(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101821721 B
(45)授权公告日 2017.04.12

(21)申请号 200880107445.2

(22)申请日 2008.07.25

(65)同一申请的已公布的文献号
申请公布号 CN 101821721 A

(43)申请公布日 2010.09.01

(30)优先权数据
60/952,075 2007.07.26 US

(85)PCT国际申请进入国家阶段日
2010.03.17

(86)PCT国际申请的申请数据
PCT/US2008/071206 2008.07.25

(87)PCT国际申请的公布数据
WO2009/015342 EN 2009.01.29

(73)专利权人 起元技术有限责任公司
地址 美国马萨诸塞州

(72)发明人 克雷格·W·斯坦菲尔
约瑟夫·S·沃利三世

(74)专利代理机构 北京林达刘知识产权代理事务所(普通合伙) 11277
代理人 刘新宇

(51)Int.Cl.
G06F 12/00(2006.01)

(56)对比文件
US 2007022077 A1,2007.01.25,

审查员 王静

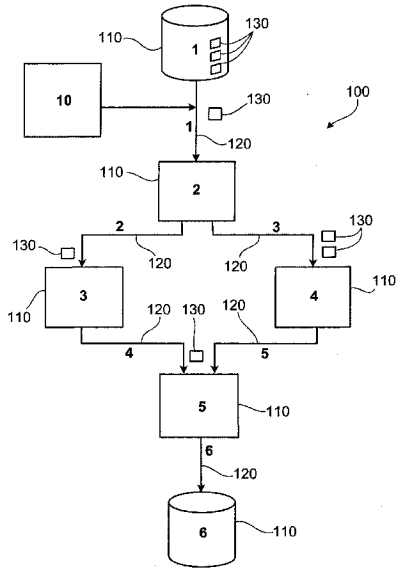
权利要求书3页 说明书11页 附图7页

(54)发明名称

具有误差处理的事务型基于图的计算

(57)摘要

使用基于图的计算处理事务包括:确定(305)一个或更多个计算图的集合中的一个计算图的多个图元的至少一个包括针对给定事务要被进行的计算,把给定事务和包括与相应图元相关联的可重用计算元素的计算图的实例相关联(310),和,执行(320)图以进行计算。



CN 101821721 B

1.一种用于使用基于图的计算处理事务的方法，所述方法包括：

确定(305)多个计算图的一个特定计算图包括至少一个图元，所述图元和针对给定事务要被进行的计算相关联；

把给定事务和该特定计算图的实例相关联(310)，其中该特定计算图包括配置为在该特定计算图的实例中实例化各自图元的各计算元素；以及

执行(320)该特定计算图的实例以进行所述计算。

2.如权利要求1所述的方法，其中，所述多个计算图中的计算图的至少某些实例共享一个或更多个计算元素。

3.如权利要求1所述的方法，其中，计算元素包括由操作系统过程和过程线程中的至少一个执行的计算。

4.如权利要求1所述的方法，其中，图元包括计算图的顶点。

5.如权利要求1所述的方法，其中，把事务和计算图的实例相关联包括在开始执行图元之前，把对应于计算图中的每一个图元的计算元素分配给计算图的实例。

6.如权利要求1所述的方法，其中，把事务和计算图的实例相关联包括在使用已分配给计算图的实例的计算元素执行一个图元之后，把对应于计算图中的另一个图元的计算元素分配给计算图的实例。

7.如权利要求1所述的方法，其中，至少两个图元使用公共资源，并且执行图以进行计算包括把使用公共资源的图元中的每一个分配给单个计算元素。

8.如权利要求7所述的方法，其中，所述单个计算元素在图元被分配给计算元素时已经被启动。

9.如权利要求7所述的方法，其中，公共资源包括数据库。

10.如权利要求7所述的方法，其中，公共资源包括特定端口。

11.如权利要求1所述的方法，其中，处理事务包括接收对事务的请求。

12.如权利要求1所述的方法，还包括：

确定该特定计算图的至少一个图元和针对第二事务要进行的计算相关联；

把第二事务和该特定计算图的第二实例相关联；以及

执行该特定计算图的第二实例以进行针对第二事务的计算。

13.如权利要求12所述的方法，其中，使用计算图的不同实例进行的针对事务的计算被以时间交织的方式进行。

14.如权利要求12所述的方法，其中，多个事务被并行处理。

15.如权利要求12所述的方法，其中，每一个事务与一个或更多个根据对应的计算图处理的工作元素相关联。

16.如权利要求15所述的方法，其中，至少某些事务各自与根据对应的计算图处理的一个工作元素相关联。

17.如权利要求1所述的方法，还包括形成至少某些计算图的多个实例。

18.如权利要求1所述的方法，还包括识别在进行针对事务中的一个的计算时已出现了错误，并继续进行针对事务中的另外一个的计算。

19.如权利要求1所述的方法，其中，在第一时间开始多个事务中的第一事务的处理，并且在晚于第一时间的第二时间开始多个事务中的第二事务的处理，所述方法还包括在完成

进行针对第一事务的计算之前完成进行针对第二事务的计算。

20.一种用于使用基于图的计算处理事务的系统，所述系统包括：

用于确定多个计算图的一个特定计算图包括至少一个图元，所述图元和针对事务要被进行的计算相关联的装置；

用于把给定事务和该特定计算图的实例相关联的装置，其中该特定计算图包括配置为在该特定计算图的实例中实例化各自图元的各计算元素；和

用于执行特定计算图的实例以进行所述计算的装置。

21.如权利要求20所述的系统，其中，所述多个计算图中的计算图的至少某些实例共享一个或更多个计算元素。

22.如权利要求20所述的系统，其中，计算元素包括由操作系统过程和过程线程中的至少一个执行的计算。

23.如权利要求20所述的系统，其中，图元包括计算图的顶点。

24.如权利要求20所述的系统，其中，用于把事务和计算图的实例相关联的装置包括用于在开始执行图元之前、把对应于计算图中的每一个图元的计算元素分配给计算图的实例的装置。

25.如权利要求20所述的系统，其中，用于把事务和计算图的实例相关联的装置包括用于在使用已分配给计算图的实例的计算元素执行一个图元之后、把对应于计算图中的另一个图元的计算元素分配给计算图的实例的装置。

