务。例如,在由用户配置的服务周围提供包装器。例如,客户需要专门对客户的公司的运输服务。因此,由可配置单元执行的服务可由客户定义并建立,但包装器允许运行时间引擎312来自动调用服务。这使得客户能够定义他们各自组织所需要的服务。

[0071] 服务可以在不同业务处理中重新使用。服务可以被封装并被配置为接收待执行服务的公共签名。例如,对于每个业务处理,可以提供不同的参数(即用不同的价格订购不同的产品,等等)。这导致了不同的输入变量输入到了服务中。公共签名定义了一个数据结构,以允许对不同的可执行处理310重复使用该服务。这样,用相同部署的服务来处理不同订单的不同输入变量,但是可以获得不同的结果。以这种方式,就能抽象出订单实施处理。不同的用户可以定义哪个服务要被执行而不用考虑到这些处理是如何用编排语言编码的。

[0072] 接口308可以是一个管理用户接口。例如,图形用户接口允许用户在一个抽象级别上模型化业务处理。例如,服务库306可以被提供给客户机304。然后用户可以使用接口308来定义使用了服务库306中的服务的业务处理的步骤。用户可以定义业务处理中的许多步骤。每个步骤可以与服务库306中的服务相关联。这些步骤可以存储在数据表中,数据表可以包括由运行时间引擎312使用的元数据,用以编排可执行处理310。数据表被示为存储在存储器314中。可以理解,数据表可以存储在任意区域中,例如在客户机304、编排系统302或其他设备中。元数据可以由用户来定义、从数据表和/或编排规则中确定。用户定义了要调用的服务的顺序以及需要的条件或并行分支,来影响业务处理规则。当用户选择了用于一个处理步骤的服务时,用户也提供了附加的元数据,用于确定在运行时间中如何在订单的处理过程中执行处理数据。例如,定义条件或并行分支。

[0073] 在运行时间中,运行时间引擎312接收元数据并使用元数据来确定用于编排可执行处理310的参数。运行时间引擎312使用这些参数来确定可执行处理310中的哪些步骤要执行,什么时候执行。例如,运行时间引擎312通过调用一系列由用户定义的步骤中的服务,来编排可执行处理310。如下更详细的描述,也可以执行步骤的并行和条件处理。而且,元数据能够用于确定用来调用服务的输入变量。

[0074] 在运行时间中读取表的元数据,并调用服务,这允许执行对可执行处理310的修改,并自动地在运行时间中实现。运行时间引擎312读取所定义的每个步骤并执行这些步骤。如果需要在服务中进行修改,用户可以使用接口308来增加/删除/替换一个服务。在运行时间中,当读取表时,可以自动地执行这种修改。

[0075] 图4示出了根据一个实施例的接口308的实例。处理级别表416概括了已被模型化的不同业务处理。如图所示,业务处理-地毯安装和处理1-已由用户模型化。

[0076] 在处理级别表416中,处理名称列418示出了业务处理地毯安装业务处理和处理1已经被模型化。描述列420描述了该处理。处理分类列422描述了处理的分类。状态列426是可执行处理的状态。可执行处理310可以具有不同的状态。例如,某些业务处理可以被批准为生产、被批准为测试、或可以是新的。被批准为生产意味着服务被批准为正常的业务使

用,被批准为测试意味着被批准为进行测试,而新的是在开发中的服务。

[0077] 表416中的业务处理可以被选择,并且数据表400示出了单个的业务处理的步骤细节。一个业务处理被命名为地毯安装,数据表400的步骤细节示出了已经对地毯安装定义的每个服务。

[0078] 在数据表400中,步骤列404标识了业务处理中的步骤。例如,提供步骤10-60。用于这些步骤的服务可以在运行时间中被执行。这些步骤可以从顶部到底部按顺序(或以任意其他的次序)执行。在这种情况下,执行步骤10,然后当完成后,执行步骤20,等等。另外,尽管未示出,还可以使用接口308定义条件和并行步骤。条件步骤是依赖于结果的发生的步骤(例如另一步骤的结束),而并行步骤是并行执行的。用户定义了步骤是条件的还是并行的。

[0079] 步骤名称列406提供了用于各步骤的描述名称。例如,提供运输地毯、等待运输、安装地毯、等待完成、开账单等步骤。

[0080] 任务类型列408描述了什么类型的任务正在被执行。例如,对于运输地毯的任务,外部系统可以执行运输任务,对于开账单的步骤,账单系统可以为一个账单开账单。

[0081] 服务列412标识了与该步骤相关的服务。任务名称列414是任务的名称。例如,那些与地毯相关的任务被命名为地毯运输、地毯安装、为地毯开账单。如果正在安装的是地毯之外的其他货物,任务名将会不同。例如,水槽运输、水槽安装、为水槽开账单将会成为这些任务的名称。

[0082] 用户可以使用接口308来产生数据表400。用户可以从用于服务库306的菜单中选择服务。例如,用户使用菜单接口432来从服务库306中选择服务。下拉菜单、拖放选项和其他的可视处理可以用于定义可执行处理310。用户具有一个具体编排接口,用于以合适的验证呈现业务处理数据,而不是要去学习多用途IT开发环境的复杂性。这允许用户以抽象的方式模型化一个业务处理,但是使可执行处理310从该模型中产生和执行。

[0083] 服务库306中的服务可由非可配置单元和可配置单元来构成。例如,非可配置单元设置在列440中,可配置单元设置在列442中。如图所示,非可配置的服务包括运输、应收帐(“AR”)、开账单以及全局订单承诺(“GOP”)。而且,可配置的单元被指定为A、B、C和D。如图所示,使用菜单412在接口308中产生表400。表400与元数据和任意变量相关,其中元数据描述了要被执行的服务,变量用于调用这些服务。

[0084] 一旦业务处理在接口308中被模型化并通过设置处理状态而被释放,那么运行时间引擎312用于编排服务的调用。图5示出了根据一个实施例的运行时间操作。在该实施例中,表读取器502从接口308接收定义了业务处理的元数据。表读取器502可以将数据复制到运行时间表506中,但这不是必须的。

[0085] 根据该实施例,在运行时间期间,步骤读取器504被配置为读取运行时间表506中的步骤。步骤读取器504可以分析元数据并确定那些步骤应当被执行以及何时被执行。例如,步骤读取器504验证并行或条件分支是否与一个步骤相关。元数据还用于确定用于服务的输入变量。输入变量可以从元数据、查找表中的数据中确定,或者使用规则来确定。

[0086] 根据该实施例,步骤读取器504使用来自服务306的封装的服务以及元数据来组合可执行处理310。例如,为可执行处理310确定在这些步骤中被模型化的每个服务的代码。还可以确定每个服务的输入变量。例如,元数据用于确定输入变量,以便使服务能处理用于业务处理的订单。而且,可使用元数据来确定任意的合作者链接,以便允许服务与外部系统交

互。基于业务处理中的步骤的定义来组合可执行处理310。由于服务是可重复使用的，用于一个服务的相同代码可以用于不同的业务处理。然而，输入变量或合作者链接可以是不同的。由于相同的代码被重复使用，因此提供了可执行处理310的自动组合。

[0087] 在该实施例中，流程序列器508用于基于可执行处理310在适当的时间动态地调用这些步骤。如方块507所示，步骤10可以确定要调用的服务。然后执行步骤20、30、40和50中的一个。然后步骤60确定是否需要执行其他步骤。在这种情况下，步骤20、30、40和50中的一个可能会被执行。流程序列器508可以依赖于接收的元数据的内容来确定相关的输入变量。然后这些输入变量用于调用一个服务。例如，流程序列器508可包括一个任务层读取器510，其确定要调用的服务。任务调用器512然后动态地调用该服务。使用任意的输入变量来调用该服务。在调用服务的过程中，执行已封装服务的代码来协调服务的执行。例如，准备可执行代码并向外部系统发送一个消息来执行该服务。

[0088] 然后可以执行服务并且在结果接收器514处来接收结果。在一个实例中，如果任务是运输，那么运输服务就对运输系统产生一个关于货物运输的消息。一旦该运输系统运输该货物，存储了结果的消息就会返回到该运输服务。

[0089] 在接收结果之后，验证是否要执行更进一步的序列。例如，一个“当”活动模块检查是否需要处理更进一步的服务。例如，处理可以返回到流程序列器508，以便允许动态调用处理中的其他步骤。而且，该“当”活动模块可以等待直到并行分支完成为止。

[0090] 因此，自动地基于该运行时间表来确定调用服务所需的信息。在一个实例中，在BPEL中，已经创建了所有调用所必需的合作者链接，并且它们被用于调用服务。在BPEL合作者链接中所呈现出的服务被部署成BPEL处理，其不需要更多的配置就可以用在多个业务处理的定义中。当运行时间引擎调用了一个服务时，在下层BPEL处理中访问相应的合作者链接。通过使用运行时间表中的元数据来产生服务的组合以及任意服务的修改，并且可以通过接口308来管理。

[0091] 因此，用户可以在业务处理中建立这些步骤。可执行处理310可以自动地在运行时间上组合。可执行处理310中使用的代码不是由建立该业务处理的用户产生。而是，可以定义元数据并用其来组合已封装的用于可执行处理310的服务。

[0092] 图6示出了根据一个实施例的使用流程序列器508调用服务的一个实例。在步骤602，根据该实施例，确定是否需要分支。如果遇到条件声明，处理就基于满足了哪个条件从而前进到合适的分支。如果遇到并行分支，就产生并行流程序列实例，来执行附加的分支。确定分支并随后将其用在服务的调用中。然后处理前进到在其中确定服务的步骤604。

[0093] 然后执行各种服务。这些步骤包括调用服务步骤、调度步骤、运输步骤、等待步骤、账单步骤和子处理步骤。同样的处理序列采用并行流动的方式，直至到达合并步骤。每个流程序列包含相同的下层代码化处理(例如BPEL处理)，但是不同的处理序列可以用在不同的可执行处理310中。也就是说，一个序列可以包含调度、运输、账单，而另一序列可以包括调度、活动、运输、活动、账单，虽然包含下层代码化处理的运行时间引擎不变。也就是说，尽管正运行着不同的可执行处理，但被调用的每个服务的代码保持相同。

[0094] 流程序列器的每个分支中包含有外部服务调用，每个可被调用的服务具有一个分支。分支包含所有用于设置应当被包括在发送给特定外部服务的消息中的数据所必须的步骤以及用于格式化从该服务接收的响应的必须的步骤。一旦完成服务，“当”活动模块验证

是否有进一步的服务要处理,然后返回到流程序列器508、继续处理的下一步骤或者等到任意并行分支的完成。

[0095] 方块606示出了可执行处理310的概念性执行。不是所有的步骤都立即运行。例如,为步骤10调用该调用服务,并且确定要调用的服务。一旦这些完成之后,步骤608确认是否需要执行其他的步骤。在这种情况下,步骤604确定调度、运输、等待、账单和子处理服务应当被执行。一旦全部的流程都已完成,就会构造统一的结果集合。基于该可执行处理的定义,确定是否应当执行附加处理。执行不同的分支,每个分支调用相关的服务。服务的输入变量从运行时间表中的元数据产生。当已执行所选的服务时,步骤608确定是否应当执行附加的服务。如果是,处理返回步骤602重复处理。如果不是,结束处理。

[0096] 使用表400的信息来提供服务的编排。然而,除了编排之外,服务需要与外部系统进行通信。图7示出了根据一个实施例的在不同的层中用于编排数据流的处理。提供编排层、任务层、外部接口层和外部系统层。在一个实施例中,在编排层之前提供一个分解层(未示出)。

[0097] 根据该实施例,步骤702为任务产生并发送一个调用。可以从订单捕获模块接收一个订单。这可导致任务的调用。使用在运行时间表中找到的数据来产生调用请求。请求被发送给任务层。

[0098] 根据该实施例,步骤704启动该任务,产生用于外部系统的消息,并发送该消息。所产生的消息表示哪个任务应当由该外部系统来执行。要被执行的任务是已经模型化的订单处理的一个方面。例如,任务可以是一个订单开账单。还可以包括执行该服务的参数。该消息被发送到一个外部接口。

[0099] 根据该实施例,步骤706转换该消息并将其发送到外部系统层。任务层产生的消息可以是通用格式的。然而,不同的外部系统可以使用其他的格式来通信。外部接口层确定外部系统所使用的格式,并转换该消息。例如,用户定义的元数据可以用于确定要使用的格式。在一个实例中,对什么外部系统调用了所订购的产品的映射,被用来翻译该消息。

[0100] 根据该实施例,步骤708接收该外部系统返回的消息并处理该产生了结果的消息。该外部系统可以是执行与处理一个订单相关的任务的系统,例如,调度系统、运输系统等等。当执行任务的时候,确定该任务的结果。结果可以是安排好的运输日期、运输货物的日期等等。然后结果被发送回外部接口层。

[0101] 在该实施例中,步骤710转换该消息并将该消息发送到任务层。步骤712基于结果更新用于该任务的信息。例如,结果可以存储在表或数据库中。然后处理继续到下一个可被调用的服务。

[0102] 通过使用以接口308定义的已封装的服务,就能对可执行处理310进行修改并在运行时间中实现。例如,在处理的运行过程中对元数据的改变会影响所采用的步骤的顺序以及各步骤的输入变量。

[0103] 图8示出了根据一个实施例的用于修改业务处理的方法的流程图800。在一个实施例中,图8的流程图800的功能,以及图中所示的其他流程图的功能,是由存储在存储器或其他计算机可读或有形介质中的软件来实现的,并且由处理器来执行。在其他的实施例中,功能可以由硬件(例如通过使用特定用途集成电路(“ASIC”)、可编程门阵列(“PGA”)、现场可编程门阵列(“FPGA”)等),或者软件和硬件的组合来执行。

[0104] 步骤802接收对业务处理的修改。例如，接口308用于修改业务处理以包含不同的步骤。在一个实例中，步骤可以被替换、删除或增加。

[0105] 步骤804接收用于修改的元数据。例如，运行时间引擎312可以接收该修改的元数据。然后步骤806修改运行时间表来反映对元数据的修改。例如，可执行处理310可被修改以包含不同服务来调用。

[0106] 当要调用一个服务时，步骤808读取运行时间表以便确定要调用的服务。例如，步骤读取器504可以在可执行处理310的处理期间读取该表。如果运行时间表已被修改，步骤读取器504基于修改确定要被执行的下一步骤。

[0107] 然后步骤810调用所确定的服务。由于可以基于不同的输入变量调用服务，因此在业务处理中的服务被修改时，就不需要进行附加的编程来重新部署新服务。相反，可以仅修改表就能自动地调用不同的服务。

[0108] 然后步骤812确定是否需要执行更多的服务。如果是，处理返回到步骤806，再次读取表来确定要执行的下一步骤。如果不是，处理结束。

[0109] 因此，受数据驱动的编排提供了抽象化和灵活性。抽象化指的是对处理元数据的基于网络的管理，该元数据定义了在可执行处理中的处理步骤。在不同的业务处理上重复使用处理代码。灵活性指的是修改处理而不需重新部署代码的能力。对运行时间元数据的修改的使用有助于对可执行处理310的修改。抽象化使处理模型更接近于商业用户并降低了管理成本。灵活性允许商业用户对修改进行响应，例如当业务处理或规则修改时对业务说明书的更改。

[0110] 图9示出了可实现本发明一个实施例的系统900的方框图。在本发明的一个实施例中，图9的系统900对应于图3的编排系统302。系统900包括总线902或其他的通信机制，用于在系统900的部件之间发送信息。系统900还包括处理器914，可操作地耦合到总线902，用于处理信息并执行指令或操作。处理器914可以是任一种通用或专用目的处理器。系统900进一步包括存储器904，用于存储信息和要由处理器914执行的指令。存储器904可包括随机存取存储器(“RAM”)、只读存储器(“ROM”)、如磁盘或光盘的静态存储器或者其他任意类型的机器或计算机可读介质的任意组合。系统900进一步包括通信设备912，例如网络接口卡或其他通信接口，用以提供对网络的接入。结果，用户可以通过网络或任意其他的方式与系统900直接或远程地连接。在本发明的一个实施例中，用户可以通过客户机，例如图3中的客户机304，与系统900连接。

[0111] 计算机可读介质可以是能够被处理器914访问的任意可用介质。计算机可读介质可包括易失性和非易失性介质、可移动和非可移动介质、通信介质和存储介质。通信介质可以包括计算机可读指令、数据结构、程序模块或其他以调制的数据信号为形式(例如载波，或者其他传输机制)的数据，并且可以包括任意的信息传递介质。存储介质可以包括RAM、闪存、ROM、可擦除可编程只读存储器(“EPROM”)、电可擦除可编程只读存储器(“EEPROM”)、寄存器、硬盘、可移动盘、压缩盘只读存储器(“CD-ROM”)或现有技术中所知的任意其它形式的存储介质。

[0112] 处理器914还能够可操作地通过总线902耦合到显示器916，例如液晶显示器(“LCD”)。显示器916可以向用户显示信息。键盘918和光标控制设备920，例如计算机鼠标，也能够可操作地耦合到总线902，来使用户能与系统900进行连接。

[0113] 根据一个实施例,存储器904能存储那些可在由处理器914执行时提供功能的软件模块。模块可以包括操作系统906、分布式订单编排模块908以及其他功能模块910。操作系统906为系统900提供操作系统的功能。分布式订单编排模块908执行业务处理的编排,如上所述以及如下所述。系统900还可以是更大系统的一部分。这样,系统900可包括一个或多个附加的功能模块910,以用于包含多个附加的功能。例如,功能模块910还包括中间件模块,其是来自Oracle公司的“Fusion”产品的一部分。

[0114] 处理器914还可以通过总线902可操作地耦合到数据库934。数据库934能够以逻辑关联的记录或文件的集成收集来存储数据。数据库934可以是操作数据库、分析数据库、数据仓库、分布式数据库、终端用户数据库、外部数据库、导航数据库、存储器内数据库、面向文档的数据库、实时数据库、关系数据库、面向对象数据库或现有技术中所知的任意其它形式的数据库。

[0115] 图10示出了根据一个实施例的分布式订单编排系统1000,该系统能够处理修改请求。在本发明的一个实施例中,系统1000对应于图3的系统300,并且只有系统300中的与此处的讨论相关部分包含在系统1000中。为了清楚略去了系统300的所有其它部分。

[0116] 在图10所示的实施例中,图9的分布式订单编排模块908被表示为两个模块:分解模块1020和编排模块1030。然而,本领域技术人员将会容易地意识到,单个模块也可以提供分解模块1020和编排模块1030的功能,并且也包含在本发明的范围中。而且,图9的分布式订单编排模块908可以呈现为任意数量的模块,并且也包含在本发明的范围中。

[0117] 下面将参照图10所示的分解模块1020和编排模块1030来描述编排的一个示意性实施例。然而,本领域技术人员将会容易地理解,所描述的实施例只是一个示意性实施例,根据可选的实施例也可以实现这种编排,并且也包含在本发明的范围中。

[0118] 分解是将从一个或多个订单捕获模块中接收的数据转换为内部规范格式,用于处理该数据。如上所述,编排是要在业务处理中执行的服务的协调和调用。

[0119] 在一个实施例中,分解模块1020从一个或多个订单捕获模块中接收订单,并作为一个或多个订单捕获模块和编排模块1030之间的中介。订单捕获模块能够在多个通道上捕获订单。在所描述的实施例中,订单捕获模块由订单捕获模块1010来呈现。在本发明的一个实施例中,订单捕获模块1010可以捕获由用户通过图3的接口308输入的信息。然而,本领域技术人员将会容易地理解,订单捕获模块1010可以采用其它的形式,并且也包含在本发明的范围中。

[0120] 根据该实施例,分解模块1020还可用对从订单捕获模块1010发送的对象进行翻译和分解,其中该对象代表一个订单。使用翻译和分解的输出,分解模块1020创建了一个要发送到订单模块1030的分布式订单编排订单(“D00订单”)。D00订单是用来代表从订单捕获模块接收的订单的对象,并且已被转换为可被编排系统使用的对象格式。这样,对“订单”的引用就是对订单捕获系统中用户输入的业务订单的引用,而对“D00订单”的引用就是对编排系统创建和实施的实体的引用,以用于代表由用户输入的业务订单。

[0121] D00订单能够包含分布式订单编排首标(“首标”)。首标是包含了订单的整体分层结构的对象。D00订单能够包括一个或多个群,其中群是用于收集分布式订单编排订单行(“行”)的实体,用于在编排处理的一个实例中进行处理。每个群可以包含一个或多个行。行是包含订单的相应的行的信息的对象。每个行可以包括一个或多个分布式订单编排实施行

(“实施行”)。实施行是一个与相应的实施系统的供应活动相对应的对象，其中该实施系统能够处理该订单行。这样，实施行是实施相应的实施任务的供应行。

[0122] 在本发明的一个实施例中，分解模块1020对订单的创建包含了下列步骤。首先，创建首标。接着创建一个或多个行并与该首标相关联。随后，对于每个行，创建一个或多个实施行，其中一个实施行可以仅与一个行相关联。接着，调用一个服务来为每个行分配单独的可执行处理。然而，在本发明的某些实施例中，省略了为每个行分配单独的可执行处理的步骤，并且用单个可执行处理来处理整个D00订单。在任意一个方案中，分解模块1020基于可执行处理的名称和创建日期来选择一个可执行处理。

[0123] 最后，根据该实施例，分解模块1020保存D00订单状态。

[0124] 编排模块1030在执行和实施由分解模块1020通过可执行处理的创建而创建的D00订单的同时，控制着出现事件的顺序。在图10所示的实施例中，编排模块1030包含子模块订单处理管理器(“OPM”)1040，以及三个核心处理编排管理器(“OM”)1050，步骤管理器服务(“SMS”)1060，以及分割处理管理器(“SPM”)1070。然而，本领域技术人员将会容易地意识到，这是一个实例，编排模块1030可包含任意数量的子模块和处理，并且也包含在本发明的范围中。

[0125] 根据该实施例，分解模块1020通过传递D00订单的首标标识来调用编排模块1030的OPM 1040。OPM 1040能发动一个或多个可执行处理，还能连接和控制一个或多个可执行处理。OM 1050、SMS 1060和SPM 1070是构成可执行处理的核心模块，该可执行处理在处理并实施由分解模块1020创建的D00订单的同时，控制着可出现的事件的顺序。OM 1050由OPM 1040来调用，并能够执行用于给定群的处理步骤。SMS 1060由OM 1050调用，并能够封装用于预处理、错误处理和修改管理的业务逻辑。SPM 1070由OM 1050来调用并能够控制分割单元。分割单元定义了可执行处理中可被一起分割的一系列步骤。例如，可执行处理包括了调度、运输和账单的步骤。在该实例中，分割单元可以被定义为包括调度和运输步骤，但不包括账单步骤。基于该分割单元定义，在分割该可执行处理的方案中，可以并列执行所得到的分割步骤，并且只有当两个步骤都完成后才能调用该账单步骤。