26.如权利要求20所述的系统，其中，至少两个图元使用公共资源，并且执行图以进行计算包括把使用公共资源的图元中的每一个分配给单个计算元素。

27.如权利要求26所述的系统，其中，所述单个计算元素在图元被分配给计算元素时已经被启动。

28.如权利要求26所述的系统，其中，公共资源包括数据库。

29.如权利要求26所述的系统，其中，公共资源包括特定端口。

30.如权利要求20所述的系统，包括用于接收对事务的请求的装置。

31.如权利要求20所述的系统，还包括：

用于确定该特定计算图的至少一个图元和针对第二事务要进行的计算相关联的装置；

用于把第二事务和该特定计算图的第二实例相关联的装置；以及

用于执行该特定计算图的第二实例以进行针对第二事务的计算的装置。

32.如权利要求31所述的系统，其中，使用计算图的不同实例进行的针对事务的计算被以时间交织的方式进行。

33.如权利要求31所述的系统，其中，多个事务被并行处理。

34.如权利要求31所述的系统，其中，每一个事务与一个或更多个根据对应的计算图处理的工作元素相关联。

35.如权利要求34所述的系统，其中，至少某些事务各自与根据对应的计算图处理的一个工作元素相关联。

36.如权利要求20所述的系统，包括用于形成至少某些计算图的多个实例的装置。

37.如权利要求20所述的系统，包括用于识别在进行针对事务中的一个的计算时已出现了错误，并继续进行针对事务中的另外一个的计算的装置。

38.如权利要求20所述的系统，其中，在第一时间开始多个事务中的第一事务的处理，并且在晚于第一时间的第二时间开始多个事务中的第二事务的处理，所述系统包括用于在完成进行针对第一事务的计算之前完成进行针对第二事务的计算的装置。

具有误差处理的事务型基于图的计算

[0001] 相关申请的交叉引用

[0002] 本申请要求2007年7月26日提交的序列号为60/952,075的美国申请的优先权，该申请通过引用被合并于此。

技术领域

[0003] 本发明涉及基于图(graph-based)的计算的执行。

背景技术

[0004] 复杂的计算经常可以被表达为通过有向图的数据流，计算的组件(component)和图的顶点相关联，并且组件之间的数据流对应于图的连线(link)(弧、边)。在美国专利5,966,072“EXECUTING COMPUTATIONSEXPRESSED AS GRAPHS”中描述了实施这种基于图的计算的系统。执行基于图的计算的一种方法是执行许多个过程(process)，每一个均和图的不同顶点相关联，并根据图的连线在过程之间建立通信路径。例如，通信路径可以使用TCP/IP或者UNIX域套接字，或者使用共享存储器在过程之间传递数据。

发明内容

[0005] 在一个方面中，一般地，一种用于使用基于图的计算处理事务的方法包括：确定一个或更多个计算图的集合中的一个计算图的多个图元中的至少一个包括针对给定事务要被进行的计算，把给定事务和包括与相应图元相关联的可重用计算元素的计算图的实例相关联，和，执行图以进行计算。

[0006] 方面可以包括下列特征中的一个或更多个。

[0007] 计算图集合中的图的至少某些实例共享一个或更多个计算元素。

[0008] 计算元素包括由操作系统过程(operating system process)和过程线程(process thread)中的至少一个执行的计算。

[0009] 图元包括计算图的顶点。

[0010] 把事务和计算图的实例相关联包括在开始执行图元之前，把对应于计算图中的每一个图元的计算元素分配给计算图的实例。

[0011] 把事务和计算图的实例相关联包括在使用已分配给计算图的实例的计算元素执行另一个图元之后，把对应于计算图中的图元的计算元素分配给计算图的实例。

[0012] 至少两个图元使用公共资源，并且执行图以进行计算包括把使用公共资源的图元中的每一个分配给单个计算元素。

[0013] 单个计算元素在图元被分配给计算元素时已经被启用。

[0014] 公共资源包括数据库。

[0015] 公共资源包括特定端口。

[0016] 处理事务包括接收对事务的请求。

[0017] 所述方法也包括：确定同一计算图和针对第二事务要进行的计算相关联，把第二

事务和计算图的第二实例相关联，和，执行图的第二实例以进行针对第二事务的计算。

[0018] 使用计算图的不同实例进行的针对事务的计算被以时间交织的方式进行。

[0019] 多个事务被并行处理。

[0020] 每一个事务和一个或更多个根据对应的计算图处理的工作元素相关联。

[0021] 至少某些事务各自和被根据对应的计算图处理的一个工作元素相关联。

[0022] 所述的方法还包括形成至少某些计算图的多个实例。

[0023] 所述的方法还包括识别在进行针对事务其中之一的计算时已出现了错误，并继续进行针对事务中的另外一个的计算。

[0024] 在第一时间开始多个事务中的第一事务的处理，并且在晚于第一时间的第二时间开始多个事务中的第二事务的处理，所述方法还包括在完成进行针对第一事务的计算之前完成进行针对第二事务的计算。

[0025] 在另一方面中，一般地，一种用于使用基于图的计算处理事务的系统包括：用于确定一个或更多个计算图的集合中的一个计算图的多个图元的至少一个包括针对事务要被进行的计算的装置，用于把给定事务和包括与相应图元相关联的可重用计算元素的计算图的实例相关联的装置，和，用于执行图以进行计算的装置。

[0026] 在另一个方面中，一般地，一种计算机可读介质存储用于使用基于图的计算处理事务的计算机程序。所述计算机程序包括指令，用于导致计算机系统：确定一个或更多个计算图的集合中的一个计算图的多个图元的至少一个包括针对给定事务要被进行的计算，把给定事务和包括与相应图元相关联的可重用计算元素的计算图的实例相关联，和，执行图以进行计算。

[0027] 在又一个方面中，一般地，一种用于处理基于图的计算的方法，包括：在包括代表根据联结顶点的连线处理工作元素的图组件的顶点的图内，提供至少一个被配置成向图外部的过程提供错误信息的错误处理图组件，和，处理数据，包括响应于图组件在处理时遇到错误，把处理重定向到错误处理图组件，包括根据至少一个到代表错误处理组件的顶点的连线，把至少某些工作元素定向到错误处理组件。

[0028] 方面包括下列特征中的一个或更多个。

[0029] 把处理重定向到错误处理图组件包括从至少一个输入队列去除工作元素。

[0030] 把处理重定向到错误处理图组件包括处理被定向到错误处理图组件的工作元素。

[0031] 处理被定向到错误处理图组件的工作元素包括回滚在错误之前做出的对数据库的变化。

[0032] 处理数据包括，对于在处理错误中未包括的图组件，丢弃被定向到那些图组件的工作元素。

[0033] 提供了子图，所述子图包括被配置成提供错误代码作为子图的输出的错误处理子图组件。

[0034] 如果由子图提供的输出指示子图中出现的错误，则把处理重定向到错误处理图组件。

[0035] 把处理重定向到错误处理图组件包括把当发生错误时图组件正在处理的工作元素从遇到错误的图组件传送到错误处理图组件。

[0036] 工作元素被根据到代表错误处理组件的顶点的连线传送。

[0037] 把处理重定向到错误处理图组件包括把关于错误的报告信息从遇到错误的图组件传送到错误处理图组件。

[0038] 报告信息被根据遇到错误的图组件和错误处理组件之间的隐式连接传送。

[0039] 响应于用户请求，把隐式连接显露为代表遇到错误的图组件的顶点和代表错误处理组件的顶点之间的显式连线。

[0040] 提供错误处理图组件包括提供多个错误处理图组件，并且把处理重定向到错误处理图组件包括基于从遇到错误的图组件提供的输出选择错误处理图组件。

[0041] 处理数据也包括，如果图组件在处理时遇到错误，则输出导致该错误的工作元素的标识。

[0042] 处理包括：使能图的第一组件；禁止错误处理组件；和，对于第一组件下游的除了错误处理组件以外的每一个组件，如果该组件上游的最接近该组件的组件被使能，则使能该组件。

[0043] 把处理重定向到错误处理图组件包括：停止执行每一个被使能的图组件；禁止遇到错误的组件；使能错误处理组件；禁止在遇到错误的组件下游但不在错误处理组件下游的组件；和，使能错误处理组件上游的组件。

[0044] 把处理重定向到错误处理图组件包括：在第一组件中出现错误的情况下，如果在第一条件下出现错误，则把过程流从第一组件定向到第一组件上游的第一错误处理组件，并且，如果在第二条件下出现错误，则把过程流从第一组件定向到第一组件下游的第二错误处理组件。

[0045] 第一条件是计数器在界限以下。

[0046] 第二条件是计数器在界限以上。

[0047] 把处理重定向到错误处理图组件还包括使能图组件的集合，该集合在错误之前已被确定。

[0048] 在另一个方面中，一般地，一种用于处理基于图的计算的系统包括，在包括代表根据联结顶点的连线处理工作元素的图组件的顶点的图内，用于提供至少一个被配置成向图外部的过程提供错误信息的错误处理图组件的装置，和，用于处理数据的装置，处理数据包括响应于图组件在处理时遇到错误，把处理重定向到错误处理图组件，包括根据至少一个到代表错误处理组件的顶点的连线，把至少某些工作元素定向到错误处理组件。

[0049] 在又一个方面中，一般地，一种计算机可读介质存储用于处理基于图的计算的计算机程序，所述计算机程序包括指令，用于导致计算机系统：在包括代表根据联结顶点的连线处理工作元素的图组件的顶点的图内，提供至少一个被配置成向图外部的过程提供错误信息的错误处理图组件，和，处理数据，包括响应于图组件在处理时遇到错误，把处理重定向到错误处理图组件，包括根据至少一个到代表错误处理组件的顶点的连线，把至少某些工作元素定向到错误处理组件。

[0050] 从下面的描述，以及从权利要求，本发明的其他特征和优点将会清晰。

附图说明

[0051] 图1是示出基于图的计算的实例的图。

[0052] 图2是用于处理工作流的系统的逻辑框图。

[0053] 图3A是用于处理每一个工作流的流程图。

[0054] 图3B是用于处理错误的流程图。

[0055] 图4A、图4B、图5和图6是错误处理图的例子。

具体实施方式

[0056] 1.概述

[0057] 本申请涉及2002年10月10日提交的美国专利申请10/268,509“Startupand Control of Graph-Based Computation”,以及2007年4月10日提交的作为申请10/268,509的继续申请的11/733,579“Transactional Graph-BasedComputation”。两者均通过引用被合并于此。

[0058] 下面描述的系统实施了用于执行依据计算图定义的计算的方法。参考图1,计算图100的例子包括许多个由单向连线120联结(join)的顶点110。在图1中所示的例子中,顶点110被从1到6进行编号,并且连线120也被从1到6进行编号。计算图100处理由一系列工作元素130组成的工作流,工作元素例如根据和事务(transaction)处理系统相关联的计算图来处理的各个事务。一个事务可以由多个工作元素构成。每一个顶点均和由整个计算图定义的计算的一部分相关联。在这个例子中,顶点1提供对用于和一个或更多个事务相关联的初始的一系列工作元素130的存储器的访问,并在其输出连线1上传递该系列。实施和每一个顶点相关联的计算的过程依次处理工作元素130,并且通常在该顶点的一个或更多个输出连线上产生工作元素。

[0059] 当至少一个工作元素在顶点的每一个输入处排队时,用于顶点的过程准备好运行。如图1中所示,在连线1上,工作元素130在运输中,在顶点3,一个工作元素排队准备好进行处理,并且在顶点4,两个工作元素排队等候处理。因此,用于顶点3和4的过程准备好运行以处理排队的工作元素。如图所示,顶点5在其输入之一——连线4——上具有排队的工作元素,但是在另一输入——连线5——上没有排队的工作元素。因此,和顶点5相关联的过程未准备好运行。

[0060] 在某些例子中,工作流可以包括来自多个事务的工作元素(即一个和更多个工作元素的第一集合对应于第一事务,一个和更多个元素的第二集合对应于第二事务,等等)。事务可以包括代表全部要作为集合处理的动作的工作元素集合,所以如果一个动作失败,则不会执行任何一个动作。图的多个实例可被用来处理多个事务,并且,通过利用可重用计算元素(例如,操作系统过程)实施图组件的计算,可以按需要生成各个图组件的多个实例(由计算图的顶点代表)。通过把不同的事务和图的不同相应实例相关联,多个事务可以被并行(concurrently)处理。如下面更详细地描述的那样,由于使多个计算元素能够按需要分配给图实例,因此有效的资源共享能够通过使计算元素由一个图实例使用并被另一个图实例重新使用来实现。

[0061] 参考图2,用于执行计算图以处理包含事务的工作流的系统200包括存储的图数据结构210。这些数据结构包括计算图的说明,其包括图的顶点和连线的特性。无需加载整个图就可以访问这些数据结构的部分,例如,可以加载各个图组件的说明以便把工作元素分配给该图组件的新生成的实例。

[0062] 系统的事务预订(subscription)模块220从事务预订图组件(例如提供命令而不

一定处理工作元素的组件，例如由图1的顶点10代表的组件）接收控制输入222，该控制输入包括使用在所存储的图数据结构210中规定的相应的计算图处理特定工作流232的命令。事务预订模块220追踪（keep track of）可用于实例化要被分配给特定事务的图实例的图计算处理资源230。事务预订模块220包括使用计算图的说明来确定如何使用图计算处理资源230实例化图实例的调度器，图计算处理资源230通常由多个过程（或过程池）组成，其中每一个过程起到实例化图实例中的给定图组件的可重用计算元素的作用。被执行以进行图组件的计算的过程可以利用外部数据和过程240，外部数据和过程240包括数据库引擎、数据存储器，或者在和计算图的顶点相关联的处理期间访问的其他模块。在某些例子中，能够进行多个不同操作的单个过程或者过程集合被绑定到图的给定实例以处理该实例的所有操作。