[0126] 在该实施例中，OPM 1040确定适当的可执行处理来编排D00订单。对于D00订单中的每个群，OPM 1040通过查找一个群表并顺序地发动用于该群的可执行处理，来确定可执行处理。如果可执行处理不存在，那么在发动可执行处理之前，OPM 1040调用一个服务来组合该可执行处理。在本发明的一个实施例中，OPM 1040还能够查询要在首标级别上执行的任务服务的列表，并在用户定义的序列中执行它们。

[0127] OM 1050是之前标识的可执行处理的实例，其生命周期开始于OPM 1040对其的异步调用。当OM 1050执行了其所有的处理步骤之后，OM 1050停止。根据该实施例，OM 1050用于启动由业务处理定义的任务层服务的调用。而且，OM 1050具有区分分割单元和非分割单元的逻辑。对于分割单元，OM 1050可以启动SPM 1070的调用，其控制着分割单元。

[0128] SMS 1060也可以被OM 1050调用。在OM 1050用作处理编排引擎时，SM 1060起到步骤编排引擎的功能。特别地，在该实施例中，SMS 1060接受来自OM 1050的请求，为每个步骤取回运行时间信息，标记步骤的状态为“已开始”，发送请求至任务层，处理来自任务层的响应，并最终将控制送回OM 1050。

[0129] 修改管理框架

[0130] 如前所述,分布式订单编排功能的基本核心是使用编排处理以用于编排订单。编排处理控制着在处理并实施订单时出现的事件的顺序。

[0131] 仍如前所述,业务处理可以是由几个业务步骤构成的长期运行的处理。在一个实施例中,业务步骤总是由用户来定义的,并且表示业务流程中的一个步骤。在分布式订单的编排业务处理中的业务步骤包含任务层服务或者子处理。业务步骤不能参与处理的办理,这是因为如果处理的办理被回滚,那么它不能被自动的撤消。

[0132] 核心功能的一个关键要求是在执行和实施订单的同时管理修改请求。修改请求包括一个引用原始订单的新订单。新订单还包括对原始订单的修改,因此,包括了对含有业务处理的业务步骤的修改。因此,修改请求将会导致对当前正执行的业务处理(以及相应的可执行处理)的显著更改。处理办理不足以处理该修改请求,这是因为业务处理的几个业务步骤与外部实施系统交互,因此超出了办理界限。因此,这些业务步骤要求特殊逻辑来容纳修改请求。

[0133] 根据本发明的一个实施例,分布式订单编排系统能够接收修改请求并确定原始订单和新订单(也被称为“修改订单”)之间的修改。然后使用原始订单和新订单之间的修改来确定哪些业务处理修改是必要的,以响应修改请求。这样,系统能够应付新业务处理的步骤与正执行的或已被完成的原始业务处理的步骤有冲突的情形。除了自动调整原始业务处理的过去的业务步骤之外,分布式订单编排系统能够包含对还未执行的业务步骤的修改。

[0134] 重要的是,编排语言补偿(例如BPEL补偿)和分布式订单编排修改管理是非常不同的。BPEL补偿用在由于可执行处理中的错误条件而回滚处理中已执行的活动的效果。分布式订单编排修改管理不仅包含对业务处理中以前执行的步骤的撤消,而且还包括对业务处理中尚未执行步骤中的修改的向前传播。换言之,后者需要撤消或重复之前执行的步骤以及更新尚未执行的步骤的能力。而且,对以前执行的步骤的撤消不仅仅只是回滚到以前的状态。相反,需要对服务的调用来采取适当的撤消动作。

[0135] 在本发明的一个实施例中,通过框架范围向修改管理框架提供嵌套的功能范围。框架范围用于以正常模式或修改模式来执行业务处理的步骤。当第一次运行分布式订单编排系统的可执行处理时,以正常模式来运行可执行处理。在正常模式下,在适当的时间动态地执行可执行处理的步骤,如之前在单独的部分中所描述的那样。然而,当接收到修改请求时,分布式订单编排系统停止了原始的可执行处理(及其所有的子处理),并且以修改模式启动新的可执行处理。在一个实施例中,停止原始的可执行处理包括终止原始的可执行处理。然而,在一个可选的实施例中,停止原始的可执行处理包括暂停原始的可执行处理,其中原始的可执行处理可以在随后的时间点上进行恢复。新的可执行处理与原始的可执行处理相关以用于允许修改处理。在修改模式中,执行适当的修改步骤来自动地调整已被执行的原始的可执行处理的步骤。使用之前保存的原始的可执行处理的状态来执行修改步骤。一旦执行了修改步骤,就以正常模式使用新的可执行处理的当前状态来执行新的可执行处理的剩余步骤。在本发明的一个实施例中,可执行处理能够在每个里程碑处保存处理的状态。保存的状态可以用在修改模式中,用于撤消或重复以正常模式执行的可执行处理的步骤。

[0136] 图11示出了根据本发明一个实施例的处理修改请求的方法的流程图1100。在1110中,以正常模式执行原始的可执行处理。如前所述,当以正常模式执行可执行处理时,可执

行处理执行包含了可执行处理的步骤并为每个步骤调用相应的服务,如参照图5所讨论的。

[0137] 在1120中,接收修改请求。如上所讨论的,该修改请求包括一个由订单捕获模块捕获的新订单,其引用了一个原始订单,其中新订单包含了对原始订单的修改。按照处理订单的相同方式来处理修改请求,如上所讨论的,创建一个新D00订单。

[0138] 在1130中,停止原始的可执行处理。停止原始的可执行处理包括停止已由该原始的可执行处理创建的任何子处理。在本发明一个实施例中,终止原始的可执行处理。在一个可选实施例中,仅暂停原始的可执行处理。在该实施例中,原始的可执行处理可以在随后的时间点上进行恢复。

[0139] 在1140中,创建一个新的可执行处理。更具体地,基于新的D00订单创建该可执行处理,其进而基于包含在所接收的修改请求中的新订单。

[0140] 在1150中,以修改模式执行新的可执行处理。当以修改模式运行新的可执行处理时,可执行处理执行修改步骤,以便更改已由原始的可执行处理执行的步骤。新的可执行处理还使用原始D00订单与新D00订单之间的差别,执行未被原始的可执行处理执行的步骤。

[0141] 图12示出了根据一个实施例的详述修改请求流程的一个实例的流程图1200。在流程图1200中,流程1210表示原始的可执行处理的流程。例如,原始的可执行处理可以对应于订购地毯的业务处理。原始的可执行处理执行步骤A、B和C。在上面的实例中,步骤A包括从库存中选择地毯,步骤B包括根据所要求的尺寸测量地毯,以及步骤C包括运输地毯。

[0142] 在该实施例中,在完成步骤B之后但未启动步骤C时接收一个修改请求(未示出),其中该修改请求将地毯订单修改成瓷砖订单,这里用于订购地毯的处理和用于订购瓷砖的处理使用了相同的业务处理。因此,原始的可执行处理被停止,新的可执行处理被启动。在上面的实例中,新的可执行处理对应于用于订购瓷砖的业务处理。在流程图1200中,流程1220表示新的可执行处理的流程。新的可执行处理执行步骤A'、B' 和C'。步骤A' 包括调整已完成的步骤A。对步骤A的调整可以根据下面的业务处理来采取多种形式中的一种。在所示的实例中,对步骤A的调整可以包括把从库存中选择地毯调整为从库存中选择瓷砖。如果之前所选的库存不包括瓷砖,那么这种调整还进一步包括选择包含瓷砖的不同的库存。相似地,步骤B' 包括调整已完成的步骤B,并根据下面的业务处理来采取多种形式中的一种。在所示的实例中,对步骤B的调整可以包括测量瓷砖,以瓷砖的测量来代替地毯的测量。最后,步骤C' 包括运输瓷砖。由于步骤C并未由原始的可执行处理来执行,因此不需要调整步骤C,这样,不执行对步骤C的调整。然而,是基于新订单来执行步骤C' 的,而不是基于原始订单。这样,在所示的实例中,步骤C' 包括运输瓷砖,而不是运输地毯。

[0143] 下面将参照图10所示的分解模块1020和编排模块1030来描述编排修改管理的一个示意性实施例。然而,本领域技术人员可以容易地理解,所描述的实施例仅是一个示意性实施例,也可以根据可选实施例来实现这种编排修改管理,并且仍包含在本发明的范围内。

[0144] 在一个实施例中,图10的分解模块1020和编排模块1030能够处理一个修改订单的请求(即修改请求)并处理一个订单。在订单捕获模块发送一个修改请求的情况下,分解模块1020按照分解模块处理原始订单的方式来处理该修改请求,如之前参照图10的描述,除了下面的关键区别点之外。

[0145] 首先,根据该实施例,分解模块1020识别出从订单捕获模块接收的新订单不是一个真正的新订单,而是修改原始订单的请求的一部分(即修改请求)。接着,分解模块1020验

证是否允许在原始订单上进行修改。该验证包括回顾原始订单的状态以及相应的可执行处理的状态。如果订单状态或处理状态具有阻止修改请求的约束条件,那么修改请求就被拒绝。然而,如果允许修改,分解模块1020就通知编排模块1030准备处理修改请求。接着,分解模块1020创建一个新的DOO订单。新的DOO订单引用了原始DOO订单。然后分解模块1020将新DOO订单关联并映射到原始DOO订单。分解模块1020还计算新的DOO订单和原始DOO订单之间的变动(δ)。最后,分解模块1020将新的DOO订单发送到编排模块1030。

[0146] 根据该实施例,编排模块1030能够通过使OPM 1040监听修改通知和修改请求来处理这些修改请求。修改通知只是来自分解模块的一个通知,用以准备处理一个修改请求,它不是实际的修改请求。在出现修改请求的情况下,OPM 1040首先接收一个修改通知,随后接收实际的修改请求。

[0147] 当OPM 1040接收一个修改通知时,根据该实施例,OPM 1040可以通知每个由OPM 1040调用的OM 1050。基于这个通知,OM1050不执行任何的新任务。当OPM 1040接收一个修改请求时,OPM 1040能够确定当前的可执行处理是否能够接受修改请求。如果该处理不能接受修改请求,那么OPM 1040就拒绝整个修改请求。如果处理能接受修改请求,那么OPM 1040就如下所述处理该修改请求。

[0148] 在处理该修改请求的过程中,根据该实施例,OPM 1040首先调用登记为可执行处理的每个步骤的一部分的通知服务,用以确定是否每个通知服务都能接受修改请求。如果任意一个所登记的通知服务表示它们不能接受修改请求,那么OPM 1040就拒绝整个修改请求,结束修改请求的处理。

[0149] 然而,如果全部所登记的通知服务表示它们能接受修改请求,那么就继续修改请求处理。然后OPM 1040通知与该可执行处理相关的等待步骤以便停止。然后根据本实施例,OPM 1040将新DOO订单与原始DOO订单合并。

[0150] 在该实施例中,然后OPM 1040调整用于新DOO订单的群信息。当分解模块1020创建一个新DOO订单时,分解模块1020创建用于每一行的新群,以及新DOO订单的实施行。取决于每一行和实施行是否是新的,OPM 1040能够调整引用键值、激活某些群,以及去激活其他群。

[0151] 根据一个实施例,为了对新DOO订单的每一行调整群信息,OPM 1040确定原始DOO订单中是否存在行。如果原始DOO订单中存在行,那么OPM 1040进一步确定新DOO订单中是否已修改了该行。如果原始DOO订单中不存在行,OPM 1040激活分解模块1020为新行创建的群,并且为该行设置属性变动。如果原始DOO订单中存在行,并且未在新订单中修改,OPM 1040就继续使用来自原始DOO订单的之前的群。如果原始DOO订单中存在行,并且已在新订单中修改,OPM 1040就激活分解模块1020为所修改的行创建的群,为该行设置属性变动,并且去激活为原始订单所创建的之前的群。OPM 1040还执行用于新DOO订单的每一实施行的那些步骤。

[0152] 然而,在可选实施例中,OPM 1040能够基于不同的标准来调整群信息,并且仍包含在本发明的范围内。

[0153] 随后,根据该实施例,OPM 1040处理所计算的原始DOO订单和新DOO订单之间的变动。一旦该变动被计算出来,OPM 1040确定变动类型并执行下列一组活动中的一个。

[0154] 如果变动类型是一个增加行变动(即增加一个新行),并且该行被增加到新DOO订

单中,那么在本发明的一个实施例中,OPM1040为该行开始一个新的可执行处理,并对其和用于该新DOO订单的其他可执行处理一起监控。在该实施例中,OPM 1040还可以修改从新DOO订单到原始DOO订单的群的引用。OPM 1040可以执行这些操作而不需考虑新行是否仅增加到新DOO订单上,或者是否新行还增加到新DOO订单的新群上。

[0155] 在一个可选实施例中,如果变动类型是一个增加行变动,并且该行被增加到新DOO订单的现有群上,那么OPM 1040就开始一个新的可执行处理,并如上所讨论地修改引用。另外,OPM 1040可以将新DOO订单中新创建的群与现有群进行合并。这种合并的一个实例是基于项目关系和共享处理的定义来将新创建的群与现有群进行合并。

[0156] 如果变动类型是一个删除行变动(即删除一个行),在本发明的一个实施例中,OPM 1040通知合适的可执行处理来删除该行。

[0157] 如果变动类型是一个变动属性修改变动(即已修改的首标、行或实施行的一个或多个变动属性),在一个实施例中,OPM 1040通知合适的可执行处理来更新该属性。在本发明的一个实施例中,如果修改的属性是一个数量属性,就由OPM 1040来分别处理该修改以最小化调整的必要和更好的优化。特别是,对于数量增加,OPM 1040为合适的行开始新的可执行处理,并对其和用于该DOO订单的其他可执行处理一起进行监控。然而,对于数量减少,OPM 1040将会通过以之前所讨论的修改模式执行新的可执行处理来调整原始的可执行处理。

[0158] 如果变动类型是一个动态变动属性变动(即已修改的首标、行或实施行的一个或多个动态变动属性),在一个实施例中,OPM 1040通过以之前所讨论的修改模式执行可执行处理来调整原始的可执行处理。在本发明的一个实施例中,用户可以在定义业务处理或定义业务处理的步骤时在订单捕获元件的接口处指出编排系统是否应该忽视对动态变动属性的修改。动态变动属性是用户定义的作为变动属性的一种属性。

[0159] 最后,根据一个实施例,OPM 1040关闭新的订单并调用新的可执行处理。在图10中,如之前所讨论的,新的可执行处理被定义为OM 1050。

[0160] 关于OM 1050,根据一个实施例,OM 1050监听修改请求。一旦OM 1050接收了修改请求,就处理该修改请求。

#### 编排处理管理

[0162] 如前所述,编排处理的修改管理使用了可执行处理的以往步骤的自动调整的组合,以及对可执行处理的未来步骤的修改的结合。随后将以本发明的一个示意性实施例来描述该编排处理和编排处理的修改管理。

[0163] 图13示出了根据一个实施例的可执行处理定义的一个实例。在该实施例中,图13示出了原始订单的可执行处理的定义。S1、S2、S3、S4、S5、S6和S7表示可执行处理的不同步骤。CB1和CB2表示可执行处理的条件分支,用于在运行时间上根据预先定义的业务规则进行评估。例如,在CB1,编排系统评估原始订单的行数量是否小于10。如果行数量小于10,那么可执行处理执行步骤S2,然后是步骤S4,再然后步骤S5。然而,如果行数量大于或等于10,那么可执行处理执行步骤S3,然后评估条件分支CB2。在CB2,编排系统评估原始订单的组织是否等于204或404。如果组织等于204,那么可执行处理执行步骤S6。然而,如果组织等于404,那么可执行处理的执行步骤S7。

[0164] 在原始订单中,如果行的数量大于10,可执行处理执行步骤S3,如果组织等于204,

那么可执行处理执行步骤S6。如果可执行处理执行那些步骤，并且接收到把行数量降到5的修改请求，那么新的可执行处理删除步骤S3至S6，然后执行步骤S2、S4和S5。由于在运行时间上依靠预定的规则评估每个条件分支CB1和CB2，于是在接收到修改请求时不知道新的可执行处理采用的是哪条路径。

[0165] 在请求修改的情况下中，编排系统通知原始的可执行处理停止。在一个实施例中，编排系统通知原始的可执行处理温和地终止(即允许任何正在运行的步骤完成而不需执行下一个步骤)。在一个可选实施例中，编排系统通知原始的可执行处理自身暂停。这样，在该实施例中，原始的可执行处理被暂停，但能够在随后的时间点上被恢复。在恢复时，原始的可执行处理将执行下一步骤。编排系统创建一个新的可执行处理，其引用了原始的可执行处理。更具体地，新的可执行处理包括引用了原始的可执行处理的原始步骤的新步骤，并包括一个新任务。对于不需要进行调整的步骤，编排系统从原始的可执行处理复制任务完成细节和任务状态。在本发明的一个实施例中，任务完成细节包括开始和结束日期。对于需要进行调整的步骤，编排系统简单地从原始的可执行处理复制任务完成细节。编排系统去激活所有与原始的可执行处理相关的信息，并且以修改模式开始新的可执行处理。一旦新的可执行处理已经调整了原始的可执行处理的所有原始步骤，新的可执行处理以正常模式恢复执行剩余的步骤。

[0166] 图14示出了根据一个实施例的新的可执行处理定义的一个实例，其中新可执行处理调整原始的可执行处理的步骤。在图14中，标题为“正常处理实例”的下面是原始的可执行处理的流程，包括步骤A、B、D、E和F以及条件分支S。标题为“补偿处理实例”的下面是一个相应的新的可执行处理的流程，包括步骤A'、B'、D'、E'和F'以及条件分支S'。在修改请求的情况下，编排系统能停止原始的可执行处理的流程，并启动新的可执行处理的流程。如果原始处理的相应的步骤(即步骤A、B、D、E和F以及条件分支S)已被执行，那么新的可执行处理的每一步骤和条件分支(即步骤A'、B'、D'、E'和F'以及条件分支S')可以自动地调整原始处理的相应的步骤。

[0167] 也可以参照图14，原始的可执行处理和新的可执行处理都能够在每个里程碑处将各自处理的状态保存在数据库中。例如，在图14中，原始的可执行处理在里程碑A保存状态S1，在里程碑B保存状态S2，在里程碑C保存状态S3，在里程碑D保存状态S4，以及在里程碑F保存状态S5。

[0168] 图15示出了根据一个实施例的在接收到修改请求时，原始的可执行处理和新的可执行处理的流程图。原始的可执行处理的流程(即在图15的图例标识为“正常订单流程”)包括步骤1-3、5、7和9。新的可执行处理的流程，即在图15的图例标识流程为“修改订单流程”)包括步骤1' -3'、5'、7'、9'和10-11。步骤4、6和8是两个流程所共有的(并且在图15的图例标识为“共有”)。

[0169] 现在描述正常订单流程的步骤。在步骤1，订单捕获模块向编排系统提交一个订单，并且编排系统的分解模块接受该订单。在步骤2，分解模块转换该订单并创建一个原始的D00订单。在步骤3，分解模块按需为原始的D00订单的行分配单独的可执行处理。分解模块还保存了原始的D00订单的状态，并通过调用编排模块的OPM来将原始的D00订单传递到编排模块。

[0170] 在步骤4，OPM基于原始订单的首标标识来查询处理信息。对于原始订单的每个处

理,OPM调用相应的OAS和计划服务。在步骤5,对于原始订单的每个群,OPM调用了一个原始的可执行处理(即OM)。在步骤6,OAS调用该计划服务并作为步骤4的一部分。

[0171] 在步骤7,执行原始的可执行处理。原始的可执行处理进一步调用SMS以便执行可执行处理的步骤。在步骤8,SMS取回必要的运行时间步骤实例数据。在步骤9,SMS调用与可执行处理的步骤相应的任务层服务。在图15所示的实例中,SMS调用了分别与S1、S2和Sn相对应的任务层服务。

[0172] 现在描述修改订单流程的步骤。在步骤1',订单捕获模块向编排系统提交一个修改请求,并且编排系统的分解模块接受该订单,其中该修改请求包括与原始订单相应的新订单。在步骤2',分解模块转换该新订单并创建一个新的D00订单,并将新的D00订单标识为与原始D00订单相对应。分解模块还按需为新的D00订单的各行分配单独的可执行处理。然后分解模块以修改模式调用群API。在步骤10,分解模块调用一个将新的D00订单映射到原始的D00订单的功能,并计算两个D00订单之间的变动。在步骤3',分解模块通过调用编排模块的OPM来将新的D00订单传递到编排模块。

[0173] 在步骤4,OPM基于新的D00订单的首标标识来查询处理信息。对于新的D00订单的每个处理,OPM调用相应的OAS和计划服务。在步骤5',OPM调用了返回一组信息的应用程序模块API,该组信息包括了原始的可执行处理的标识、与新的可执行处理相应的可执行处理的标识、原始D00订单的所有群、新D00订单的所有群以及所有变动类型,其中新的可执行处理将处理新的D00订单并且调整原始的可执行处理的各个步骤。对于每个修改的群,OPM将修改通知给外部系统,暂停原始的可执行处理,以便使原始的可执行处理温和地退出并终止所有的等待步骤。OPM还将新的D00订单与原始的D00订单合并,保存原始订单的当前状态,并为每个修改的群以修改模式调用新的可执行处理(即OM)。在步骤6,OAS调用该计划服务作为步骤4的一部分。

[0174] 在步骤7',执行新的可执行处理。新的可执行处理进一步调用SMS以便执行可执行处理的各个步骤。在步骤8,SMS取回必要的运行时间步骤实例数据,确定合适的补偿方案,标识原始D00订单和新的D00订单之间已计算的变动,并运行合适的补偿服务。在步骤9'和11',SMS调用与可执行处理的步骤相应的任务层服务。在图15所示的实例中,SMS调用了分别与S1、S2和Sn相对应的任务层服务。特别地,SMS首先执行S2的补偿,然后是Sn的补偿,然后是S1的补偿。接着,SMS重新执行S1、S2和Sn。