[0063] 在某些例子中，事务预订模块220的调度器使用远程过程调用（remoteprocedure call，RPC）过程。当调度器接收到用于给定事务的工作元素时，它把该工作元素分配给和事务相关联的（即被分配到的）图实例的适当组件。分配给该图实例的过程执行该组件的计算。和该工作元素相关联的数据被写入可用于该图实例并可被该过程访问的临时空间。该调度器被通知事务预订模块220已完成该组件，然后，调度器调度任何下游图组件用于执行。最终，该事务将穿越整个图（因为该图是使用图计算处理资源230执行的），并通过RPC公布（publish）过程输出。这取走在临时空间中累积的数据并将其交给适当的输出通道，例如图1中的数据库输出6。RPC公布过程可以与PRC预订过程复用，因此它可以访问最初在其上接收到事务的套接字。

[0064] 一般来讲，不同的事务可以被并行处理，每一个均由图的不同实例处理。系统200通过事务预订模块220为用于每一个事务的计算图的实例分配资源，并通过图计算处理资源230控制它们的执行以处理工作流。

[0065] 2.图数据结构

[0066] 系统200包括许多特征，它们提供了图计算的快速启动以及有限资源的有效共享。

[0067] 在利用计算图的实例处理事务之前，事务预订模块220在功能共享存储器（functionally shared memory）中为该图实例生成运行时数据结构。在一个实施例中，生成单个共享存储器段，在其中生成用于图实例的所有运行时数据结构。

[0068] 绑定到事务的一个过程或更多个过程和图的顶点相关联，并且每一个过程均把共享的存储器段映射到其地址空间。过程可以在为各个事务生成图实例时与顶点相关联，或者，它们可以在生成或者执行各个图组件的实例之前不与顶点相关联。在处理事务期间，过程从图实例的运行时数据结构读取工作元素，并将工作元素写入图实例的运行时数据结构。即，通过该图流动的用于事务的数据通过共享的存储器段中的这些运行时数据结构，被从组件传递到组件，并且如果多于一个的过程被绑定到该事务，则被从过程传递到过程。通过在图的每一个组件可访问的存储器空间中包含用于给定事务的数据，并利用始终如一（consistent）的过程或者过程集合执行每一个组件，可以在组件之间共享状态。除了其他优点外，这允许在确认事务被成功执行以后，和执行事务的计算相关联的所有数据库操作被立刻提交（commit）。

[0069] 3.过程池

[0070] 如上面所介绍的那样，可以使用由调度器管理和分配的过程池实施用于执行图实

例的组件的图计算处理资源230。多于许多不同类型的计算中的每一个，在使用要求该类型的计算的图组件开始处理事务的工作流之前生成过程池。当事务被分配给图实例时，如果需要特定类型的计算来进行该图实例的给定组件的计算，则调度器分配该过程池的一个成员供该图实例使用并且将该过程池成员分配给该给定组件。在处理事务的持续时间中，该过程池的成员保持和该图实例相关联，并且可被重新用于该图实例内要求同一类型的计算的其他组件。一旦在用于该事务的图实例中需要该类型的计算的最后一个组件的上游没有工作元素了，则该过程可以被释放回过程池。可能有很多不同的过程池，每一个均和对应的计算类型相关联。在池中的过程可被用于相同或者不同图实例中的组件，例如包括用于不同图实例中给定类型的组件，以及用于一个图实例中多个不同组件。

[0071] 在某些实施方案中，过程池中的每一个过程是被管理过程池的事务预订模块220调用的单独过程(例如UNIX过程)。对于每一个过程池，模块220维持单独的工作队列。工作队列中的每一个条目(entry)标识了图实例的一个特定顶点，针对该特定顶点该过程要进行计算。

[0072] 某些过程保留或者消耗固定的资源。这种过程的例子是做出到诸如Oracle®数据库之类的数据库的连接的过程。由于资源被消耗用于形成和维持每一个数据库连接，所以期望限制这种处于活动的过程的数量。如果图包括访问数据库的多个组件，则可能期望用于给定事务的所有数据库操作在单个数据库过程中发生。为了适应这一点，可以建立过程集合，使得每一个过程维持到数据库的连接，并且每一个过程均能够执行给定图实例可能要求的数据库功能。当图实例被分配给给定事务时，如上所述，来自所述集合的一个过程被分配给该图实例用于整个事务，并且所有的数据库组件被复用到该过程。当顶点要求访问数据库的过程来处理事务的工作元素时，所分配的过程(其已经建立其与数据库的连接)被和该顶点相关联。以这种方式，避免了连接到该数据库本来需要的该过程的初始化步骤的开销，并且针对给定事务的所有数据库动作都由同一过程处理。其他类型的过程可以以相同的方式处理。

[0073] 系统200支持不同的方法来配置用于顶点的过程，其区别在于顶点何时和过程相关联，以及何时启动针对顶点的计算。在一个类型的配置中，在其所有输入工作元素的所有数据完全可用之前，过程不和顶点相关联。如果工作元素较大，则对于整个工作元素由上游顶点计算并变为可用可能要花费一些时间。这种类型的配置避免了阻断(block)该过程以等待输入变得可用，从而其可以被该实例中的其他顶点使用。

[0074] 另一种类型的配置使用流模式。过程在至少每一个输入的开始可用时被和顶点相关联并启动。在过程执行的同时，其输入中的每一个的剩余部分变得可用。如果该输入足够快地变得可用，则该过程不阻断对输入的等待。但是，如果该输入未变为可用，则过程可能阻断。

[0075] 4.计算控制

[0076] 图3A是用于使用相应图实例处理每一个事务的过程300的流程图。当事务预订模块220(图2)接收到关于处理事务的请求时，它首先确定哪个计算图(以及对应的类型)适合处理该事务(步骤305)。例如，调度器确定某个计算图适合(例如包括适当的组件)进行该事务的计算。这可以由事务自身规定，或者，事务预订模块220可以包括或者可访问把特定事务类型和特定计算图相关联的数据。然后，事务预订模块220生成处理该事务所需类型的计