[0175] 关于编排的更多的实施细节在美国专利号12/617,698名为“分布式订单编排”、美国专利申请号12/718,585名为“分布式订单编排系统中的修改管理框架”以及美国专利申请号12/697,756名为“使用模版的业务处理的编排”中给出了描述。

[0176] 基于事件的编排框架

[0177] 一个实施例涉及分布式订单编排系统,在该系统中使用一组独立的和离散的事件来编排订单。根据该实施例,用事件管理器来替代分布式编排系统中的有状态的可执行处理,例如可执行处理310。代替用请求/回答的交互模式来编排订单的有状态的可执行处理,事件管理器使用一组离散的和独立的事件来编排订单。在本发明的一个实施例中,事件管理器根据一个处理在事件消息发送系统上向一组用户公布一组事件。每个订购者消耗事件管理器所公布的一个事件,并执行基于该事件的任务。根据该实施例,当事件管理器处理分布式订单编排系统中的不同的组成部分之间的通信时,离散的和独立的事件使分布式订单

编排系统中的不同的组成部分去耦合。这样，分布式订单编排系统的第一组成部分能够与第二组成部分通信，而不用知道第二组成部分的实施细节。本领域技术人员将会知道，一组事件可包括单个事件或多个事件。根据该实施例，如果分布式订单编排系统的两个组成部分互相不直接通信，而是使用媒介来通信，例如事件消息发送系统，那么它们是“不同的”。

[0178] 图16示出了根据一个实施例的依据使用事件管理器的处理来发布一组事件的方法的流程图。根据该实施例，流程包含事件管理器1600、数据库1610和订购者1620。依照该实施例，事件管理器1600是一种无状态的可执行处理。事件管理器1600是分布式订单编排系统中的不同的组成部分，并参照图19和20在编排的上下文中更详细地描述。

[0179] 数据库1610是存储着处理状态和元数据的数据库。处理状态是用于处理一个订单的处理的状态。这样，在该处理是一个编排处理的实施例中，该处理状态是该编排处理的状态。根据一个实施例，在编排一个D00订单的情况下，该处理状态包括用于D00订单的每个首标、行和实施行的属性值。如前所述，元数据用于在订单中定义任务，并用于调用与那些任务相关的服务，例如任务层服务。更特别地，元数据用于确定如何以及何时调用这些服务。而且，基于元数据，产生输入变量并发送给这些服务，以便调用所封装的服务。这样，根据一个实施例，处理状态可用于确定处理中的哪些步骤已被执行，而元数据可用于确定处理中的哪些步骤要被执行以及如何执行这些步骤。数据库1610可以是操作数据库、分析数据库、数据仓库、分布式数据库、终端用户数据库、外部数据库、导航数据库、存储器内数据库、面向文档的数据库、实时数据库、关系数据库、面向对象数据库或现有技术中所知的任意其它的数据库。根据本发明的一个实施例，图16的数据库1610与图3中的存储器314相同。

[0180] 订购者1620是用来订购具体类型的事件的分布式订单编排系统的不同组成部分。当订购者1620之一接收到发布的事件时，订购者确定所发布的事件的类型。如果所发布的事件的类型是订购者已经订购的，那么订购者就消耗该事件并执行基于所消耗的事件的任务。订购者1620从事件管理器1600上去耦合。通常，由于这种去耦合，订购者1620不知道该事件管理器1600，因此也不知道所发布的事件的来源。然而，在某些实施例中，订购者可以知道所发布的事件的来源。根据一个实施例，如将参照图17更详细地描述，订购者1620包括任务层服务，该任务层服务执行基于所消耗的事件的任务。然而，这只是一个示例，在可选实施例中，订购者1620可以是不同于事件管理器1600的分布式订单编排系统的任意一个组成部分。

[0181] 现在要描述根据本发明一个实施例的如图16所示的流程。处理开始，并在步骤1启动事件管理器1600。根据一个实施例，当启动时，事件管理器1600确定处理中的哪些步骤要被执行，并发布一组事件以用于执行所确定的步骤。

[0182] 在步骤2，事件管理器1600从数据库1610中读取一个处理状态和元数据。基于该处理状态和元数据，事件管理器1600确定要执行的处理的步骤，并产生一组要在事件消息发送系统上发布的事件。根据一个实施例，事件包括用以执行任务的指令，其中该任务与处理中的步骤相对应。将参照图21来详细描述事件以及事件管理系统。

[0183] 在步骤3，事件管理器1600在事件消息发送系统上发布一组事件，这些事件被传输到订购者1620。在事件管理器1600发布该组事件之后，事件管理器1600就终止。

[0184] 如前所述，订购者1620可以是分布式订单编排系统的任意一个组成部分。在所描述的实施例中，订购者1620包括订购者1621、订购者1622和订购者1623。然而，本领域技术

人员将会容易地理解,这只是根据一个实施例的订购者的示意性数量,在可选实施例中,订购者1620可以包括任意数量的订购者。如前所述,订购者1620中的每个订购者都订购了具体类型的事件。当订购者1620中的订购者接收一个所发布的事件时,订购者就确定所发布的事件的类型。如果所发布的事件的类型是订购者已经订购的,那么订购者就消耗该事件并执行基于所消耗的事件的任务。

[0185] 在步骤4,在订购者1620的每个订购者执行了其任务之后,每个订购者更新数据库1610的处理状态。如前所述,处理状态是处理的一个状态。因此根据该实施例,更新数据库1610中的处理状态包括更新处理状态用以标识订购者1620所执行的任务已经完成。根据本发明的一个实施例,更新处理状态包括更新D00订单的至少一个属性值。这种更新可包括更新D00订单的首标的至少一个属性值,更新D00订单的行的至少一个属性值,更新D00订单的实施行的至少一个属性值,或者它们的组合。

[0186] 根据该实施例,一旦数据库1610的处理状态被更新,事件管理器1600就再次启动。一经启动,事件管理器1600就基于数据库1610的处理状态和元数据确定是否:(1)事件管理器1600应该产生一组新事件;(2)事件管理器1600应该在产生一组新事件之前等待另一个处理状态的变迁;或者(3)处理状态已到达退出状态。如果事件管理器1600确定应该产生一组新事件,就重复步骤3和4直到处理状态达到退出状态。如果事件管理器1600确定应该在产生一组新事件之前等待另一个处理状态的变迁,事件管理器1600就终止。如果事件管理器1600确定处理状态已到达一个退出状态,那么在步骤5,事件管理器1600就终止该处理。将参照图24详细描述事件管理器1600的确定过程。

[0187] 使用事件管理器在分布式订单编排系统中处理订单的一个实例是编排一个订单。更特别地,在一个分布式订单编排系统中,编排层和任务服务层可使用一个事件管理器来通信。因此,根据本发明的一个实施例,将在编排的上下文中描述图16所示的流程。

[0188] 图17示出了根据一个实施例的使用事件管理器和任务层服务来编排订单的方法的流程图。根据该实施例,处理是用于编排一个订单的编排处理。在步骤1,分布式订单编排系统的分解层创建将被发送到该分布式订单编排系统的一个编排层的分布式订单编排订单(“D00订单”),D00订单是表示从分布式订单编排系统的订单捕获层接收的订单的对象,并且该订单已被转换为分布式订单编排系统的编排层所使用的对象格式。当启动D00订单的编排时,就启动事件管理器1600。

[0189] 在步骤2,事件管理器1600从数据库1610中读取处理状态和元数据。根据该实施例,处理状态是编排订单的编排处理的状态。在该实施例中,处理状态包括用于D00订单的每个首标、行和实施行的属性值。而且,根据该实施例,元数据是用于在被编排的订单中定义任务的数据,并用于调用与这些任务相关的任务层服务。基于该处理状态和元数据,事件管理器1600确定要在事件消息发送系统上发布的一组事件。例如,如果元数据定义了创建运输的任务,事件管理器1600就发布一个由创建运输任务层服务订购的事件,并被标识为“创建运输事件”。

[0190] 在步骤3,事件管理器1600在该事件消息发送系统上发布一组事件,并且将这些事件传输到任务层服务1720中。根据该实施例,每个事件都与一个具体的任务层服务相对应,其中该任务层服务被需要进一步编排该D00订单。在事件管理器1600发布该组事件之后,事件管理器1600就终止。

[0191] 在所示的实施例中,任务层服务1720包括任务层服务1721、任务层服务1722和任务层服务1723。然而,本领域技术人员将会容易地理解,这只是根据一个实施例的任务层服务的示意性数量,在可选实施例中,任务层服务1720可以包括任意数量的任务层服务。如前所述,任务层服务1720的每个任务层服务订购一个具体类型的事件。例如,任务层服务1721是一个“创建运输”任务层服务并订购一个事件类型“创建运输事件”。在该实例中,任务层服务1722和1723是分别订购不同类型事件的任务层服务。当任务层服务1720的任务层服务接收一个发布的事件时,任务层服务确定所发布的事件的类型。如果所发布的事件的类型是任务层服务已订购的,那么任务层服务就消耗该事件并执行基于所消耗的事件的任务。如果所发布的事件的类型不是任务层服务已订购的类型,任务层服务就忽略该事件。在该实例中,当任务层服务1721接收一个事件管理器1600发布的事件类型“创建运输事件”时,任务层服务1721就消耗该事件并执行一个“创建运输”任务。当任务层服务1722和1723接收事件类型“创建运输事件”时,任务层服务1722和1723就忽略该事件,这是因为该事件不是任务层服务1722和1723订购的事件类型。

[0192] 在步骤4,在任务层服务1720的每个任务层服务执行其任务后,每个任务层服务更新数据库1610的处理状态。在该实例中,任务层服务1721执行一个“创建运输”任务,并更新数据库1610的处理状态。根据该实施例,处理状态是编排处理的状态。这样,根据该实例,更新数据库1610的处理状态包括更新该处理状态以便标识由任务层服务1721执行的“创建运输”任务已完成。根据该实施例,更新该处理状态包括更新D00订单的至少一个属性值。这种更新可包括更新D00订单的首标的至少一个属性值,更新D00订单的行的至少一个属性值,更新D00订单的实施行的至少一个属性值,或者它们的组合。因此,根据该实例,更新处理状态包括基于“创建运输”任务的执行而更新D00订单的至少一个属性值。

[0193] 根据图17所示的实施例,一旦更新了数据库1610的处理状态,事件管理器1600就再次启动。一经启动,事件管理器1600就基于数据库1610的处理状态和元数据来确定是否:(1)事件管理器1600应该产生一组新事件;(2)事件管理器1600应该在产生一组新事件之前等待另一个处理状态的变迁;或者(3)处理状态已到达退出状态。如果事件管理器1600确定应该产生一组新事件,就重复步骤3和4直到处理状态达到退出状态。如果事件管理器1600确定应该在产生一组新事件之前等待另一个处理状态的变迁,则事件管理器1600就终止。如果事件管理器1600确定处理状态已到达一个退出状态,那么在步骤5,事件管理器1600就终止该编排处理。

[0194] 然而,虽然图17示出了在编排的上下文中的事件管理器,并示出了事件管理器可用在编排层和任务服务层之间进行通信,但本领域技术人员将会容易地理解,这只是事件管理器的一个实例。而且,本领域技术人员也将理解,分布式订单编排系统的任一层都能使用事件管理器与分布式订单编排系统的另一层进行通信。例如,分基层能够使用事件管理器与编排层进行通信。因此,根据本发明的一个实施例,事件管理器有助于分布式订单编排系统的任意两层或两个组成部分之间的通信。

[0195] 图18示出了根据一个实施例的使用事件管理器、任务层服务和中介器来编排订单的方法的流程图。图18所示的实施例与图17所示的实施例类似,区别在于,任务层服务1720包括中介器1724。本领域技术人员将会容易地理解,中介器是通过从第一个处理接收数据并将所接收的数据从第一处理理解的格式转换为第二处理理解的格式、从而允许两个或多

个处理互相通信的处理。因此，中介器用作两个或多个处理的媒介，减少了两个或多个处理之间的依赖性，并进而减少了耦合。另外，根据图18所示的实施例，中介器1724订购全局事件类型，而不是单个任务层服务订购具体事件类型。因此，在步骤3，在事件管理器1600发布一组事件并且将这些事件传输到任务层服务1720之后，中介器1724就消耗各个事件并且基于事件内所包含的一组参数，中介器1724调用任务层服务1721、任务层服务1722或任务层服务1723。被调用的任务层服务(即任务层服务1721、任务层服务1722或任务层服务1723)执行基于该事件的任务。

[0196] 在一个可选实施例中，事件管理器不仅能控制一个或多个订购者的任务处理，还能控制单个订购者的子任务处理，例如任务层服务。例如，订购者可以执行基于所消耗的事件的任务，其中该任务包括10个子任务。而且，某些子任务可以并行地执行，某些子任务可以顺序地执行，并且某些子任务需要一个或多个子任务完成之后才能执行。在这种情况下，当订购者执行第一个子任务时，订购者可以更新数据库的处理状态。基于数据库中的该处理状态和元数据，事件管理器可以确定哪个子任务(或哪些子任务)可被执行，并因此产生及发布一组事件。这些事件可由订购者来消耗，并且基于这些事件，订购者可以执行相应的一个或多个子任务。接着，订购者可以更新处理状态，并且可以重复该处理直到任务的所有子任务都已完成。

[0197] 下面将详细地描述事件管理器，特别地，在编排的上下文中进行描述。

[0198] 图19示出了根据一个实施例在订单实施的上下文中使用事件管理器提供编排的系统1900的一个实例。在该实施例中，系统1900包括一个编排系统302和一个客户机304。根据该实施例，图19的系统1900与图3的系统300类似，区别在于系统1900包括事件管理器1910而不是可执行处理310。事件管理器1910是由运行时间引擎312调用的无状态的处理。事件管理器1910包含嵌入式逻辑，被配置为确定一组要发布的事件。该逻辑是从存储器314中的元数据和用户使用接口308进行模型化的业务处理的状态中获得的。

[0199] 替代执行可执行处理，运行时间引擎312调用事件管理器1910来编排接口308中已模型化的业务处理。相似于图3所示的实施例，用户可以使用接口308来定义使用服务库306中的服务的业务处理的步骤。用户可以在业务处理中定义多个步骤。每个步骤可以与服务库306中的服务相关联。这些步骤可以存储在数据表中，该数据表可以包括能被运行时间引擎312使用并用以调动事件管理器1910的元数据。基于服务库306中用于定义业务处理的具体步骤的服务，事件管理器1910产生并发布一组被传送给订购者的事件，例如任务层服务。事件管理器1910通过调用一种内部方法来发布一个事件。一旦事件管理器1910发布了该组事件，事件管理器1910就终止。在订购者消耗该组事件并执行基于该消耗的事件的任务之后，订购者就更新已模型化的业务处理的状态，运行时间引擎312再次调用事件管理器1910。重复该处理直到编排好模型化的业务处理，然后，事件管理器1910终止该处理。

[0200] 图20示出了根据一个实施例的利用事件管理器的分布式订单编排系统2000。在本发明的一个实施例中，系统2000对应于图19的系统1900，并且在系统2000中只包含了系统1900中需要进行讨论的部分。为了清晰起见，省略了系统1900的其它部分。

[0201] 在图20所示的实施例中，图9的分布式订单编排模块908被表示为两个模块：分解模块1020和编排模块1030。然而，本领域技术人员将会容易地认识到，一个模块就可以提供分解模块1020和编排模块1030的功能，并且仍包含在本发明的范围中。而且，图9的分布式

订单编排模块908可以被表示为任意数量的模块，并且这也包含在本发明的范围中。

[0202] 根据该实施例，图20的系统2000与图10的系统1000类似，区别在于编排模块1030包含事件管理器2010而不是子模块OPM1040、核心处理OM 1050、SMS 1060和SPM 1070。下面将参照图20所示的分解模块1020和编排模块1030来描述编排的示意性实施例。然而，本领域技术人员将会容易地理解，所描述的实施例仅是示意性的实施例，这种编排可以根据可选实施例来实现，并仍包含在本发明的范围中。

[0203] 分解模块1020按照与之前讨论过的图10相似的方式来进行操作。编排模块1030控制着在处理并实施D00订单的同时发生的事件的顺序，其中该D00订单是由分解模块1020通过事件管理器2010的调用和事件发布来创建的。

[0204] 根据该实施例，分解模块1020调用编排模块1030的事件管理器2010。更特别地，分解模块1020调用事件管理器2010来编排由分解模块1020创建的D00订单。如前所述，事件管理器1910是无状态的处理，其能够被分解模块1020来调用。事件管理器2010包含嵌入式逻辑，被配置为确定一组要发布的事件。该逻辑是从存储在数据库中的元数据和处理状态中获得的。基于该逻辑，事件管理器2010产生并发布一组要传送给订购者的事件，例如任务层服务。事件管理器2010可以通过调用一种内部方法来发布一个事件。一旦事件管理器2010发布了该组事件，事件管理器2010就终止。在订购者消耗该组事件并执行基于该消耗的事件的任务之后，订购者就更新存储在数据库中的处理状态。分解模块1020再次调用事件管理器2010。重复该处理直到编排好D00订单，然后，事件管理器2010终止。

[0205] 下面将详细地描述事件消息发送系统。

[0206] 应用程序可以通过消息通道(即把发送器连接到接收器的虚拟通道)传输数据。消息是在消息通道上传输的数据分组。为了传输数据，发送器可将数据拆分为一个或多个分组，将每个分组打包为一条消息，然后在消息通道上传输该消息。同样地，接收器可以接收一条消息，从该消息中提取数据，并处理该数据。消息发送系统可以重复尝试把消息从发送器发送到接收器，直到成功为止。传输数据可包括在发送器处将数据排列为字节格式，将所排列的数据作为字节流从发送器传输到接收器，并把该数据去排列回其原始格式。

[0207] 即使在应用程序不知道哪个特定的接收器将会最终取回该应用程序，事件消息发送系统也能允许发送器传输消息。这是因为，当发送器传输数据时，发送器就将该数据添加到专用于传输这种类型的信息消息通道中。同样地，欲接收特定信息的接收器基于其所需的数据类型选择从什么消息通道获取数据。这样，发送器就能确保接收器取回数据中其所感兴趣的数据。

[0208] 消息可以包括两部分：首标和主体。首标可包括由消息发送系统使用的信息，该信息描述了正被传输的数据。主体可包括正被传输的数据。

[0209] 事件是一种表示发送器的状态发生改变的消息。这种状态改变可以基于外部对发送器进行动作而产生。例如，用户可以在一个订单捕获层中表示他想要创建一个运输。用户的表示是一种外部动作，它触发了编排层的状态的改变。作为这种状态的改变的结果，编排层可以使用消息通道向任务服务层传输一个事件，其中该事件表示用户想要创建一个运输。任务服务层可消耗该事件并执行一个任务，以基于该事件创建一个运输。

[0210] 事件的传输是单向的异步动作。这意味着传输事件的发送器不需要等待来自接收器的响应，其中该接收器订购该事件并消耗该事件。根据某些实施例，发送器在其发送了事

件之后就终止。

[0211] 图21示出了根据一个实施例的事件消息发送系统2100的一个实例。事件消息发送系统2100包括消息通道2110。如前所述,消息通道2110被配置为从发送器向接收器传输消息,例如事件。所示的实施例包括事件管理器2120和订购者2130。根据该实施例,事件管理器2120可使用消息通道2110来传输事件。在该实施例中,事件管理器2120通过消息通道2110将一个事件传输给订购者2130。订购者2130订购该事件,并因此接收该事件,订购者2130执行基于该事件的任务。

[0212] 虽然图21示出了事件消息发送系统2100具有单个消息发送通道,但这只是一个示意性实施例,本领域技术人员将会容易地理解,事件消息发送系统2100可以包括任意数量的消息发送通道,并仍包含在本发明的范围内。

[0213] 消息通道的类型有许多种。根据本发明的一个实施例,分布式订单编排系统使用一个事件消息发送系统,该事件消息发送系统包括一个或多个发布-订购消息通道。现在将详细描述发布-订购消息通道。

[0214] 图22示出了根据一个实施例的事件消息发送系统的发布-订购消息通道2200的一个实例。发布-订购消息通道2200是使一个输入通道被分割为多个输出通道(每个输出通道用于一个订购者)的消息通道。所示的实施例包括一个事件管理器220以及订购者2220、2230和2240。根据该实施例,事件管理器2210把事件2250发布到发布-订购消息通道2200的一个输入通道中。发布-订购消息通道2200然后将事件2250的一个拷贝传输给每个输出通道。根据该实施例,每个输出通道具有一个订购者(即第一输出通道具有订购者2220,第二输出通道具有订购者2230,第三输出通道具有订购者2240)。每个订购者监听事件2250并消耗事件2250的相应的拷贝。一旦事件2250的每个拷贝都被相应的订购者所消耗,那么事件2250的每个拷贝就从发布-订购消息通道2200中消失。

[0215] 虽然图22示出了发布-订购消息通道2200具有三个输出通道和三个订购者,但这只是一个示意性实施例,本领域技术人员将会容易地理解,发布-订购消息通道2200可以包括任意数量的输出通道和订购者,并仍包含在本发明的范围内。而且,虽然在图22所示的实施例中对所有订购者2220、2230和2240发布事件2250,但这只是一个示意性实施例,本领域技术人员将会容易地理解,可以向有效的订购者的子集,甚至是单个订购者发布事件。

[0216] 图23示出了根据一个实施例在分布式订单编排系统中管理事件的方法的流程图。提供了编排层、任务层、外部接口层和外部系统层。在一个实施例中,在编排层之前提供一个分解层(未示出)。在包括分解层的实施例中,分解层可以调用一个服务,用以创建处理状态和元数据并将它们存储在数据库中。

[0217] 根据该实施例,步骤2302产生一个事件并发布该事件。可以从订单捕获模块接收订单。这将导致事件被发布。使用数据库中的处理状态和元数据产生该事件。该事件被发送到任务层。

[0218] 根据该实施例,步骤2304启动该任务,产生一个用于外部系统的消息并发送该消息。所产生的消息表示哪个任务应当由外部系统来执行。要执行的任务是已模型化的订单处理的一个方面。例如,可为一个订单调用该任务。也包含执行该任务所需的参数。消息被发送到一个外部接口。

[0219] 根据该实施例,步骤2306对消息进行转换并将其发送给该外部层。由任务层产生

的消息可以是一种通用的格式。然而不同的外部系统可以使用其它的格式进行通信。该外部接口层确定外部系统所使用的格式并转换该消息。例如，用户定义的元数据可用于确定要使用的格式。在一个实例中，对哪些系统调用了所订购的产品的映射，被用来翻译该消息。