算图的图实例(如果必要的话)(步骤310),并把该事务与其相关联。作为这个过程的一部分,事务预订模块220分配共享存储器段的一部分用于图实例的运行时数据结构,并把该类型的计算图的图模板复制到运行时数据结构中,从而初始化该运行时数据结构。在通过引用合并于此的美国专利No.7,167,850中更详细地描述了使用图模板的例子。在某些例子中,图实例已被生成,并且在这个阶段,一个图实例只是被分配给当前的事务。然后,如下面更详细地描述的那样,事务预订模块220在调度器的控制下执行该图实例(步骤320)。图实例包括和可重用的相应组件相关联的(被分配的)计算元素(例如,过程)。当事务的整个工作流已被处理时,事务预订模块220提交执行该图的结果(例如,提交输出数据库的变化),并可选地释放所分配的资源和计算元素以及删除该图实例的运行时数据结构,从而允许共享存储器段的该部分被重新用于其他图实例(步骤330)。

[0077] 5.替换方案

[0078] 如上面指出的那样,在预期到存在将需要计算图的事务时,有可能预先生成该计算图的已被实例化的实例的图池。当事务被接收并需要图实例时,如果可从图池获得一个图实例,则从图池分配而不必生成该图实例。以这种方式,事务的启动成本被进一步降低。当完成用于事务的计算时,通过把变量恢复到被分配给该事务之前的其初始值并释放任何动态分配的存储器,来将该图复位。在图实例被复位后,其被返回图池。

[0079] 在某些例子中,图池中图实例的数量可被允许按需要增长,例如,可能存在每一个图的最小数量的实例,并且可以按需要生成更多实例。

[0080] 在上面的描述中,过程可以按需方式被分配给图中的顶点,其中,在到该顶点的所有输入都可用之前,过程不和顶点相关联,尽管它们被绑定到特定的图实例和事务。另一种方法是在事务和图实例相关联时把过程和顶点相关联,并维持所述关联,直到事务的整个工作流已被处理为止。

[0081] 6.应用

[0082] 上面描述的类型的计算图的一个应用是在银行应用中处理金融事务。一般地,不同类型的事务要求不同类型的计算图。典型的计算图与客户事务的类型和处理该事务所需的“后端”服务的某种组合相关联。例如,事务可以是ATM请求、银行出纳员(bank teller)输入,以及计算机或者web服务器之间的企业对企业(business to business)事务。不同客户可能具有不同的后端系统,特别是当银行合并并且客户被从不同的原银行组合时。他们的帐户可能在非常不同的后端系统上维护,尽管他们都是收购方(acquiring)银行的客户。因此,图中的不同顶点可被用来处理不同的事务。不同的服务可被和图中的顶点相关联。例如,一些顶点可以和例如更新余额、在帐户中存钱,或者进行开户以便在帐户中保存资金的功能相关联。在某些实施方案中,过程到顶点的即时(on-the fly)分配避免了使用于未使用的顶点的过程保持空闲的开销。

[0083] 以每个事务为基础分配图实例的优点是它允许数据流的并行化,否则,这些数据流将必须被串行处理。分配给不同事务的图实例可以按和它们开始的顺序不同的顺序结束,例如,如果第一个事务比第二个事务更复杂。在串行化系统可能仍在处理第一事务时,这可以允许第二个图实例被释放并可用于处理第三事务。

[0084] 7.错误处理

[0085] 以每个事务为基础分配图实例的优点是因执行图实例时的错误所致的故障

(failure)可以被包含到该事务，并且不危害其他图实例的并行处理。通过把提交计算图的结果延迟到整个事务完成，如果发生错误，则数据可以被“回滚(rolled-back)”到其在系统开始处理该事务之前所处的状态。可以用几种方式处理错误。

[0086] 在某些例子中，“错误处理”组件被包括在图中。错误处理组件是一种特殊情况，因为其不是必须执行以使图完成。在任一顶点处的组件产生错误的情况下，不是导致整个计算中止(abort)，而是把图的执行重定向到错误处理组件。给定组件和错误处理组件之间的显式关系(包括从组件的输出端口到错误处理组件的输入端口的工作流)被称为异常流(exception flow)。调度器从图实例去除作为失败计算的一部分的工作元素，并且错误处理组件提供输出，该图可以使用所述输出来提供错误消息作为到调用其的过程的输出。依赖于实施方案，除了通过异常流以外，错误处理组件还可以通过其它方式接收数据输入。

[0087] 图3B示出了执行图和处理在图中出现的错误的示范性过程350的流程图。调度器根据连线处理图组件中的工作元素的工作流(步骤360)。当调度器识别出在图组件中已出现错误时(步骤370)，调度器把处理重定向到错误处理组件。这种重定向的一个方面是根据到该错误处理组件的任何异常流把工作元素定向到该错误处理组件(步骤380)。如下面更详细地描述的那样，处理异常流使错误处理组件能够向图外部的过程提供错误信息(步骤390)，图外部的过程代表在图开始处理其中出现错误的事务之前图处理的状态。

[0088] 对于图中的任何组件，存在指定的错误处理组件。这可以是从另一图组件直接接收异常流输出或者其他错误数据输出的组件，或者，它可以被定义为用于组件集合的指定的错误处理组件，而不管其是否接收到异常流。在某些例子中，如图4A到4B中所示那样处理异常流。在这个例子中，图被设计用于执行事务型(transactional)计算，并且示出了预订902和公布904组件，但是，相同的技术可以在用于非事务型工作流的图中使用。在图4A中，调度器已经激活图900。从第一组件预订902开始，在任何非异常路径下游中的每一个组件被标记为“使能”。异常路径是在异常情况下(例如，如上所述引向错误处理组件的异常流)只接收工作元素流或者其他错误数据的路径。这被称为使能传播(enablement propagation)。如果在其他组件下游的给定组件的输入中的任何一个被连接到使能的上游组件，则该给定组件被使能。也就是，复制(replicate)906、重新格式化908、调用web服务910、上滚(rollup)912、融合(fuse)914和公布(publish)904都被使能，但是未从任何使能的组件接收到非异常输入流的错误处理器916及其下游的两个组件回滚918和错误日志920仍保持“禁止”。

[0089] 如果出现错误，则调度器停止(halt)出错组件的执行，允许任何其他已在执行的组件执行完毕，并把任何相关数据(例如，已完成的组件的异常流输出，或者出错组件的“错误报告输出”)传播到错误处理组件。例如，如果调用web服务组件910引发(trigger)了错误，则来自复制组件906的异常流和来自调用web服务组件910的拒绝端口(reject port)921的错误报告输出分别在输入922和924处输入到错误处理组件916。错误报告输出端口(被示为图900中某些组件的底部的端口)可被用来提供关于已发生的任何错误的信息，包括例如描述发生了什么错误、哪里发生了错误以及与该错误相关联的任何拒绝的工作元素的信息。