[0220] 根据该实施例，步骤2308接收由该外部系统返回的消息并处理该产生了一个结果的消息。该外部系统可以是执行与处理订单相关的任务的系统，例如调度系统、运输系统等等。当执行任务时，确定任务的结果。结果可以是安排好的运输日期，运输货物的日期等。结果然后被发送回该外部接口层。

[0221] 在该实施例中，步骤2310转换该消息并将其发送到任务层。步骤2312基于结果更新数据库中的处理状态。然后处理进行到下一个可被产生的事件。重复处理直至处理完成。

[0222] 图24示出了根据一个实施例的确定一组要发布的事件的方法的流程图。根据一个实施例，该方法可在调用事件管理器时由事件管理器来实施，并且该方法可确定事件管理器要为具体处理发布哪组事件。

[0223] 在2400，开始该方法。在2410，确定是否有处理步骤“剩余”。换言之，确定是否有处理步骤还未执行。根据一个实施例，这种确定是基于存储在数据库中的处理状态和元数据来做出的。根据一个实施例，处理状态表示哪些处理步骤已被执行，元数据表示该处理的所有处理步骤。根据该实施例，如果一个处理步骤还未执行，但之前已被当前的实现方法所选择，那么处理步骤就不能被认为是“剩余”，并且不可以被选择。如果没有处理步骤剩余，方法就在2460结束。如果剩余了一个或多个处理步骤，那么方法就前进到2420。

[0224] 在2420，选择处理步骤。根据一个实施例，基于存储在数据库中的处理状态和元数据来选择处理步骤。例如，可以从处理状态和元数据中来确定哪些处理步骤已被执行。这样，就可以根据该方法选择尚未执行的处理步骤。根据一个实施例，处理状态表示哪些处理步骤已被执行，元数据表示该处理的所有处理步骤。根据该实施例，之前已被当前的实现方法所选择的处理步骤不可以被选择。

[0225] 在2430，确定所选择的处理步骤是否依赖于其它的处理步骤。根据一个实施例，确定基于存储在数据库中的元数据来确定所选择的处理步骤是否依赖于其它的处理步骤。例如，如果选择了处理步骤4，就可以确定处理步骤4只能在处理步骤1、2和3完成之后再执行。可选地，可以确定处理步骤4可以在任何时间来执行，而不考虑步骤1、2和3是否完成。如果确定出所选择的处理步骤不依赖于其它的处理步骤，那么方法就前进到2440。如果确定出处理步骤依赖于其它的处理步骤，那么方法就前进到2450。

[0226] 在2440，基于所选择的处理步骤产生一个事件并发布该事件。根据该实施例，通过发布该事件，该事件被传输给一个或多个订购者。一个或多个订购者可以消耗该事件并执行一个基于该所消耗的事件的任务，其中该消耗的事件与该处理步骤相对应。

[0227] 在2450，确定所选择的处理步骤所依赖的其它处理步骤是否已完成。根据一个实施例，基于数据库中存储的元数据确定其它处理步骤是否已完成。例如，如果选择了处理步骤4，并且处理步骤4依赖于步骤1、2和3，就确定处理步骤1、2和3是否已完成。如果确定出其它处理步骤已完成，那么方法就前进到2440，并且如上所述产生一个事件并将其发布。然而，如果确定出其它处理步骤尚未完成，那么方法就返回2410，这是因为在其它处理步骤完成之前不能执行所选择的处理步骤。

[0228] 在2460,方法结束。然而,每次调用事件管理器时可以实施该方法。可以调用事件管理器直到所有的处理步骤都已执行,这样,就可以每次实施该方法直到所有的处理步骤都已执行。根据该实施例,一旦为一个处理步骤产生一个事件并将其发布,在随后实施该方法时该处理步骤就不可以被选择。然而,如果选择了一个处理步骤,但不对该处理步骤产生事件并将其发布(例如因为该处理步骤依赖于未完成的其它处理步骤),那么在随后实施该方法时该处理步骤就可以被选择。

[0229] 根据一个实施例,数据库包含并行控制,以便如果两个或多个步骤同时完成,能正确地确定出两个或多个步骤已完成。因此,根据该实施例,可以基于数据库的处理状态和元数据来确定所选择的处理步骤所依赖的其它处理步骤是否已完成。

[0230] 如前所述,分布式订单编排系统的用户可以定义一个要被调用的服务的序列,这些服务使用并行分支来影响业务处理规则。当用户为一个处理步骤选择一个服务时,该用户可以提供附加的元数据,用以在运行时间上处理该订单期间确定该处理数据如何被执行。在某个实施例中,可以定义并行分支。根据一个实施例,事件管理器可被用于为并行的处理步骤发布事件。根据该实施例,如果事件管理器基于存储在数据库中的处理状态和元数据确定出两个或多个处理步骤要并行地执行,该事件管理器就同时发布两个或多个事件。下面将更加详细地描述事件管理器为并行处理步骤发布事件的一个实例。

[0231] 图25示出了根据一个实施例的事件管理器2500为并行处理步骤发布事件的一个实例。根据该实施例,事件管理器2500基于存储在数据库(未示出)中的处理状态和元数据,确定要并行运行的三个处理步骤。事件管理器2500产生三个事件,事件2501、2502和2503,并同时发布这三个事件。订购者2510消耗事件2501并执行基于所消耗的事件的任务,订购者2520消耗事件2502并执行基于所消耗的事件的任务,订购者2530消耗事件2503并执行基于所消耗的事件的任务。这样,与事件2501、2502和2503相应的任务分别由订购者2510、订购者2520和订购者2530并行执行。

[0232] 虽然在该实例中,有三个处理步骤并行运行,但本领域技术人员将会容易地理解,这只是一个实例,任意数量的步骤都可以并行地运行。因此,在可选实施例中,事件管理器2500可以为任意数量的并行处理发布事件。

[0233] 在一个可选的实施例中,图25所示的三个处理步骤是条件步骤而不是并行步骤。如前所述,条件步骤是依赖于结果的发生的步骤(例如另一个步骤的结束)。根据该实施例,事件管理器2500基于存储在数据库(未示出)中的处理状态和元数据,确定要执行三个处理步骤中的哪一个。事件管理器2500产生事件2501、事件2502或事件2503,并发布该事件。当事件管理器2500发布事件2501时,订购者2510消耗事件2501并执行基于所消耗的事件的任务。同样地,当事件管理器2500发布事件2502时,订购者2520消耗事件2502并执行基于所消耗的事件的任务。相似地,当事件管理器2500发布事件2503时,订购者2530消耗事件2503并执行基于所消耗的事件的任务。

[0234] 仍如前所述,分布式订单编排系统的用户可指示一个处理要被分割为两个或多个处理。根据一个实施例,事件管理器可用于为分割的处理步骤发布事件。根据该实施例,如果事件管理器基于存储在数据库中的处理状态和元数据来确定一个处理要被分割为两个或多个处理,该事件管理器就产生一组用于一个或多个分割单元的事件,并发布该组事件。下面将详细描述事件管理器分割处理和发布相应的一组事件的一个实例。

[0235] 图26示出了根据一个实施例的事件管理器2600分割一个处理的一个实例。根据该实施例，事件管理器2600基于存储在数据库(未示出)中的处理状态和元数据，确定当前的处理要被分割为两个处理。事件管理器2600确定处理的剩余步骤并创建两个分割单元，分割单元A和分割单元B。如前所定义的，分割单元定义了可被一起分割的处理中的一组连续的步骤。然后事件管理器2600产生一组事件，事件2601、2602、2603和2604，这里事件2601和2602是分割单元A的一部分，事件2603和2604是分割单元B的一部分，每个事件与处理的一个剩余的步骤相对应。然后事件管理器2600发布事件2601、2602、2603和2604。根据一个实施例，事件管理器2600顺序地发布事件2601、2602、2603和2604。在另一个实施例中，事件管理器2600并行地发布作为分割单元A的一部分的一个事件以及作为分割单元B的一部分的一个事件，然后再并行地发布作为分割单元A的一部分的另一个事件以及作为分割单元B的一部分的另一个事件。在第三个实施例中，事件管理器2600并行地发布事件2601、2602、2603和2604。

[0236] 根据一个实施例，订购者2610消耗事件2601并执行基于所消耗的事件的任务。另外，订购者2620消耗事件2602和事件2603，并执行基于每个所消耗的事件的任务。而且，订购者2630消耗事件2604并执行基于所消耗的事件的任务。

[0237] 根据一个实施例，当处理被分割时，由事件管理器2600产生的事件相比于不分割处理时所产生的事件，可以具有不一样的有效负载。根据该实施例，不同的有效负载可以标识与分割处理的一个步骤相对应的事件。

[0238] 虽然在该实例中，产生了四个事件，其中该四个事件与处理的四个步骤相对应，但本领域技术人员将会容易地理解，这只是一个实例，可以相应于任意数量的处理步骤来产生任意数量的事件。另外，虽然在该实例中，对三个订购者发布四个事件，但本领域技术人员也将理解，这也只是一个实例，所产生的事件可以发布给任意数量的订购者，每个订购者可以消耗任意数量的事件。并且，虽然在该实例中，处理被分割为两个分割单元，但这只是一个实例，处理可被分割为任意数量的分割单元。因此，事件管理器2600可以将一个处理分割为任意数量的分割单元，其中每个分割单元包括任意数量的剩余步骤。而且，事件管理器2600能够产生任意数量的事件，并能够将这些事件发布给任意数量的订购者。

[0239] 仍如前所述，分布式订单编排系统的用户可以启动一个修改请求。如前所述，修改请求包括引用一个原始订单的新订单。该新订单还包括对原始订单的修正，因此也就包括对含有原始处理的业务步骤的修正。该原始处理可以是正在执行的，或者之前已被执行过。根据一个实施例，事件管理器可用于处理一个修改请求。根据该实施例，如果事件管理器基于存储在数据库中的处理状态和元数据确定出用户已启动一个修改请求，那么事件管理器就产生一组事件，每个事件用来补偿原始处理的一个步骤。下面将详细描述事件管理器使用事件来处理修改请求的一个实例。

[0240] 图27示出了根据一个实施例的事件管理器2700使用事件来处理修改请求的一个实例。根据该实施例，事件管理器2700基于存储在数据库(未示出)中的处理状态和元数据来确定已经启动一个修改请求。基于处理状态和元数据，事件管理器2700产生一组事件以用于补偿一个原始处理。在图27所示的实施例中，事件管理器2700产生事件2701、2702和2703，每个事件都与原始处理的一个步骤相对应。根据该实施例，事件管理器2700每次发布该组事件中的一个事件。这是因为原始处理的步骤是顺序补偿的，而不是并行的。在所示的

实施例中,事件管理器2700首先将事件2701发布给订购者2710。订购者2710消耗该事件并基于所消耗的事件执行一个补偿原始处理的第一步骤的任务。接着,事件管理器2700将事件2702发布给订购者2720。订购者2720消耗该事件并基于所消耗的事件执行一个补偿原始处理的第二步骤的任务。然后,事件管理器2700将事件2703发布给订购者2730。订购者2730消耗该事件并基于所消耗的事件执行一个补偿原始处理的第三步骤的任务。

[0241] 虽然在该实例中产生了三个事件,其中这三个事件与一个原始处理的三个步骤相对应,但本领域技术人员将会容易地理解,这只是一个实例,可以相应于原始处理的任意数量的步骤来产生任意数量的事件。而且,虽然在该实例中,对三个订购者发布三个事件,但这只是一个实例,一组事件可以发布给任意数量的订购者。

[0242] 图28示出了根据一个实施例的分布式订单编排模块的功能的处理图。根据该实施例,分布式订单编排模块启动事件管理器用于使用一个处理来处理一个订单。根据该实施例,事件管理器执行图28所示的功能。

[0243] 在2810,基于存储在数据库中的处理状态和元数据来确定要执行的一个处理步骤。在2820,基于所确定的处理步骤来产生一个事件。根据该实施例,事件包括用于执行与该处理步骤相应的任务的指令。在2830,在事件消息发送系统上发布该事件。

[0244] 这样,根据本发明的一个实施例,分布式订单编排系统的事件管理器可以基于存储在数据库中的处理状态和元数据来产生并发布一组事件。一组订购者可以消耗该组事件,每个订购者可以执行基于所消耗的事件的任务。

[0245] 根据该实施例,分布式订单编排系统中的组成部分可以互相去耦合,这是因为事件管理器可以处理这些组成部分之间的通信。分布式订单编排系统中的计算可以与通信相分离。因此,分布式订单编排系统的每个组成部分或每个层可以包括较少的源代码,这是因为每个组成部分的源代码可以只包括完成该组成部分的目的的功能,并且不需要包括处理与其它组成部分通信的源代码。而且,可以对每个组成部分的源代码进行修改,例如修改事件的有效负载,而不需要修改源代码以修改该组成部分怎样与其它组成部分进行通信。另外,由于事件管理器是无状态的,因此产生并保留较少的数据。因此,根据该实施例,分布式订单编排系统可以更加可伸缩和灵活。

[0246] 尽管上面是参照这里的特定实施例来进行的描述,但这些特定实施例仅是示意性的并且是非限定性的。尽管描述了BPEL,但可以理解其它的业务项目语言也可以使用。

[0247] 任意适当的编程语言都可用于实现特定实施例的例程,它们包括:C、C++、Java、汇编语言等。可以采用不同的编程技术,例如过程的或者面向对象的。例程可在多个处理器上执行。尽管以具体次序给出步骤、操作或计算,但是该次序可以在不同的特定实施例中进行修改。在某些特定实施例中,在说明书中顺序示出的多个步骤是可以同时执行的。

[0248] 可以在利用指令执行系统、装置、系统或设备使用或与它们连接的计算机可读介质中实现这些特定实施例。能够以软件或硬件或它们的组合中的控制逻辑的形式来实施特定实施例。当该控制逻辑被一个或多个处理器所执行时,可操作地执行在特定实施例中描述的内容。

[0249] 可以使用已编程的通用目的数字计算机、使用应用程序专用集成电路来实现特定实施例,可以使用可编程逻辑设备、现场可编程门阵列、光、化学、生物、量子或纳米设计的

系统、元件和机制。通常，特定实施例的功能可由本领域所公知的任意方式来实现。可以使用分布式、网络系统、元件和/或电路。数据的通信或传输可以是有线的、无线的或通过任意其它的方式。

[0250] 可以理解，在附图中描述的一个或多个元件也可以采用更加分离或集成的方式来实现，或者在某些情形下甚至作为不能操作的元件进行移除或放弃，尽管它在某个特定的应用中是有用的。实施存储在机器可读介质中的程序或代码，以便允许计算机执行上述任意一种方法，这也落入本发明的精神和范围内。

[0251] 如在说明书和整个权利要求书所使用的，除非上下文中有明确的描述，“一个”、“该”以及“所述”包括复数个引用。而且，如在说明书和整个权利要求书所使用的，除非上下文中有明确的描述，“在……中”意味着“在……中”或“在……上”。

[0252] 因此，虽然这里已描述了特定实施例，但是容许在之前的描述中进行修改、各种改变以及替换，可以理解，也可以在在某些实例中采用特定实施例的某些特征而不用相应地使用其它的特征，这不会背离所表述的范围和精神。因此，可以做出许多修改以便使特定的情况和材料适应基本的范围和精神。

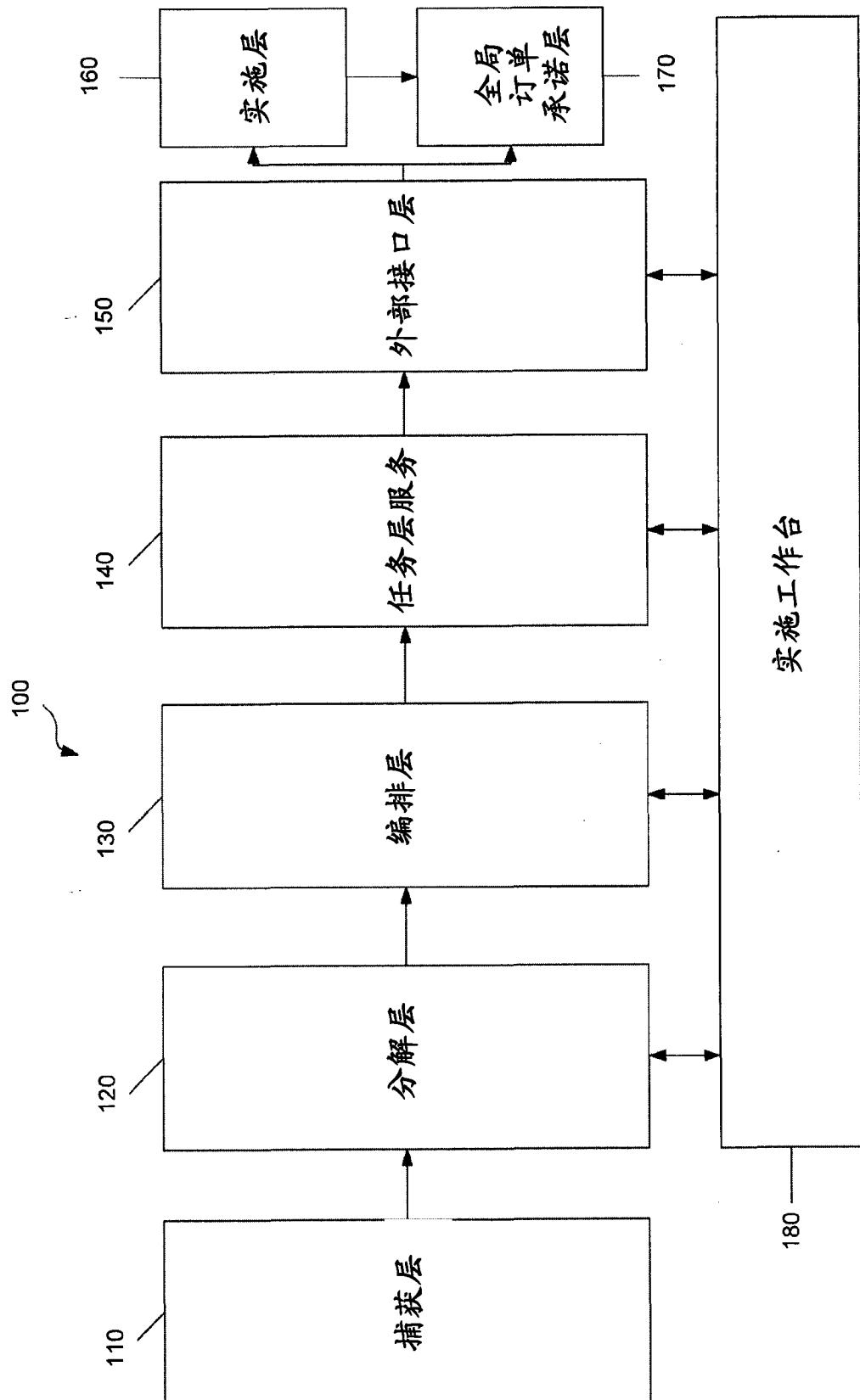


图 1

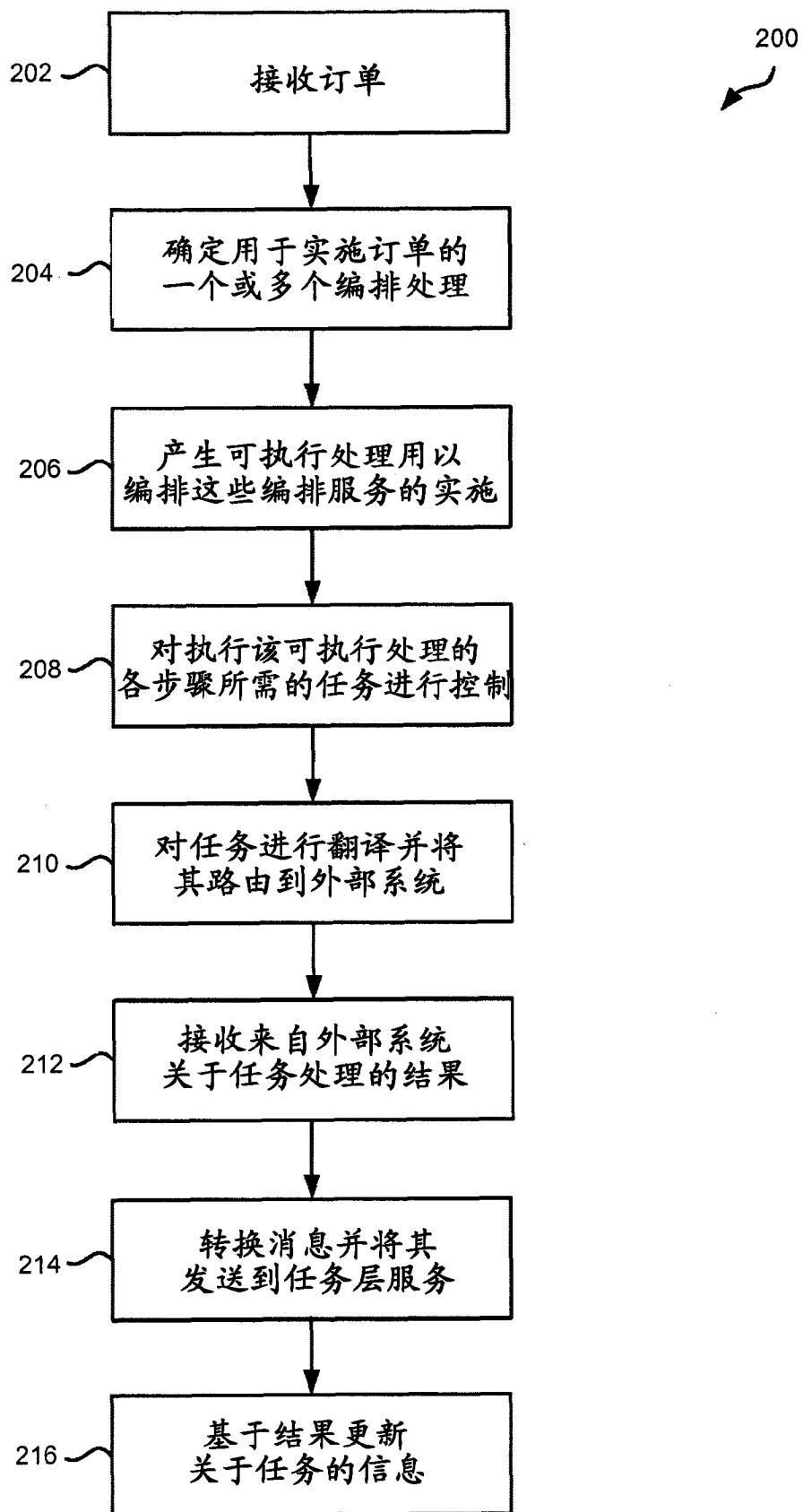


图2

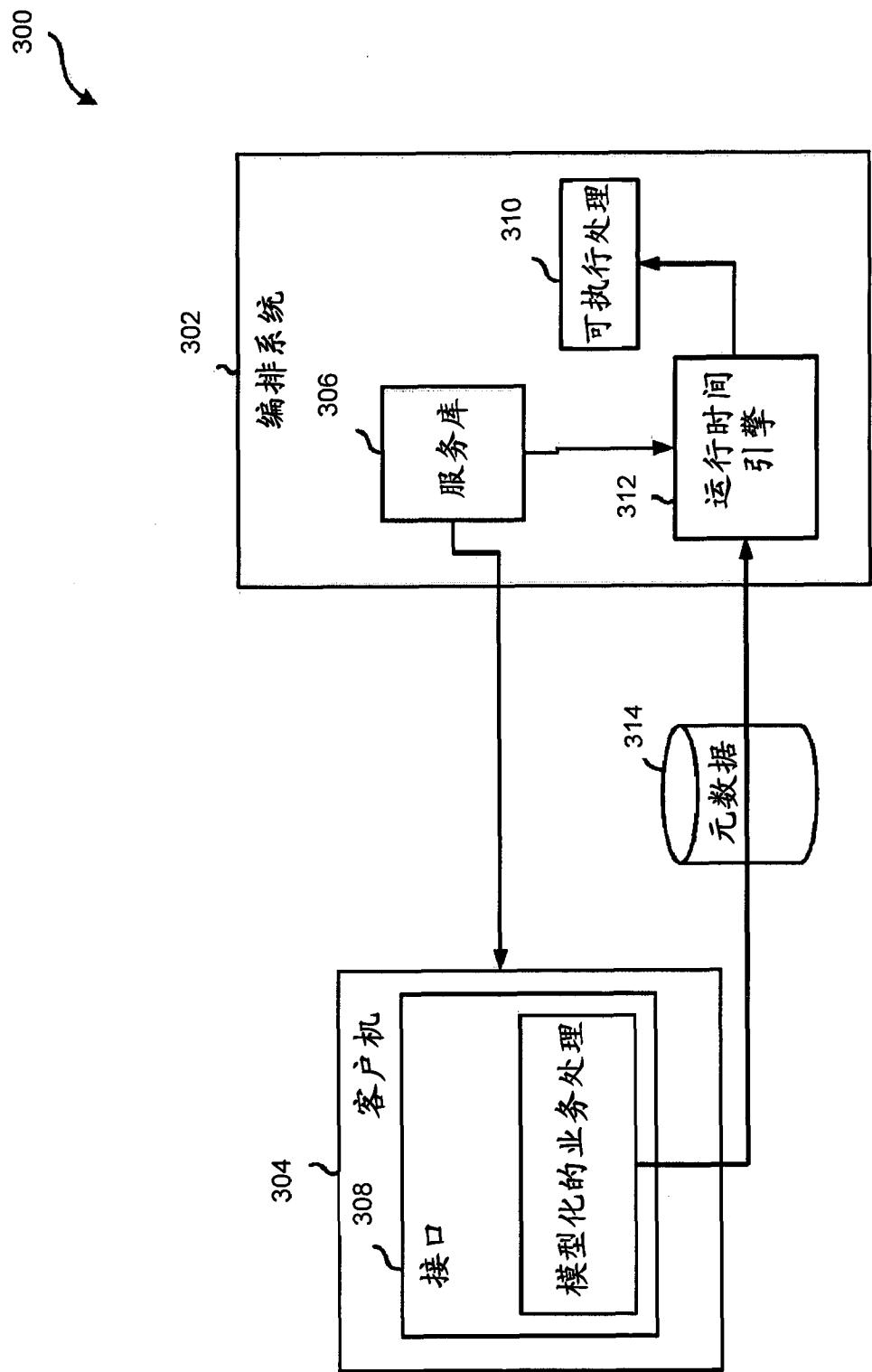


图3

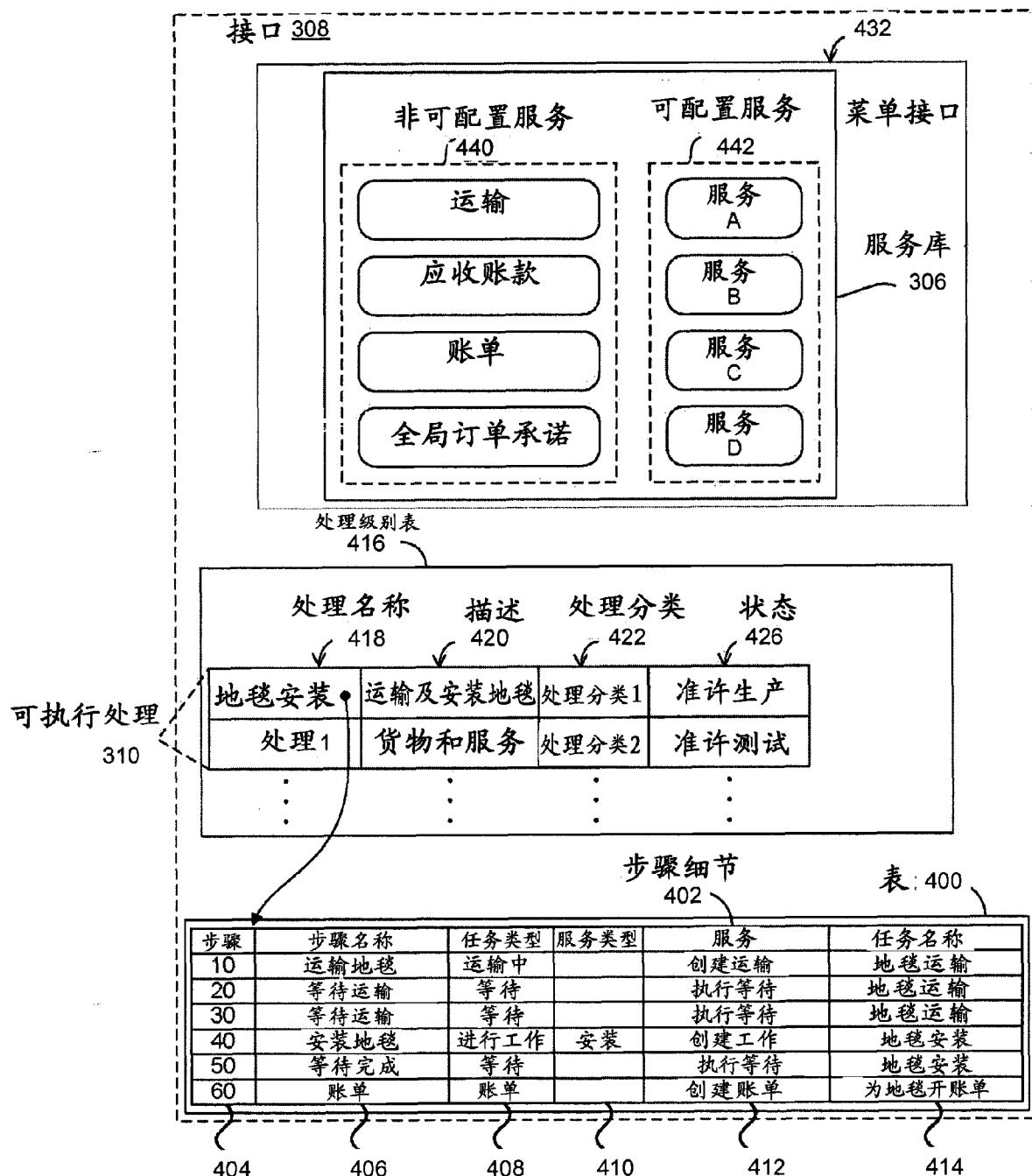


图4

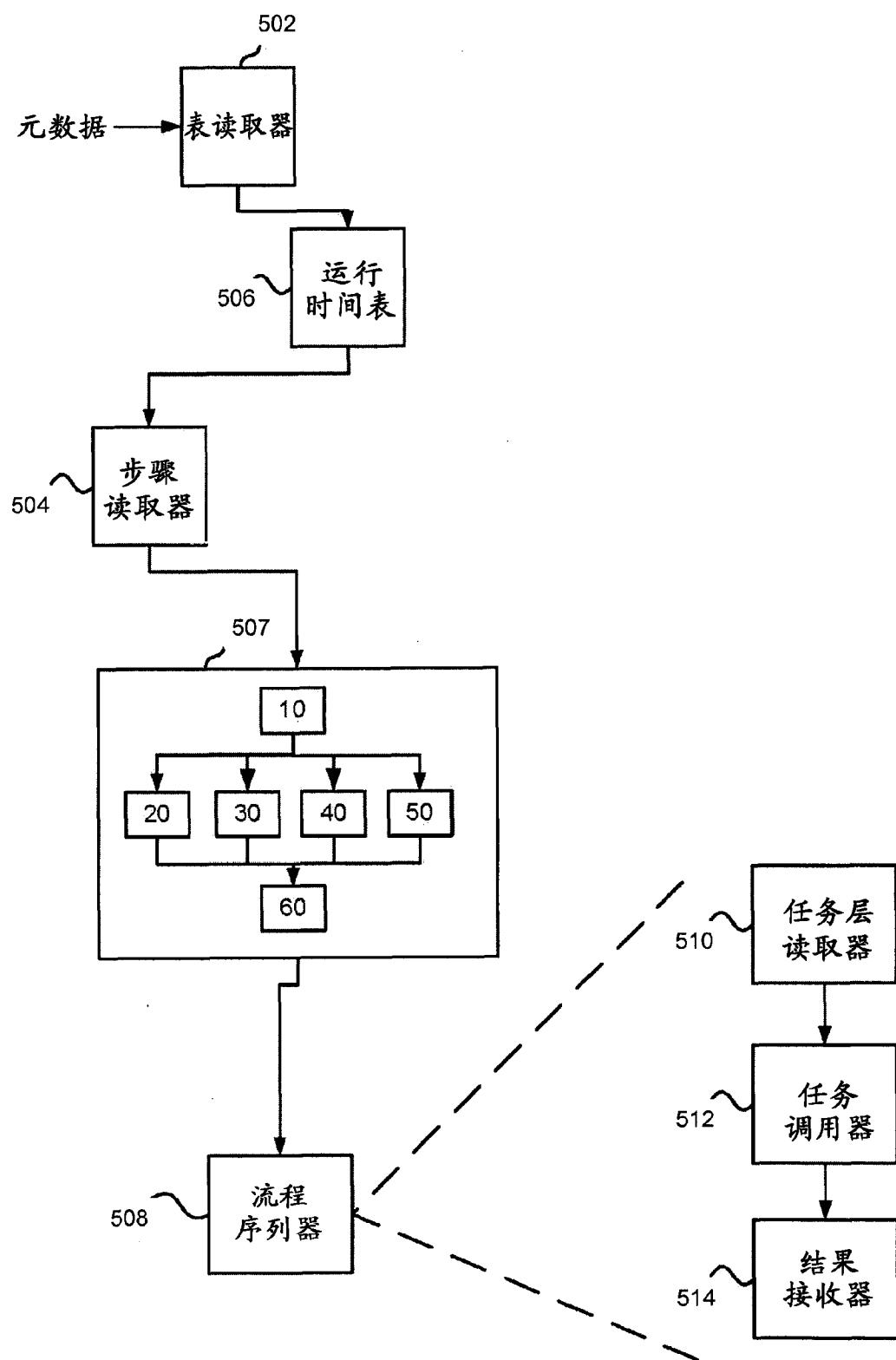


图5

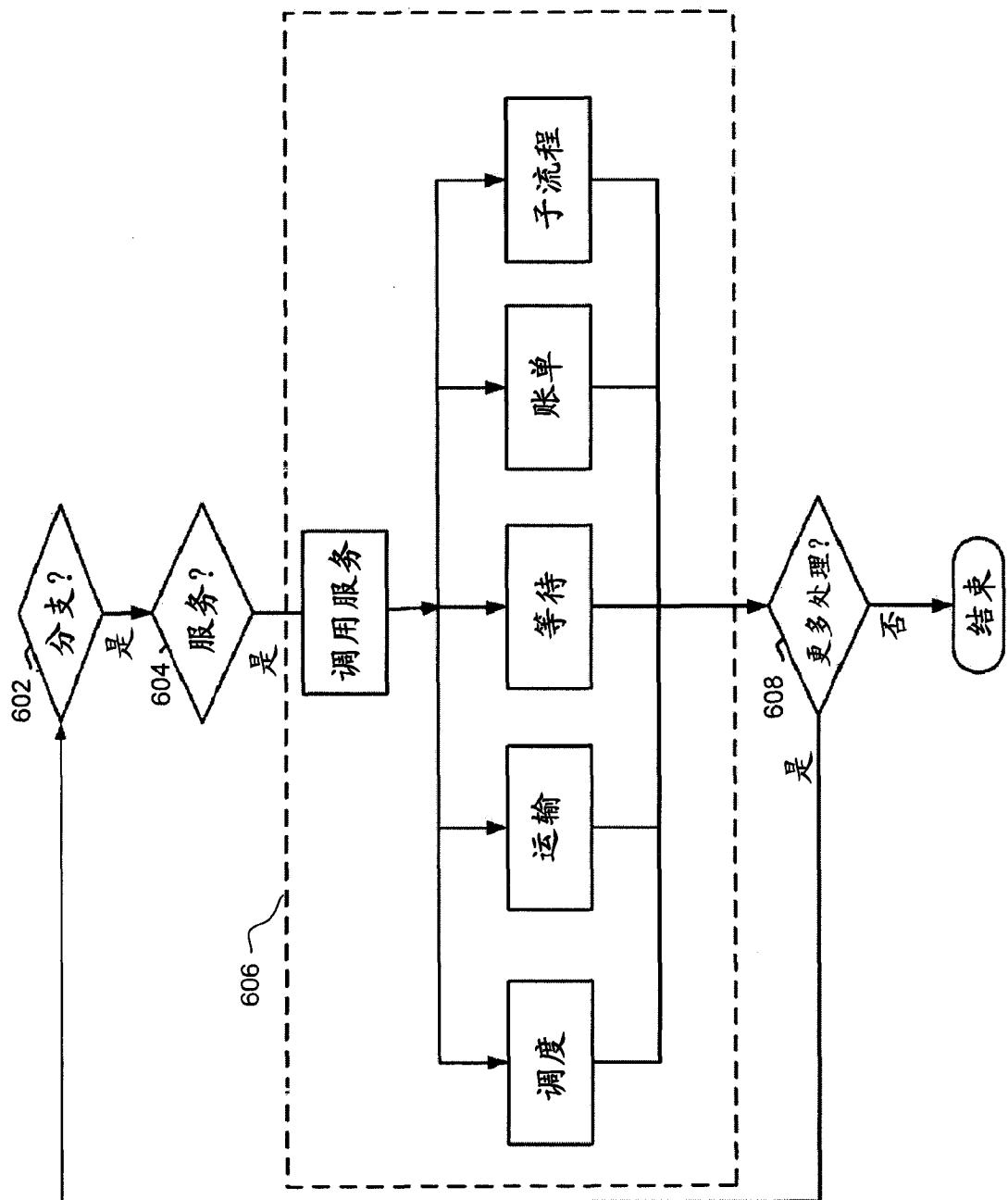


图6

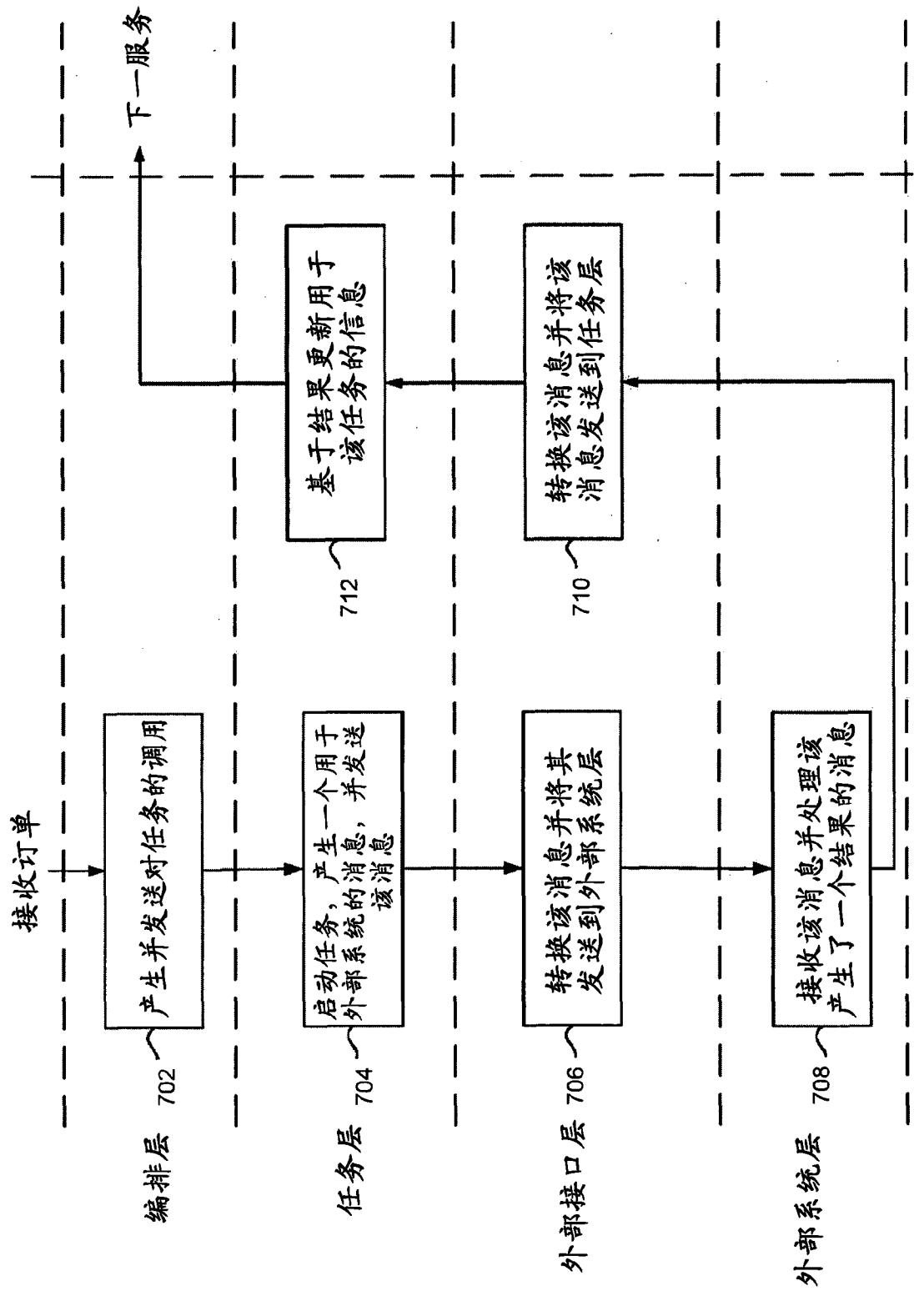