[0090] 在这个例子中，对于复制组件906，存在三个错误报告输出端口。拒绝端口921提供可能已经导致该错误或者以某种方式和该错误相关的工作元素。错误端口923提供描述关

于该错误的相关信息的错误消息。日志端口925能够可选地提供记录发生了该错误的信息。即使没有错误发生，日志端口925也能够提供关于在执行的正常过程期间的事件的日志信息。在这个例子中，对于可能需要使用拒绝端口921的那些组件(例如，调用web服务组件910)，拒绝端口921被显式地示为被连接。但是，错误端口923和日志端口925未被显式地示为被连接，而是具有到错误处理组件916的隐式连接。例如，端口可以由开发者连接，然后使用接口控制隐藏。在某些实施方案中，系统可以自动地确定到默认错误处理组件的隐式连接，随后这些隐式连接可以被开发者无效。对于较大和/或复杂的图，用于一个或更多个类型的错误报告端口的这种“隐式连线”提高了开发者对图的直观理解(visual comprehension)，这是基于图的编程的益处之一。在某些实施方案中，可以提供直观线索来指示一个端口被隐式地连接到另一个组件的端口(例如，图标或者阴影的或者彩色的端口)。隐藏的隐式工作流连接中的一些或者全部也可以响应于用户请求(例如，点击按钮或者在端口上悬浮)而被显露为显式连线。

[0091] 如果在错误出现前该复制组件906已经结束操作，则来自复制组件906的异常流可能已经在输入922处排队。然后，调度器使能错误处理组件(在这个例子中为916)，禁止出错组件(在这个例子中910)，并从错误处理组件进行使能传播(在这个例子中使能918、904、920)。被禁止的出错组件下游的任何组件只要未从错误处理组件下游的使能组件接收到流，就被禁止(在这个例子中禁止912和914)。最后，向使能组件提供流的任何剩余组件被使能(在这个例子中使能906和902)。

[0092] 因此，这个过程的结果由图4B中的“<使能>”和“<禁止>”组件的指示示出。在错误处理器916之后把公布组件904连接回该流允许该事务被完成，尽管具有用于其输出的错误消息。丢弃已传播到现在禁止的组件的数据，例如来自重新格式化组件908的输出。

[0093] 如上面所指出的那样，数据可以作为异常流的一部分或者作为另一组件的错误报告输出的一部分流到错误处理组件。在出现错误之前可用的数据，例如，来自图4B中的复制模块906的输出数据，被暂时保持在错误处理器916的输入队列中，直到确实需要它时为止。如果该图无错误地完成，则错误处理器916永不激活并且数据被丢弃。如果的确出现错误，则错误处理器916使用其已经接收到的任何输入数据来拟定(formulate)响应。如图4B中所示，在某些例子中，使用回滚组件918。来自复制组件906的输入数据告诉错误处理器916在该图开始处理事务之前事情(things)的状态是什么。错误处理器916将它输出到回滚组件918，回滚组件918使用它把被其他组件修改的任何数据恢复到其在该事务执行之前的状态。然后，执行流既去往记录错误的错误日志920，也去往公布组件904，所以错误可以被报告并被任何递送到图900的较高级别的过程适当地处理。从任一组件到错误处理器916的异常流也可以包括数据。如果除了来自复制组件906的原始数据以外还存在到错误处理器916的输入，例如来自调用web服务组件910的错误输出或者来自任何其他组件(未示出)的异常流，则这可以被用来在错误日志或者公布组件拟定更详细的错误消息。

[0094] 在某些例子中，如图5中所示，图包括被实施为子图的顶点，例如子图950，每一个子图可以具有其自己的错误处理组件952。因此，可以存在一种子图层次结构，其具有顶层图，该顶层图具有作为子图的顶点，这些子图处于较低“图层”，依此类推。如果在子图950的任何组件954、956、958、960、962中出现错误，则处理流被路由到错误处理组件952，它在子图错误报告端口974上提供错误报告输出。错误处理组件952的范围是子图950。错误处理组

件可以具有接收来自另一图元(graph element)的异常流(例如,元素954)或者另一图元(例如,元素958)的错误输出959的输出的输入966、968,所示另一图元自身可以是嵌套的子图。在某些例子中,如果错误处理组件具有多个输入,则只有最近接收数据的输入被使用。如果子图950的所有组件成功地完成它们的操作,则输出(工作流)被递送到正常子图输出端口970,并且子图950以外的处理流正常地继续进行。如果出现错误,则其可以被处理并在错误流输出972或者错误报告输出974上报告。在其他的例子中,也可以在标准输出970上报告错误。

[0095] 如果子图不具有错误处理,其错误在该子图是其一部分的子图层次结构中向上流动,直到它们到达具有错误处理的图层为止,此刻,该层的错误处理组件被激活。

[0096] 暂时保存在错误处理组件的输入处的数据可以是工作流的子集,其可以是和事务相关联的全部数据,或者,其可以是整个数据流。如果错误处理组件具有错误输出端口,其将基于暂时保存的数据或者从具有错误的组件接收到的输入来输出导致错误的记录或者其他错误信息。如果它不具有这样的端口,则它可以在其输出端口上简单地输出违规记录(offending record)作为正常输出。

[0097] 如果子图不具有错误处理,其组件中的错误在该子图是其一部分的子图层次结构中向上流动,直到它们到达不具有错误处理的图层为止,此刻,该层的错误处理组件接收适当的输入,并产生适当的错误输出。

[0098] 错误处理可以允许在基于图的计算中通常会避免的循环图排列。例如,如图6中所示,在图1100中,来自错误处理器1104下游的计算组件1112的错误输出1116把流返回到该同一错误处理器1104。错误处理器1104也从预订组件1102接收输入并把输出提供给回滚组件1106,如图4A中所示。回滚组件1106基于由预订组件1102输入到错误处理器1104的数据,把数据返回到尝试该失败的计算之前其所处的状态。计数器组件1108可以从回滚组件1106接收流,并在把流返回到收集组件1110之前使其值递增。计算组件1112可以以几种不同方式使用从计数器组件1108输入的值。例如,它可以在执行其计算之前查考(consult)该值,以查看其是否应该对其操作做出某些改变。它也可以在错误之后查考计数器,以查看是否已经做出了某个阈值数量的尝试。如果已经超过阈值,则其把其输出定向到通向第二错误处理器1120的第二错误输出1118,而不是再次通过输出1116把错误输出返回到错误处理器1104。如果未使用计数器组件,则可以使用某个其他技术来打破该循环并确保该图最终完成。