图 7

800

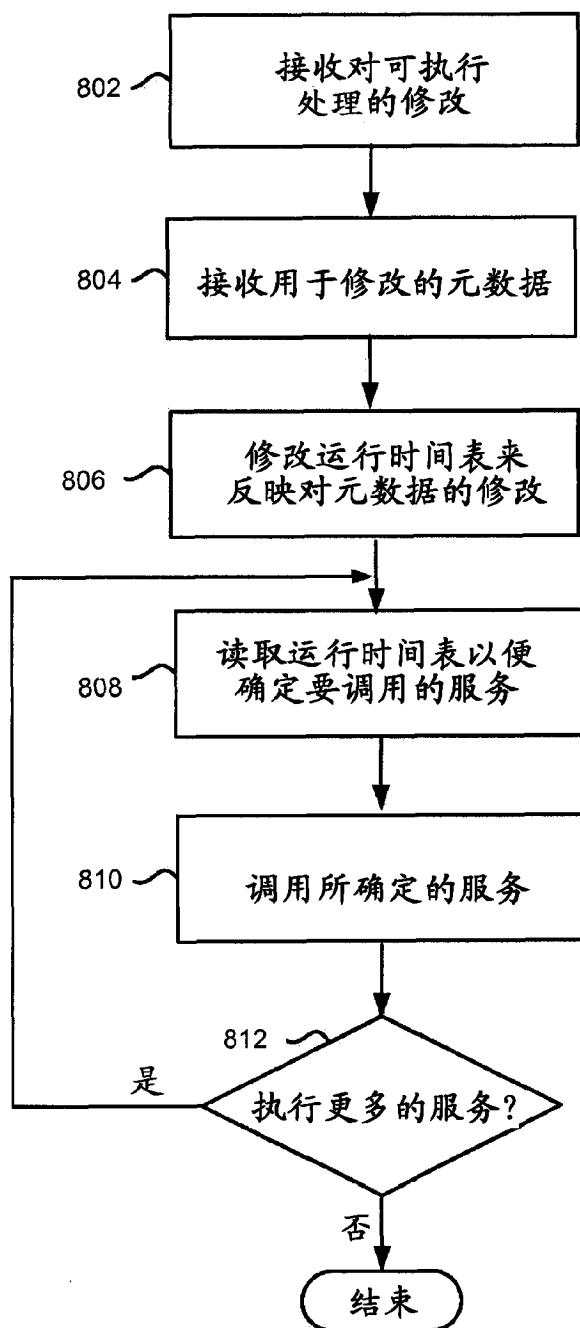


图8

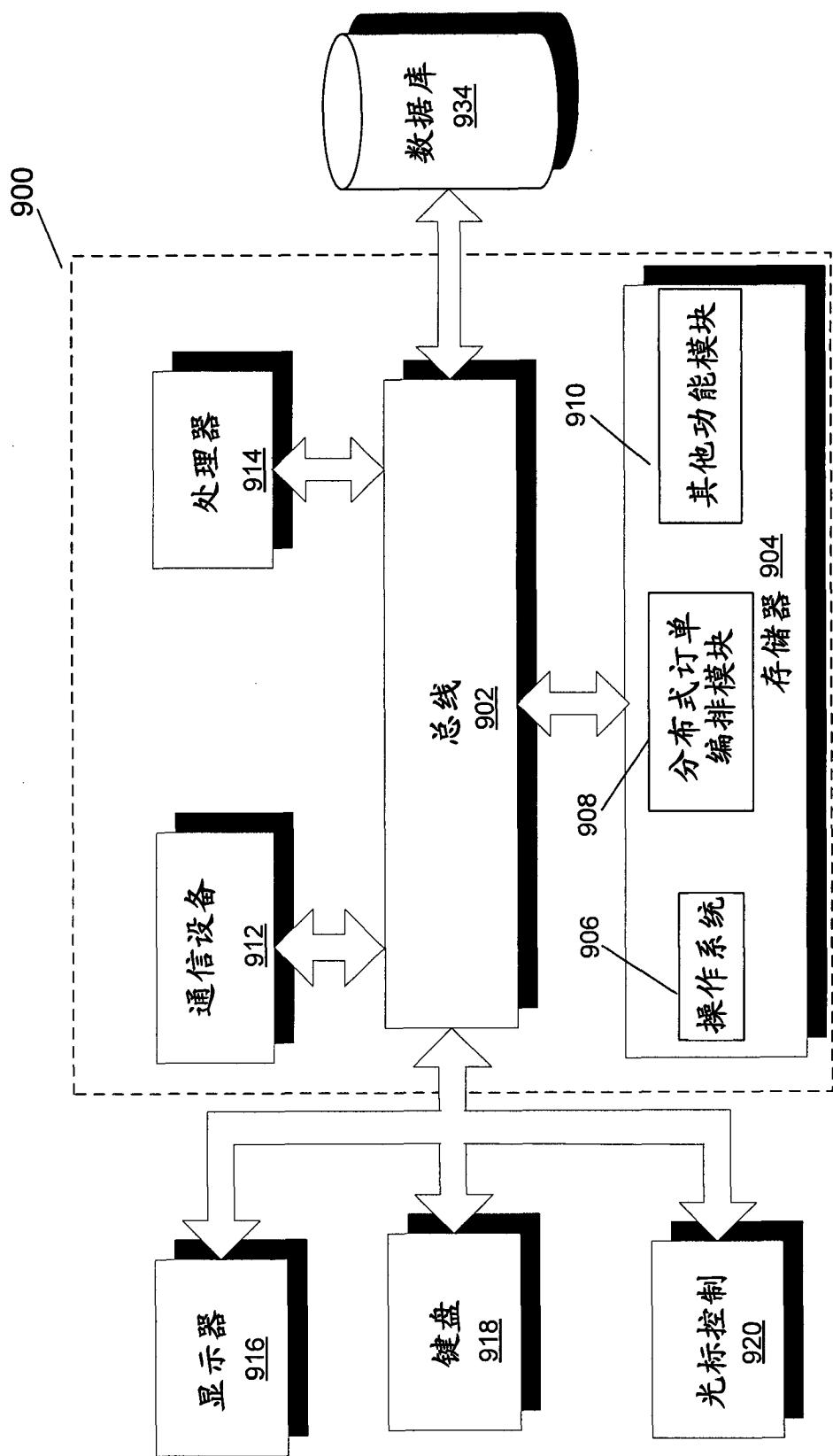


图9

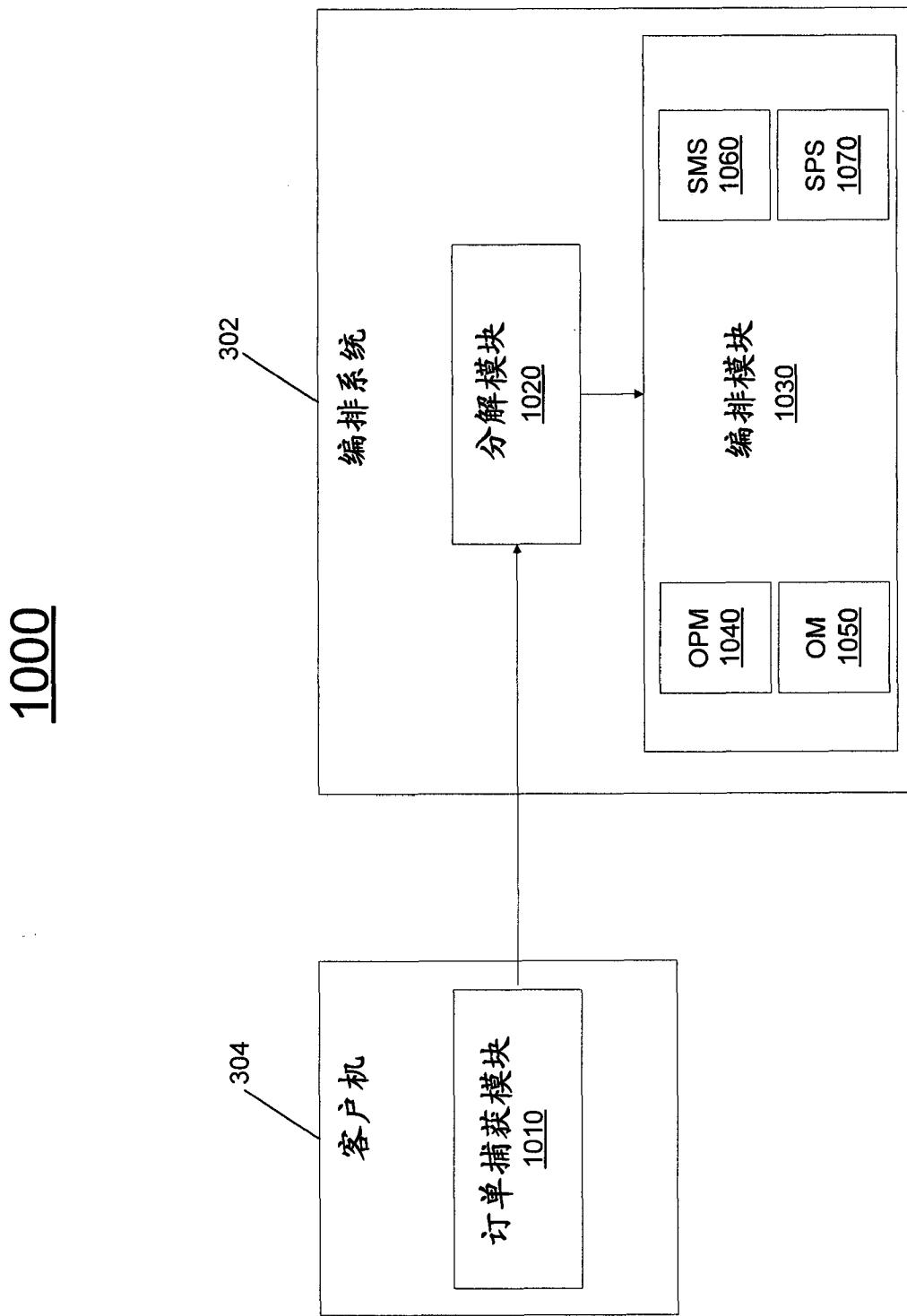


图10

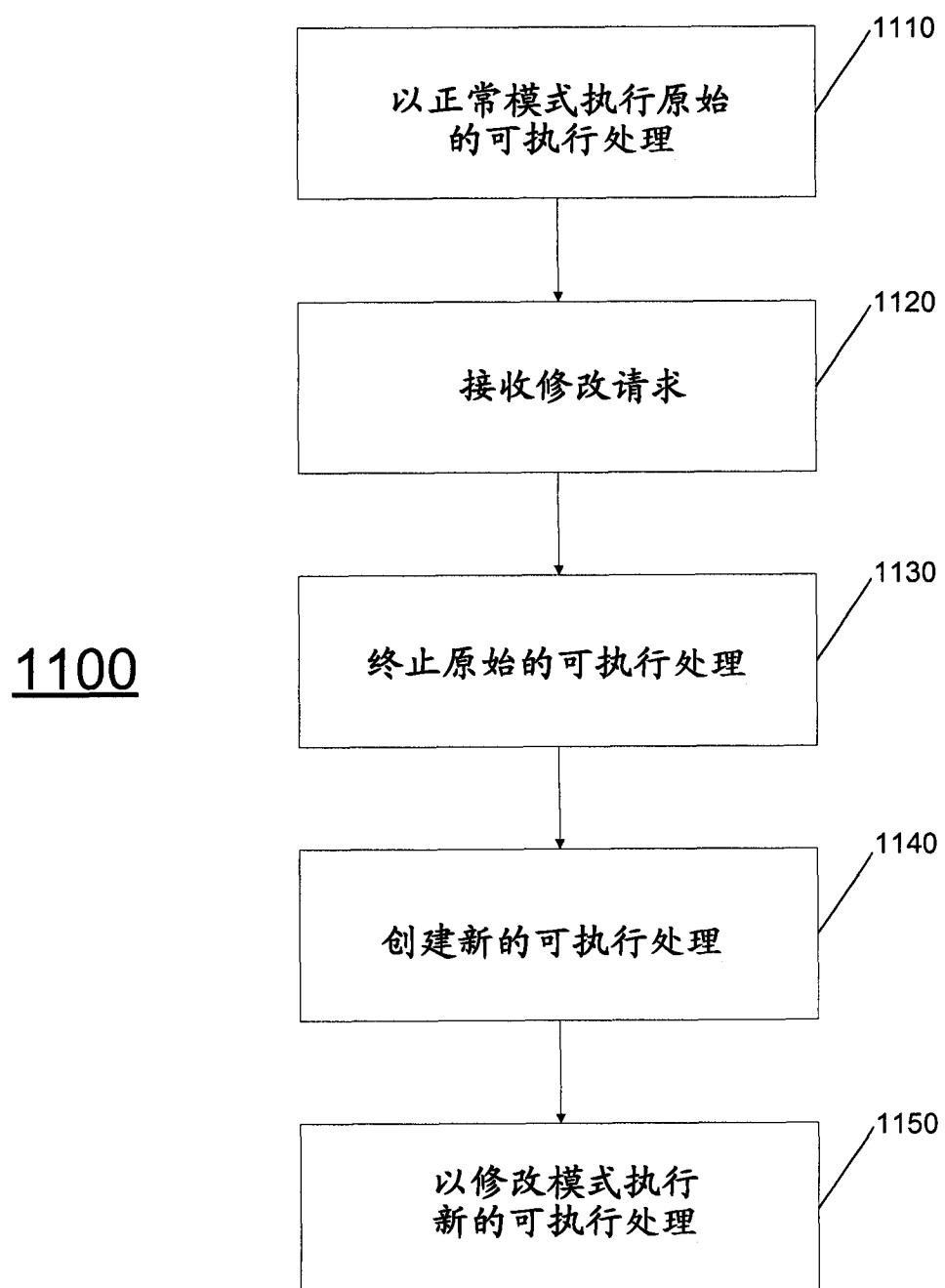


图11

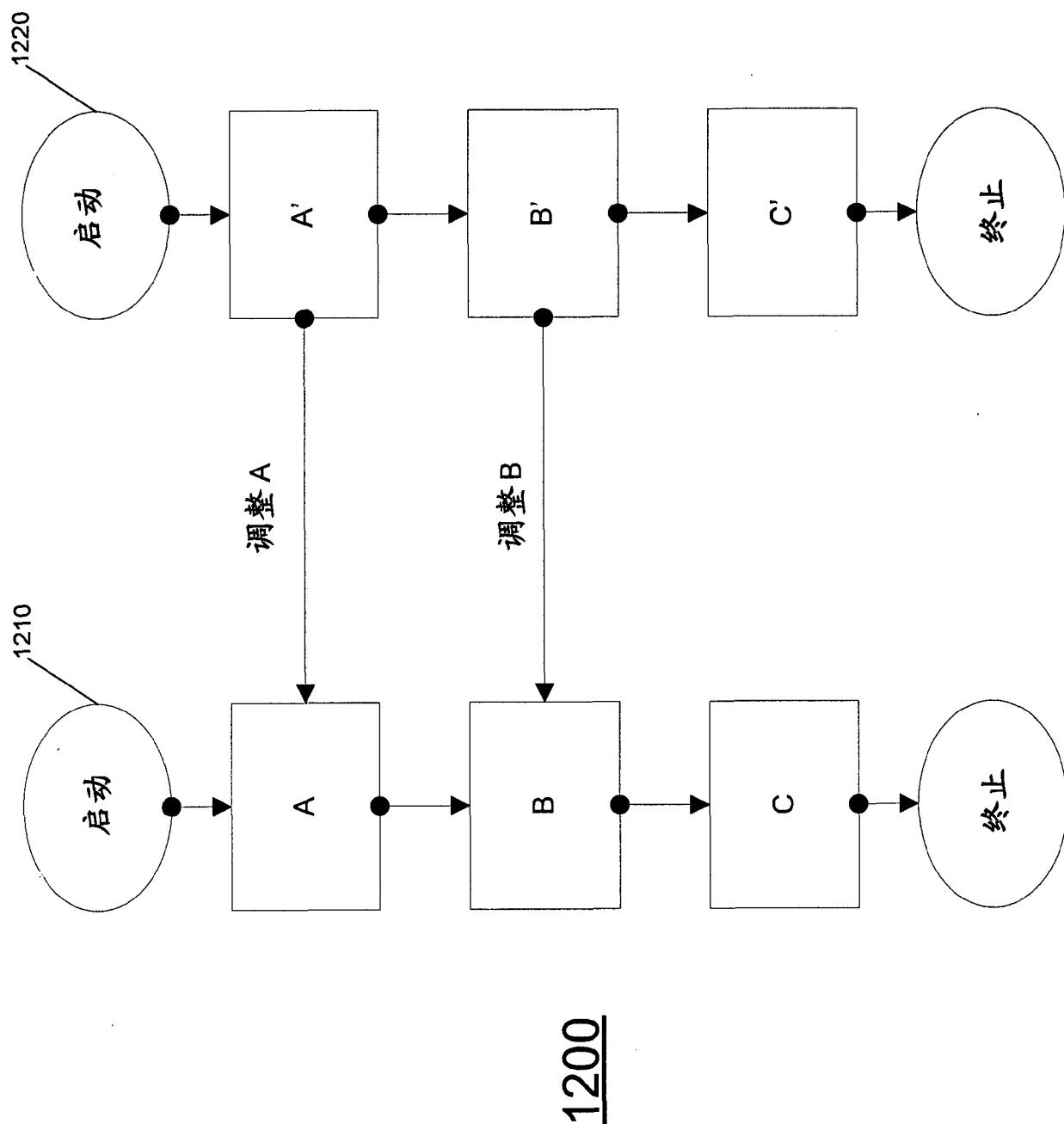
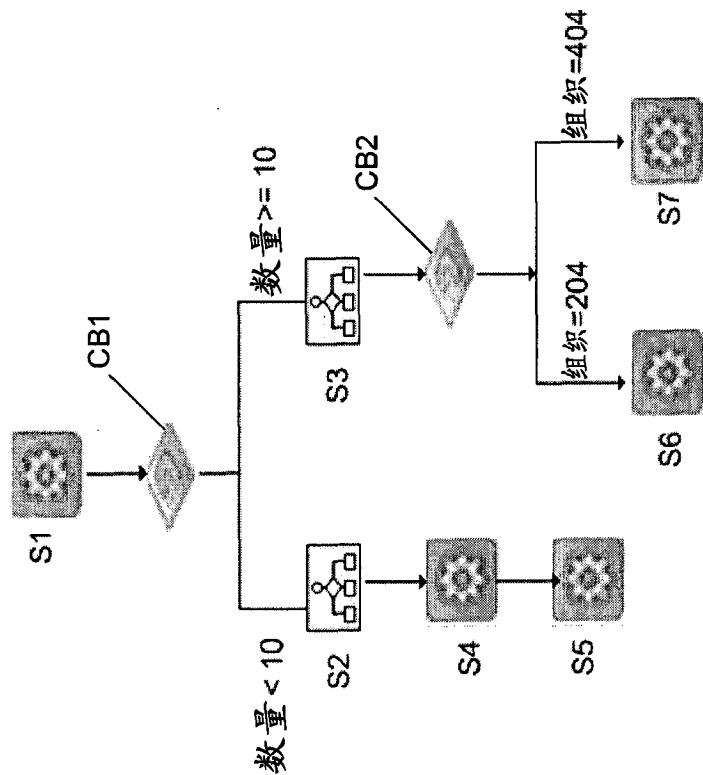