[0099] 为了确保循环图被明确地定义(well-defined),在错误时将被使能的元素的集合被基于图的拓扑预先确定,而不是如上所述那样当需要时确定。

[0100] 在某些例子中,可以使用其他规则来确保错误处理正确地工作。例如,在某些实施方案中,可以只在图内的一个组件的异常端口上触发错误处理(可以忽略任何同时的错误)。如果图组件或者子图连线到错误处理组件,则其必须将该组件用在任何错误上。如果图组件或者子图未连线到错误处理组件,则错误必须由用于当前范围的通用错误处理器处理。每一个图组件一般就与一个错误处理器相关联。这些规则可以依赖于系统的要求被修改或者组合。它们在需要紧密控制用于每一个事务的过程的情况下能够用得到。

[0101] 在某些例子中,当出现错误时,操作系统确定哪个错误处理组件和经历该错误的组件相关联,然后确定应该使用到该错误处理组件的哪个输入流(如果有的话)。如果存在

多个输入，则使用最近有数据写入其中的那一个。

[0102] 如刚才描述的，错误处理可以是主动的，其中组件或者子图处理它们自己的错误并产生能够被其他组件用来诊断或者避开(work around)错误的错误代码，或者，它可以是被动的。在被动系统中，遇到错误的图简单地失效，并允许操作系统提供错误处理，例如，通过给调试过程提供堆栈转储。

[0103] 图的每一个组件隐式地连接到调度器，其不需要来自图的专门邀请来干预和处理错误。调度器可以从图实例去除和错误有关的数据，并且，在某些例子中，不需要知道错误的本质。在某些情况下，调度器可以把分配给图的资源分阶段地返回到它们相应的池中，从而允许图完成处理未受错误影响的工作元素。

[0104] 8.实施方案

[0105] 本发明可以被实现在硬件或者软件中，或者两者的组合(例如可编程逻辑阵列中)。除非另外规定，否则所描述的算法并非固有地与任何特定的计算机或者其他装置有关。具体来说，各种通用机可以和根据这里的教导所写的程序一起使用，或者，构造更专门化的装置(例如集成电路)来执行特定功能可能更方便。因此，本发明可以被实施在一个或更多个在一个或更多个被编程或者可编程计算机系统(它们可以是各种体系结构，例如分布式、客户端/服务器，或者网格)上执行的计算机程序中，所述计算机系统均包含至少一个处理器、至少一个数据存储系统(包括易失和非易失存储器，和/或存储元件)、至少一个输入设备或者端口，以及至少一个输出设备或者端口。程序代码被应用于输入数据以执行这里所描述的功能，并产生输出信息。输出信息被以已知方式应用于一个或更多个输出设备。

[0106] 每一个这样的程序均可以被用任何期望的计算机语言(包括机器、汇编，或者高级程序、逻辑，或者面向对象编程语言)实施以便和计算机系统通信。在任何情况下，语言可以是编译或者解释语言。

[0107] 每一个这样的计算机程序最好被存储或者下载到可被专用或者通用可编程计算机读取的存储介质或者设备(例如固态存储器或者介质，或者磁性或光学介质)，用于在计算机系统读取存储介质或者设备来执行这里所描述的程序时配置和操作计算机。发明系统也可以被视为被实施为计算机可读存储介质，配置有计算机程序，其中，如此配置的存储介质导致计算机系统以特定和预先定义的方式工作来执行这里所描述的功能。

[0108] 要理解，前面的描述旨在说明而非限制本发明的范围，本发明由所附权利要求的范围限定。其他的实施例在下列权利要求的范围内。

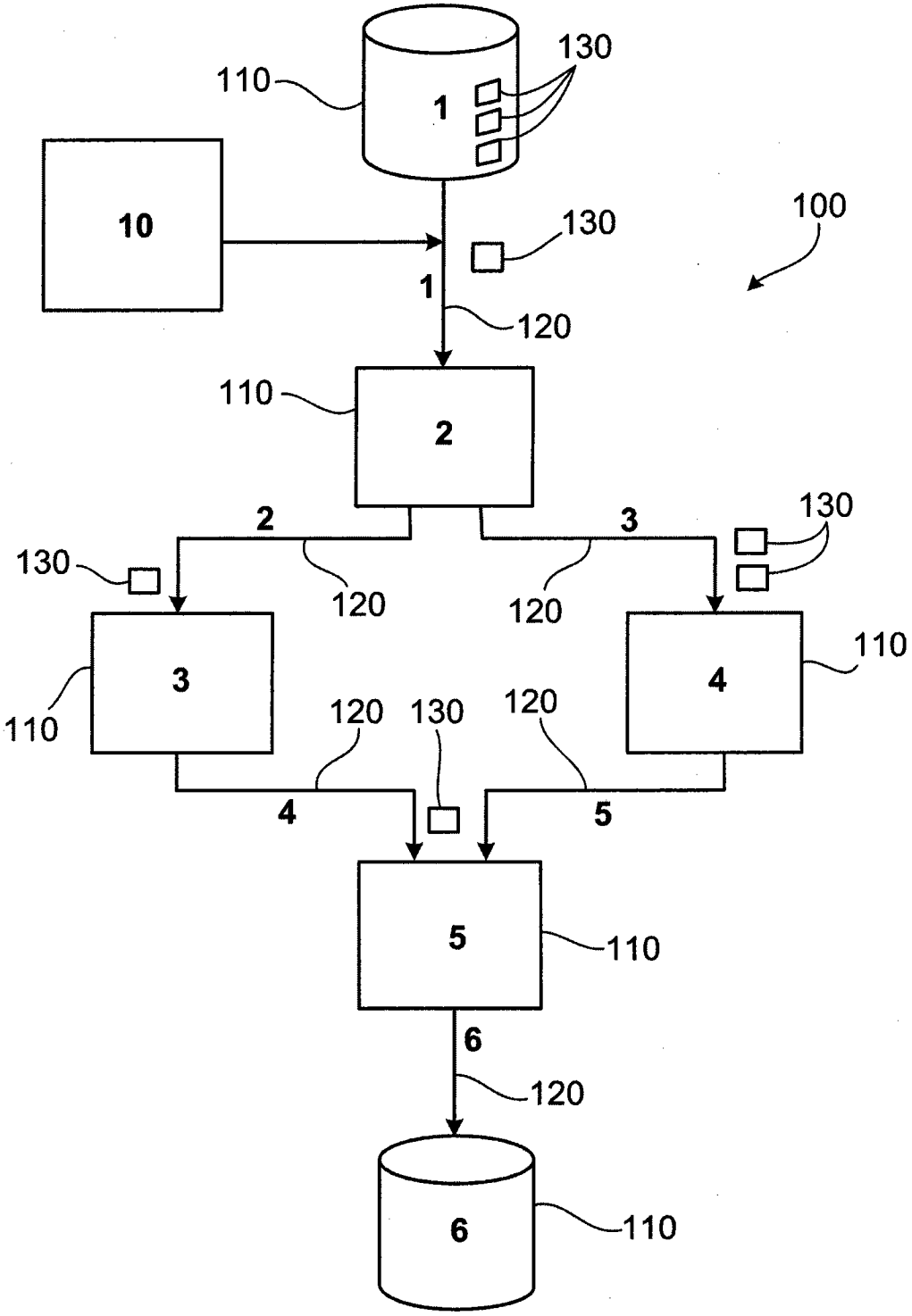

110
130
1
10
130
100
1
120
110
2
2
3
130
130
120
120
3
110
4
110
120
130
120
4
5
5
110
6
120
6
110