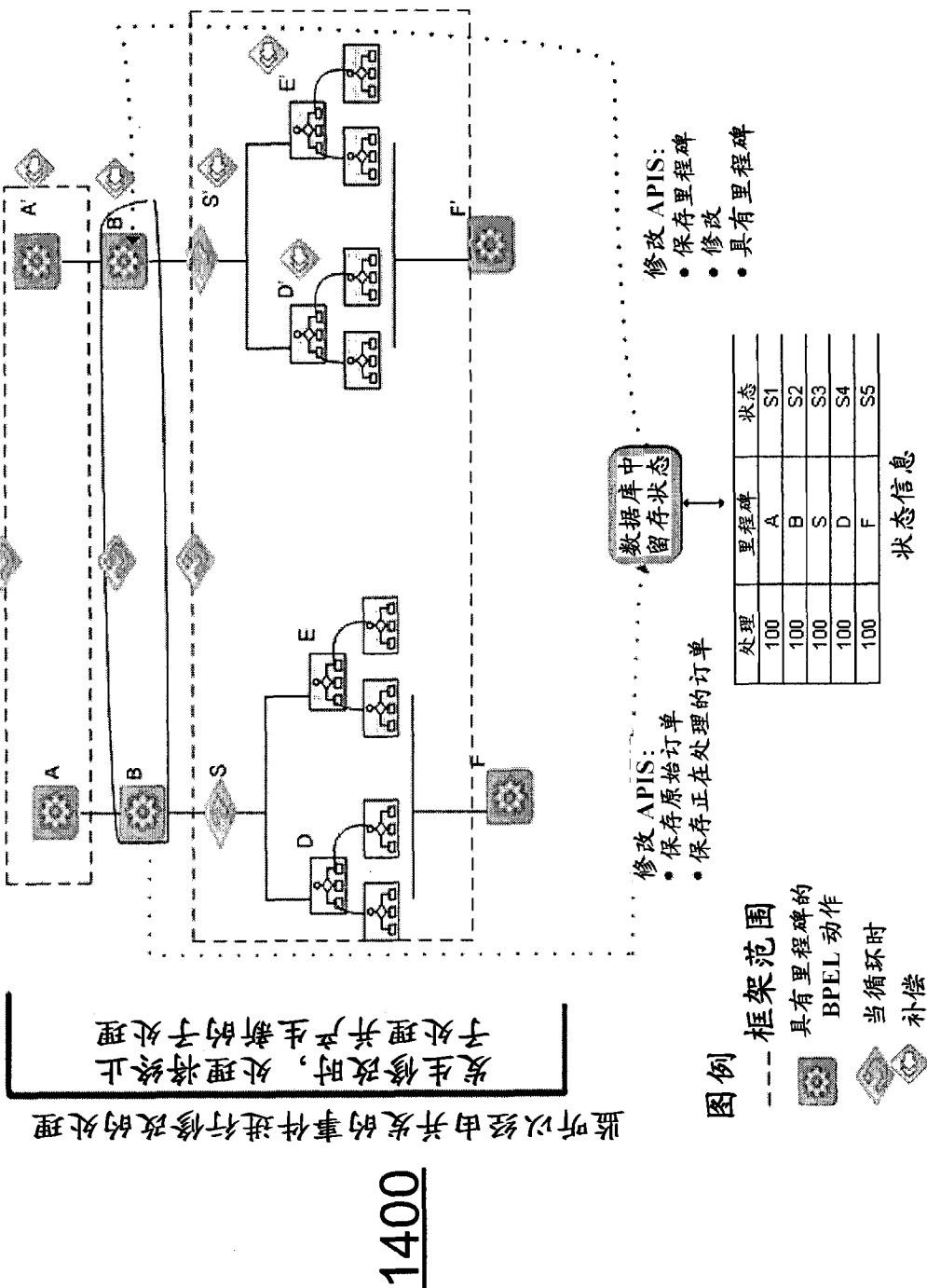
图12



1300

图13

**正常处理实例**



图例

1400

图14

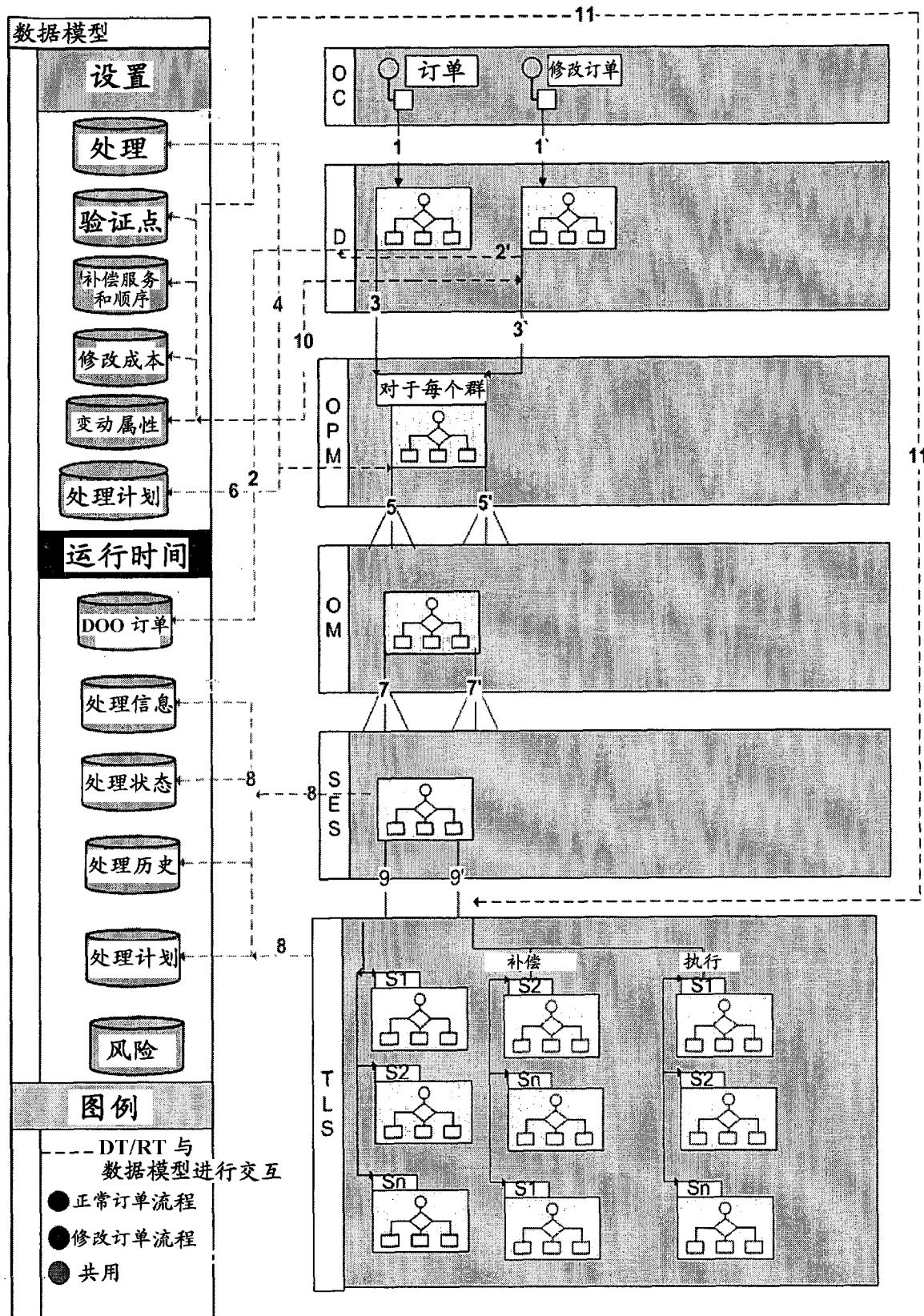


图15

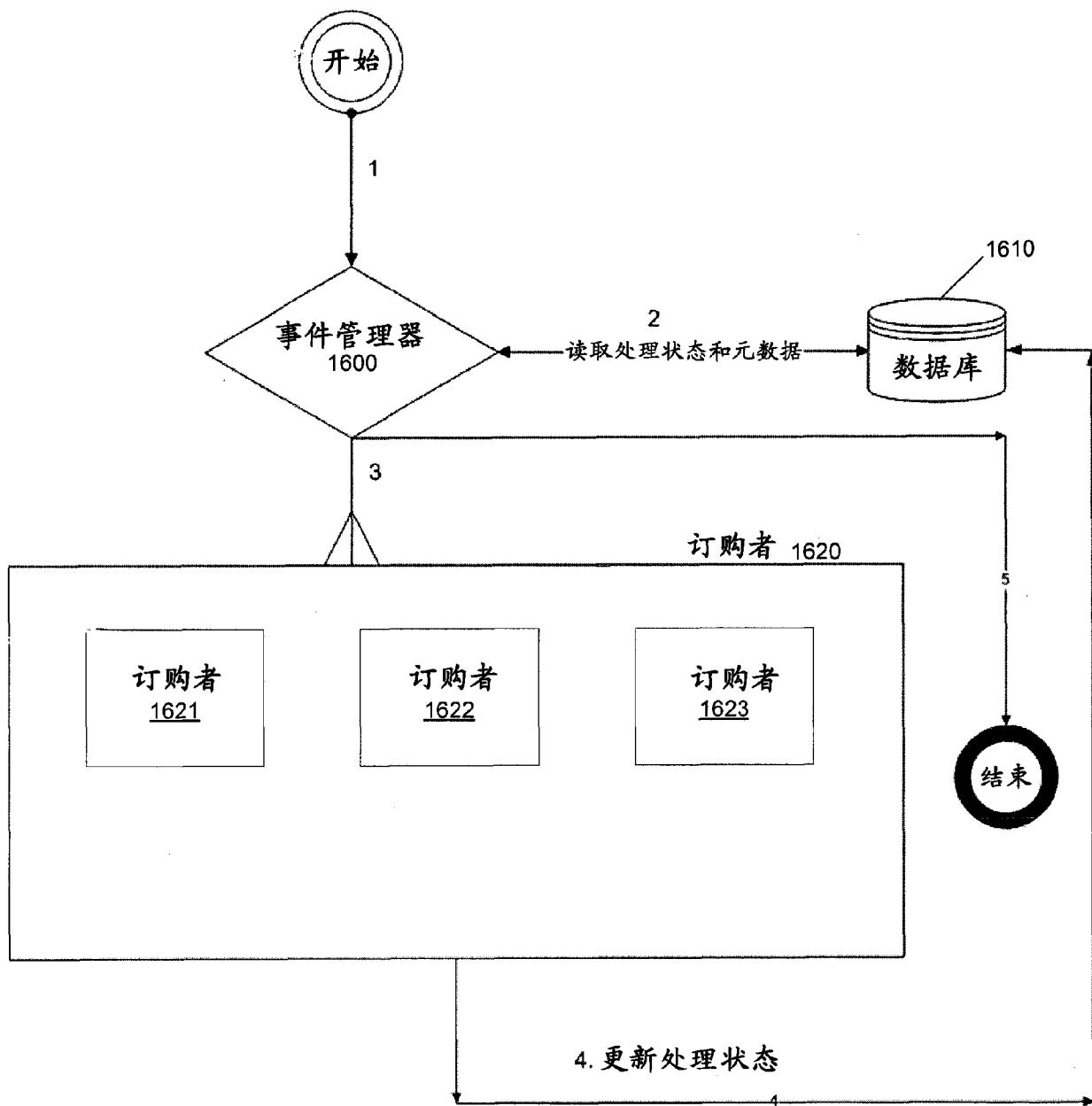


图16

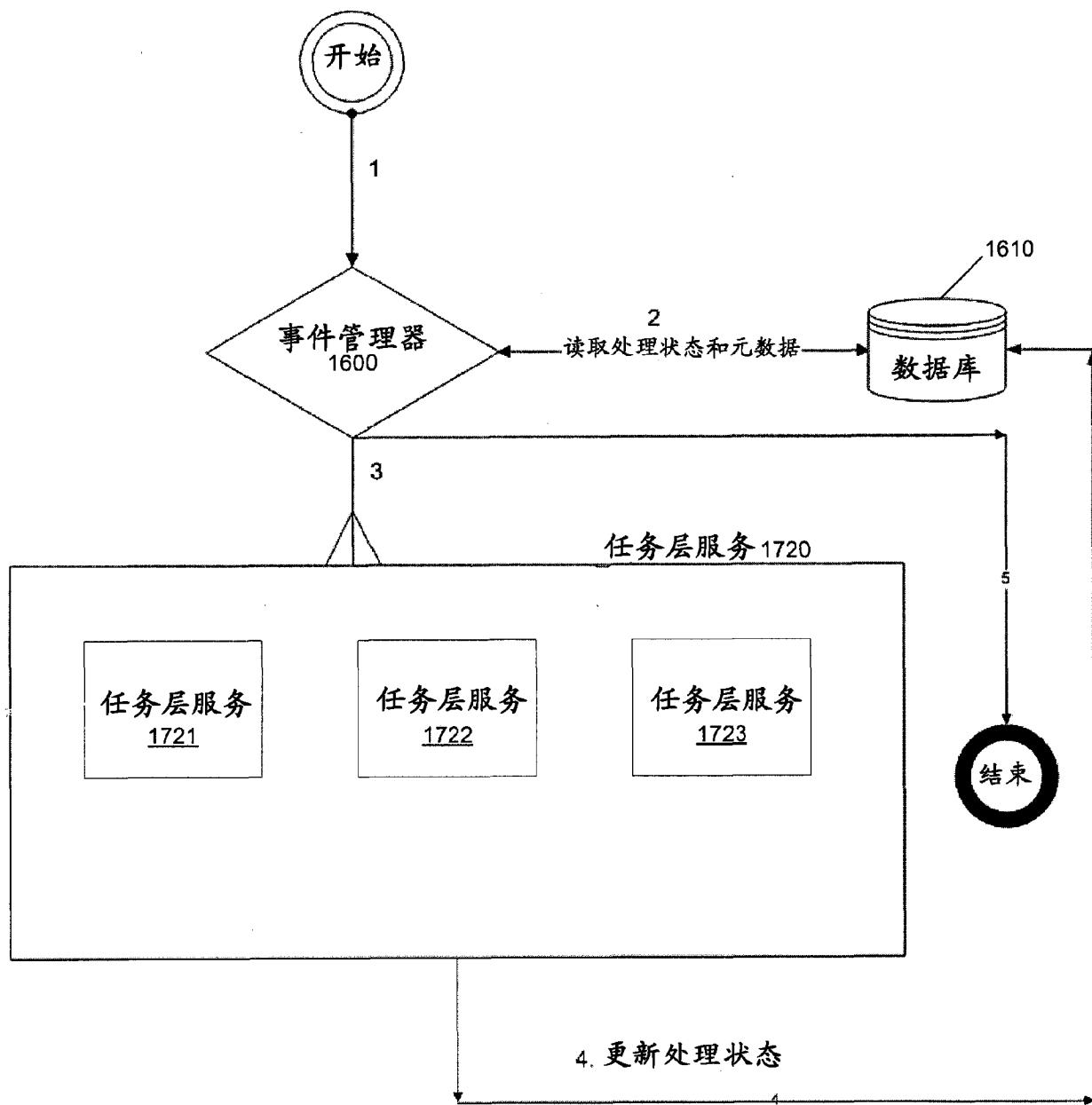


图17

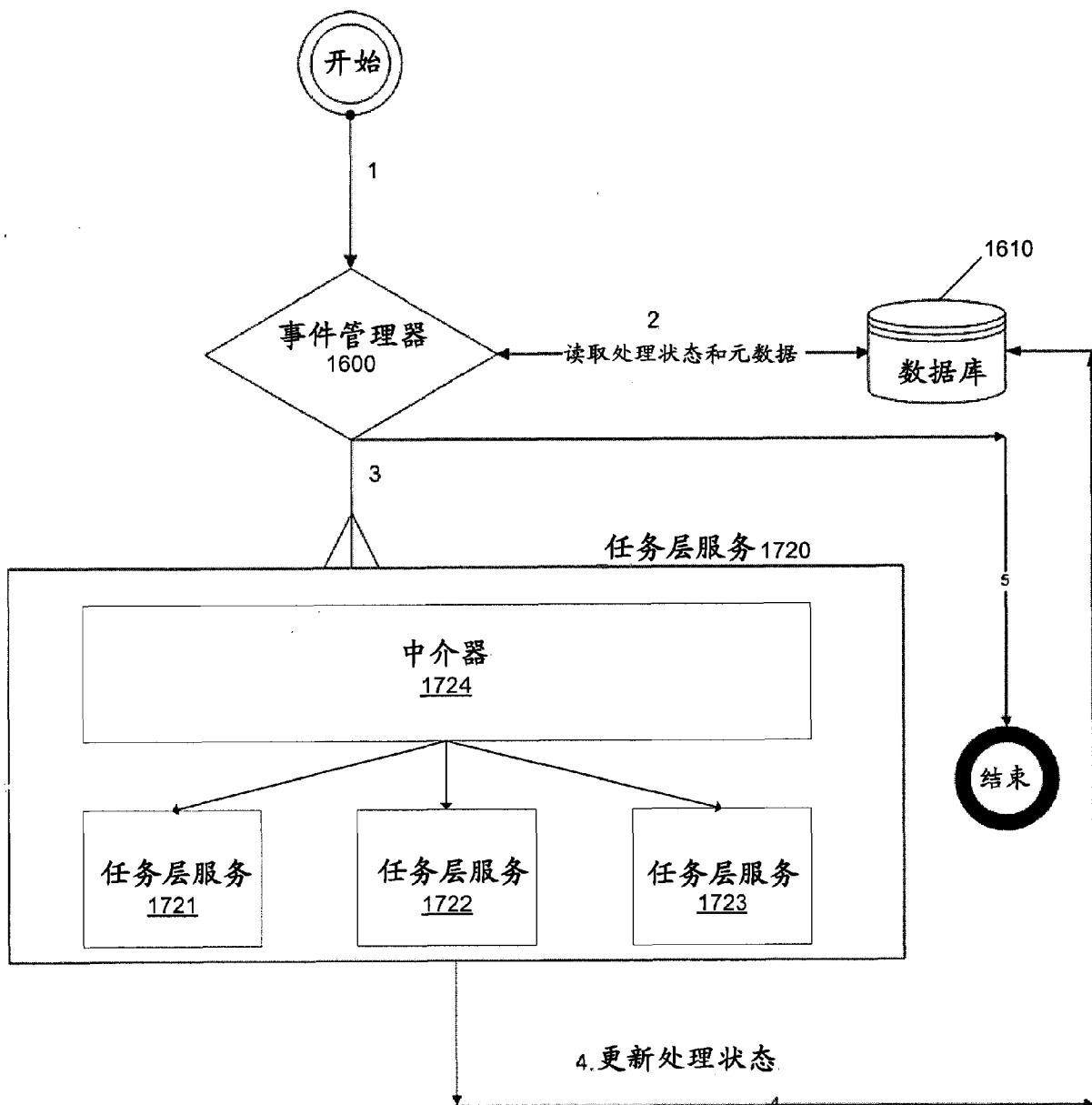


图18

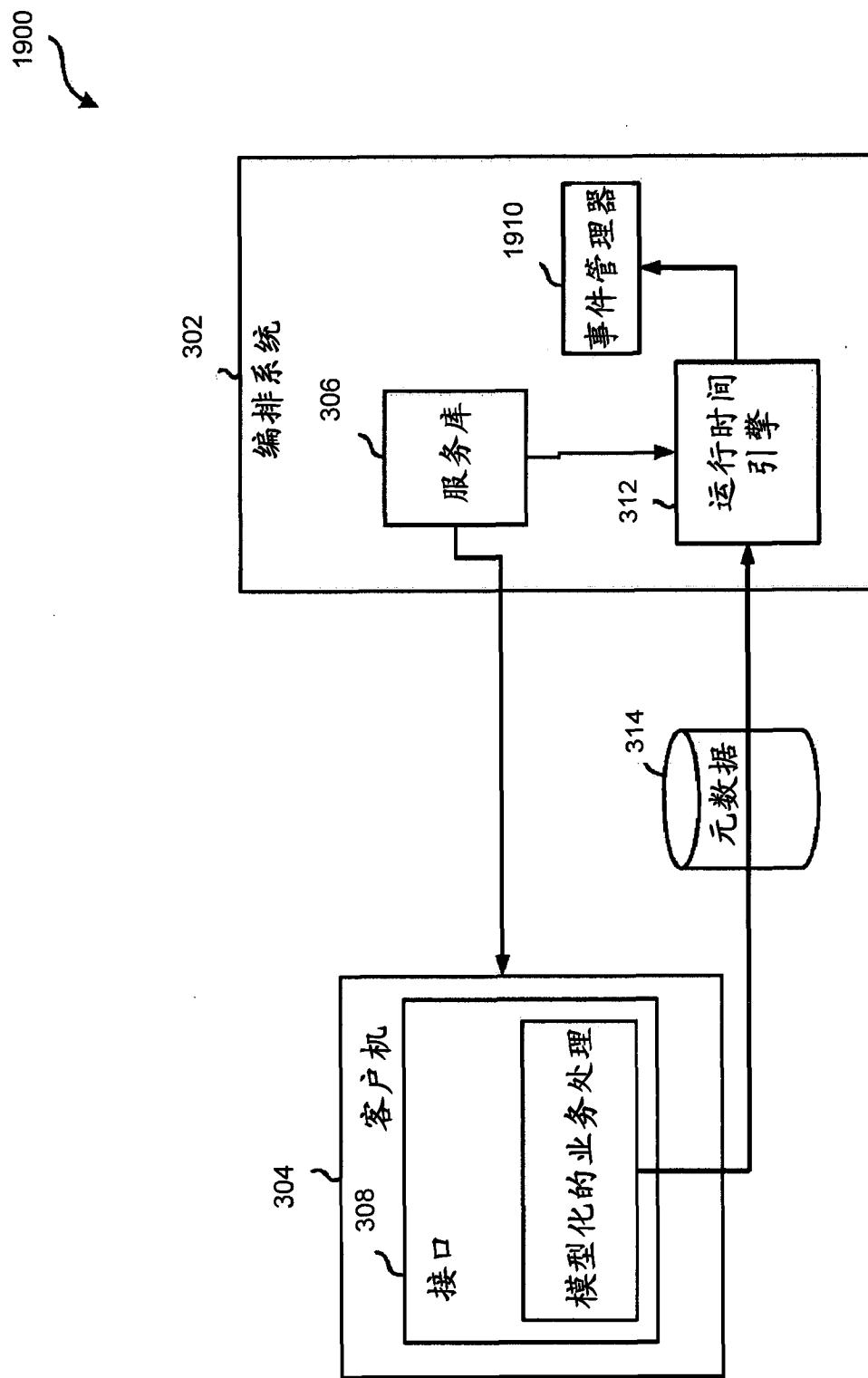


图19

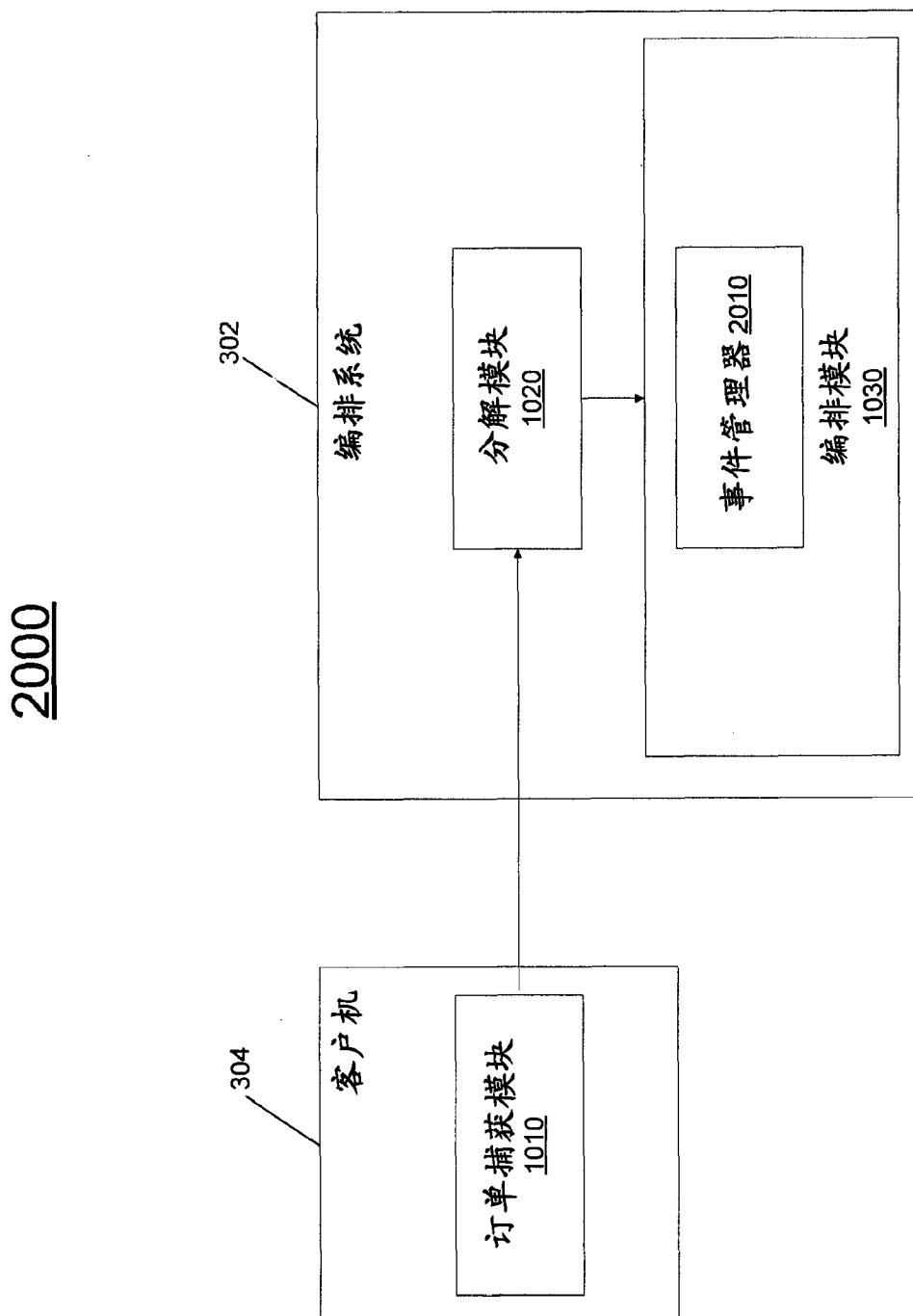


图 20

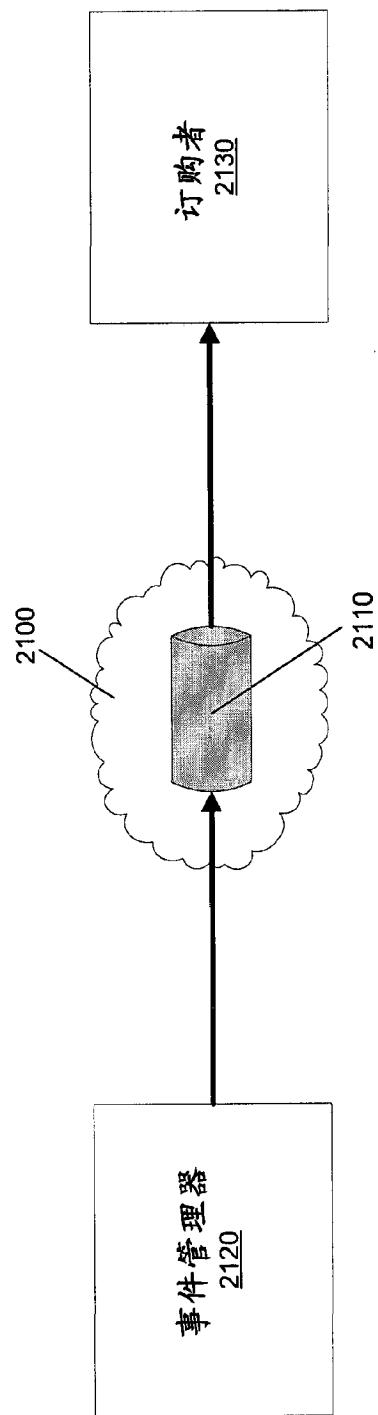


图21

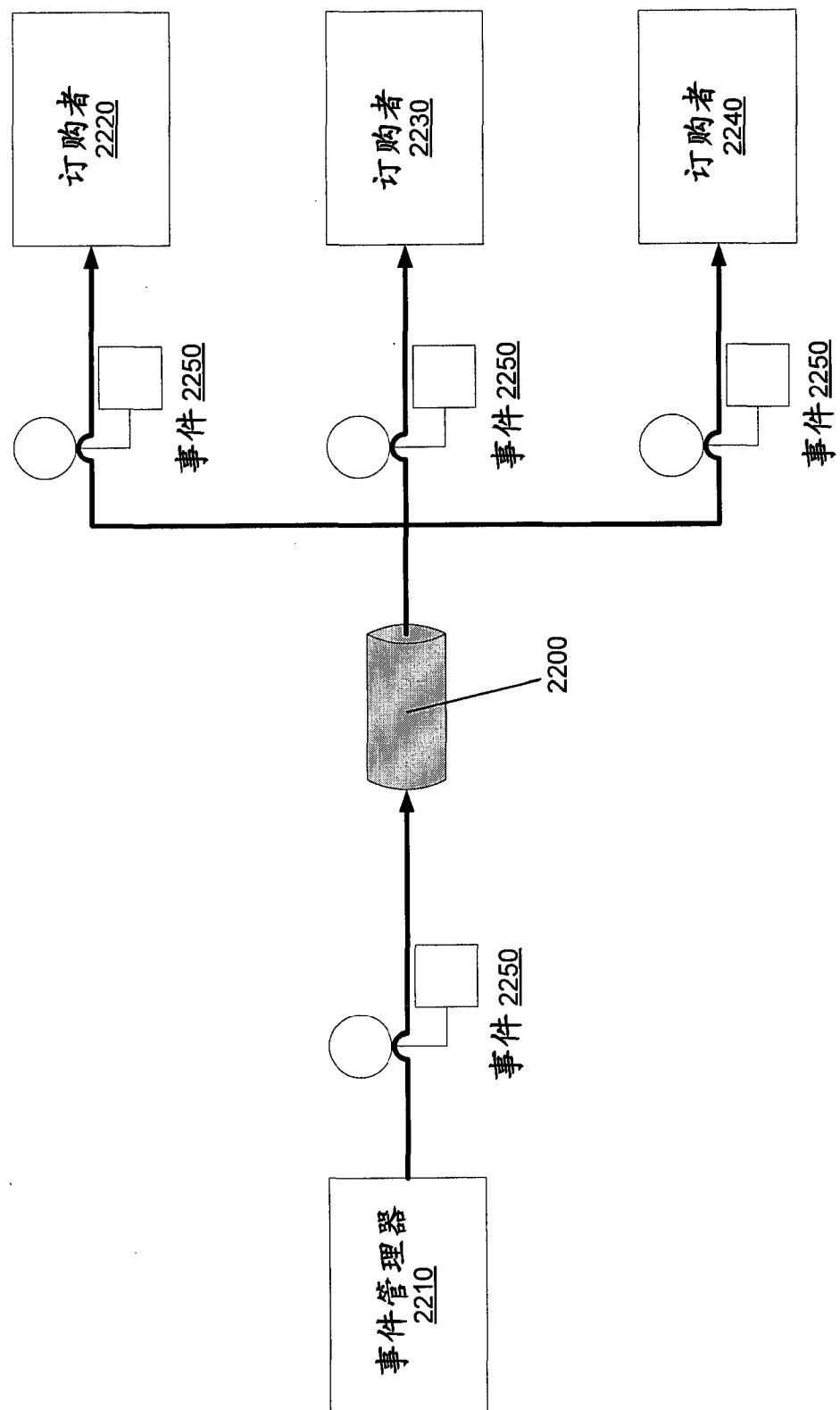


图22

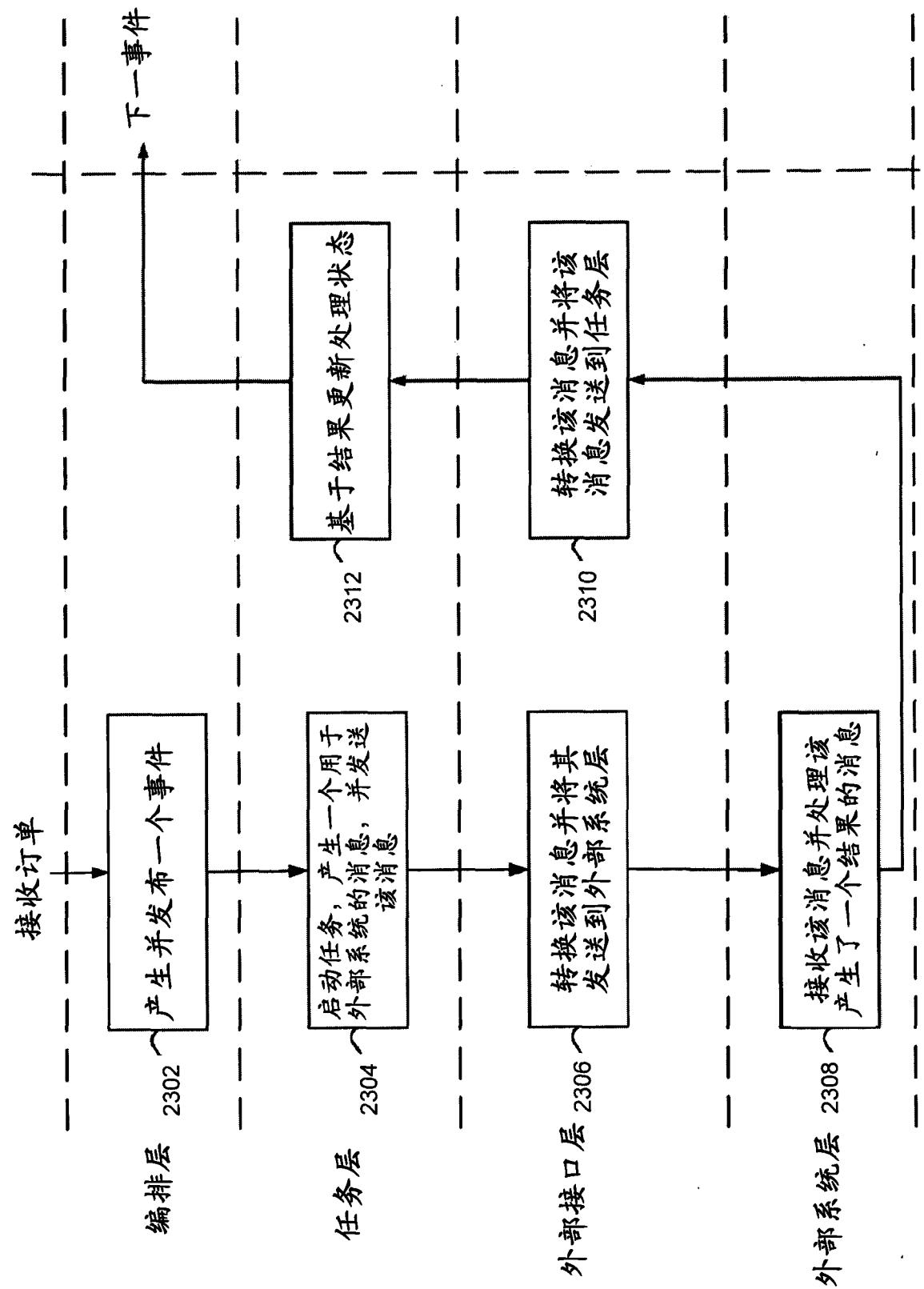


图23

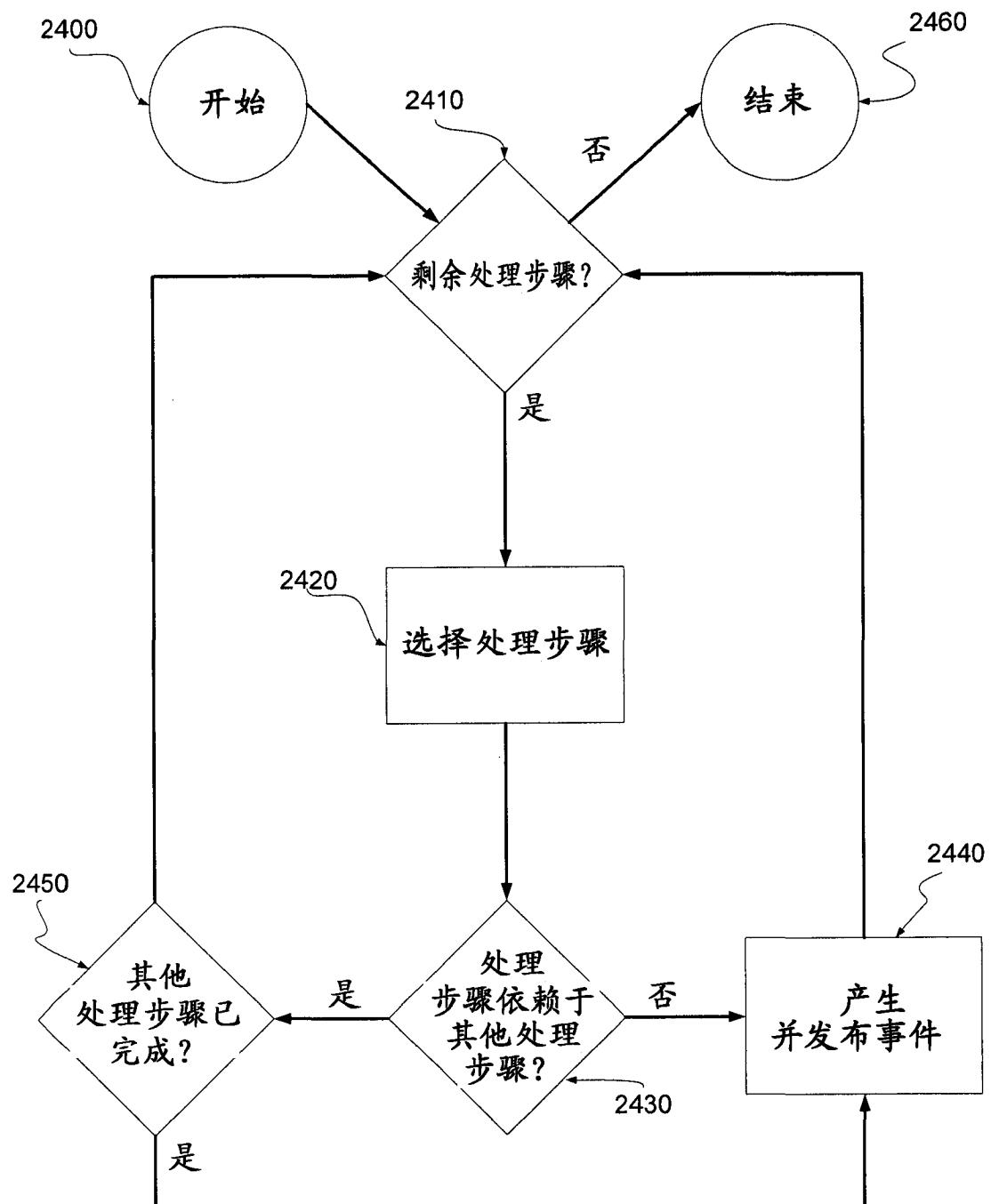


图24

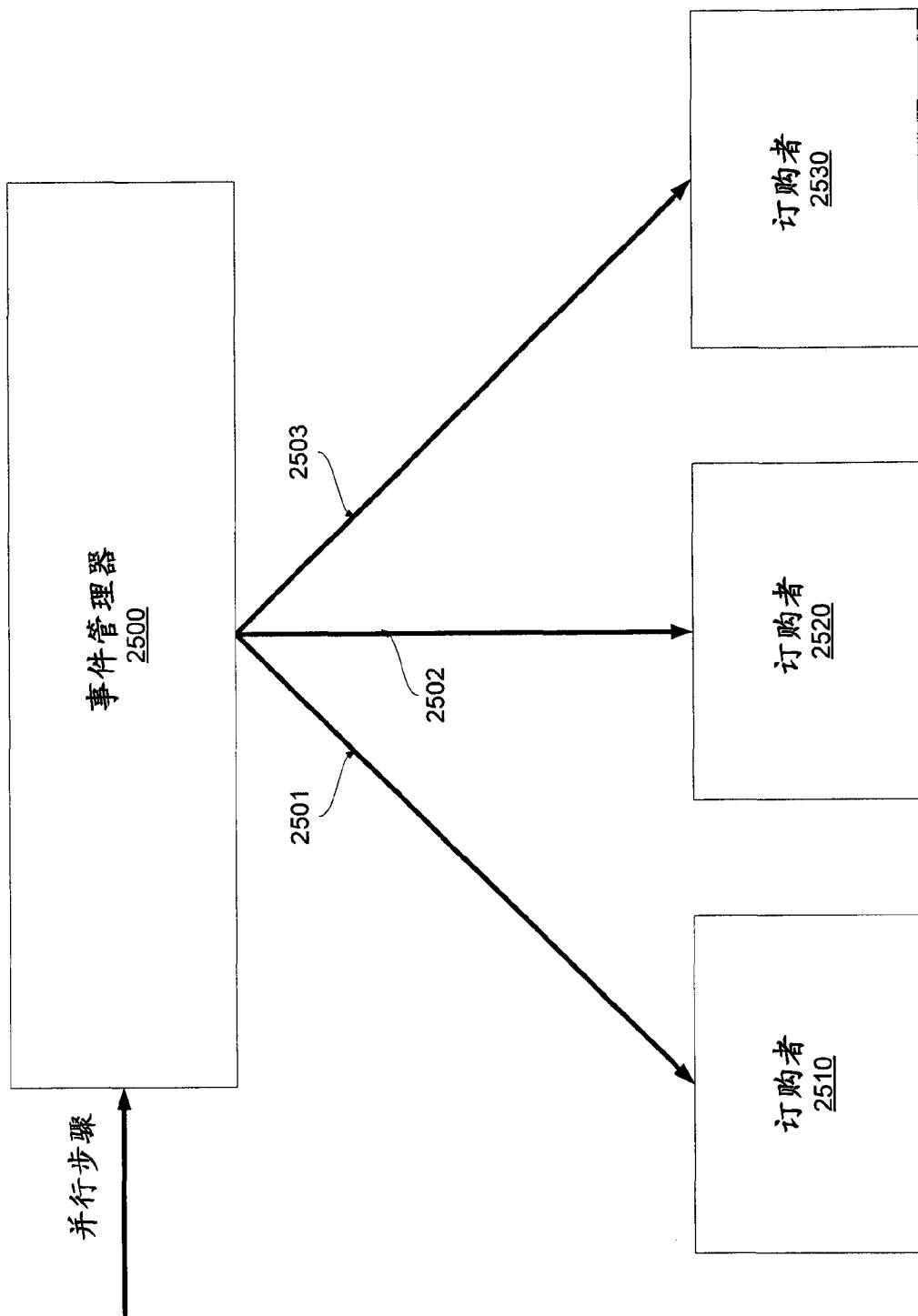


图25

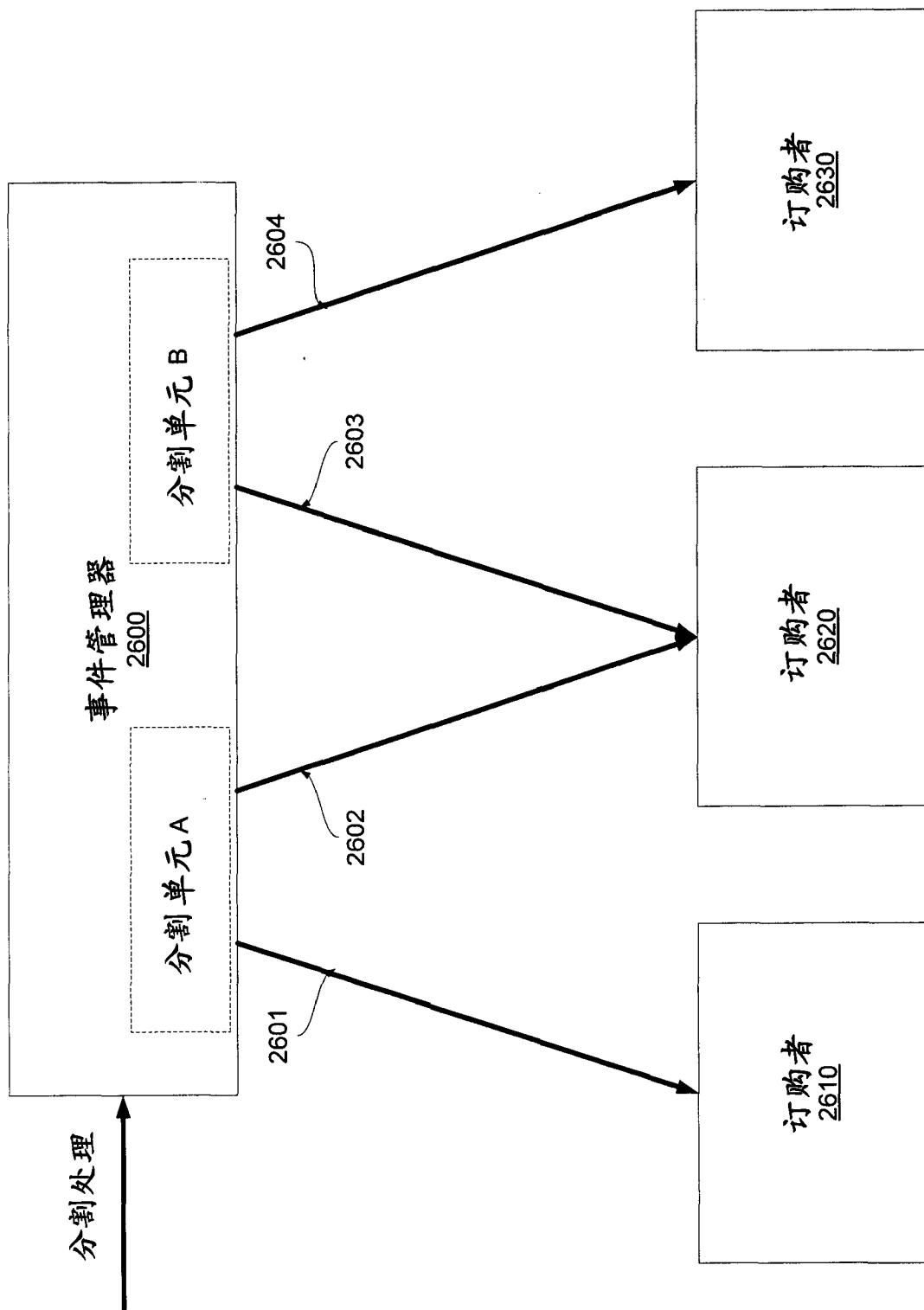


图26

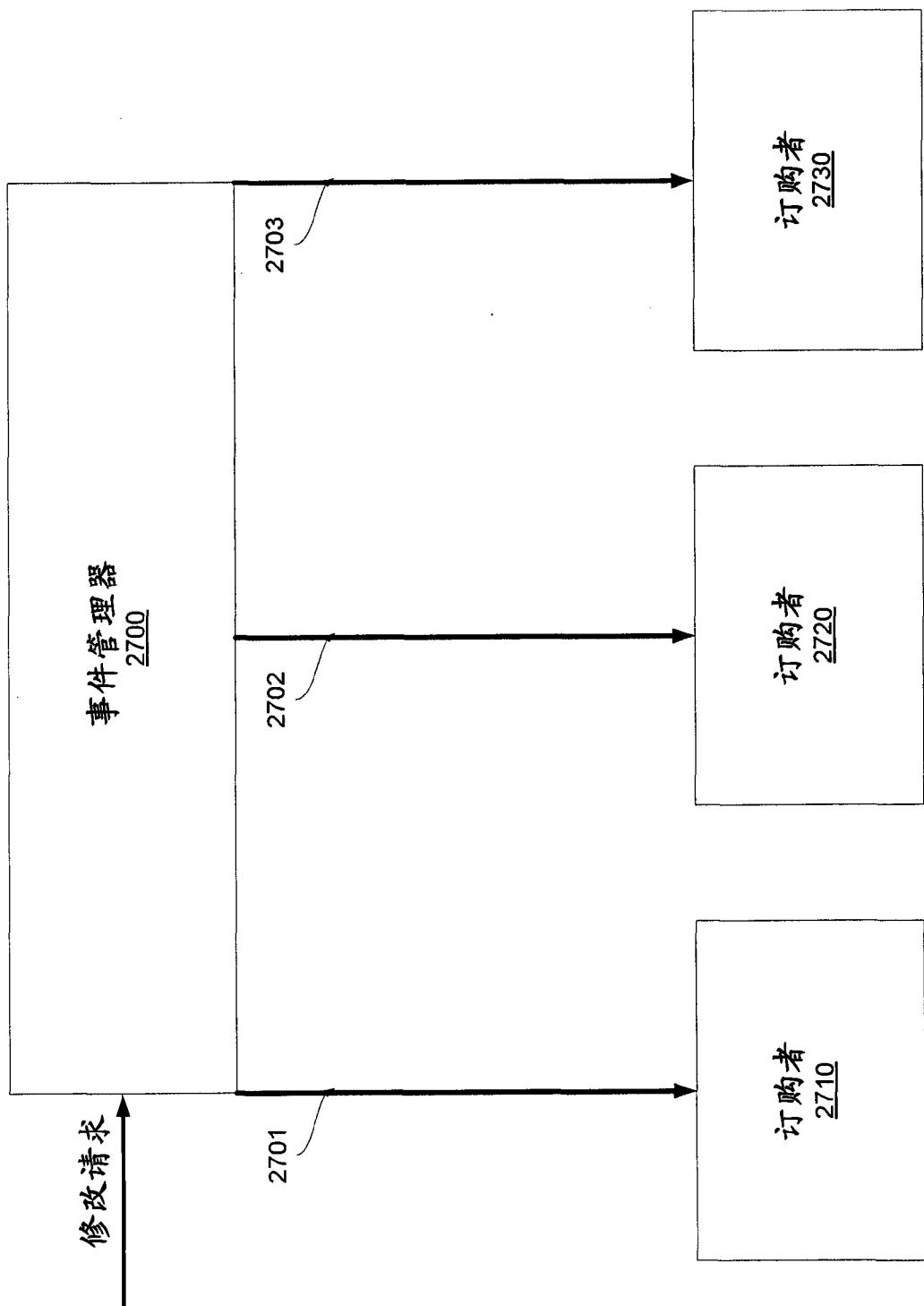


图27