图1

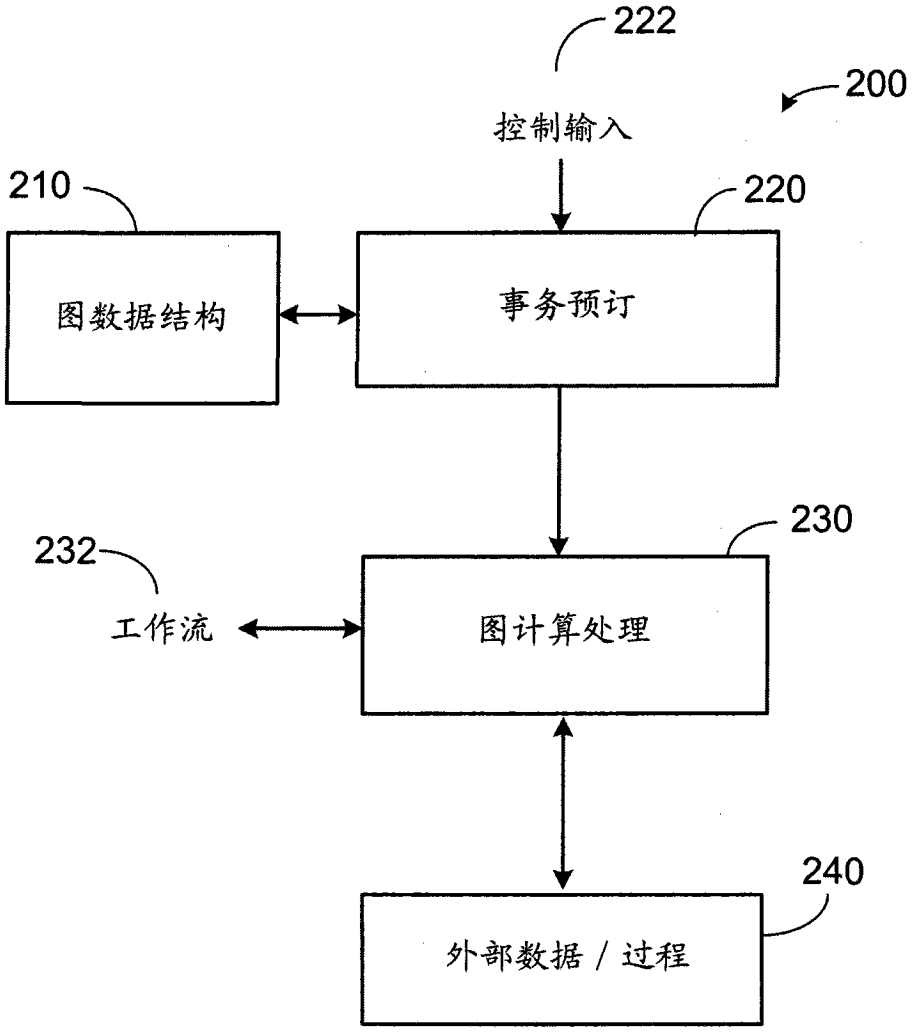


图2

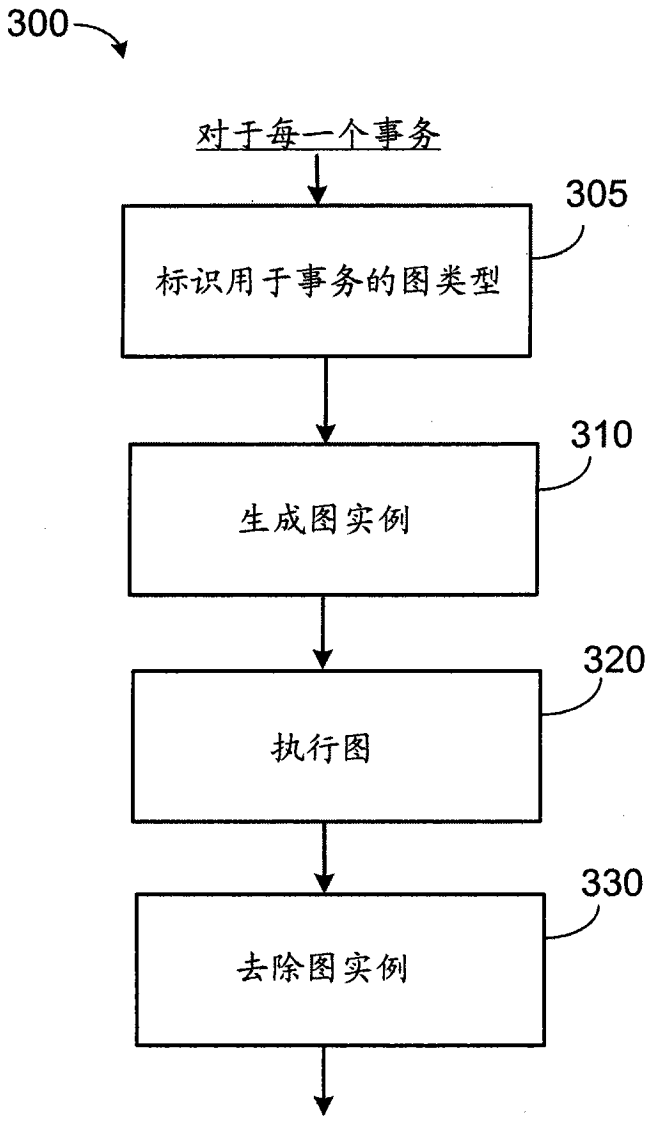

300
对于每一个事务
305
标识用于事务的图类型
310
生成图实例
320
执行图
330
去除图实例

图3A

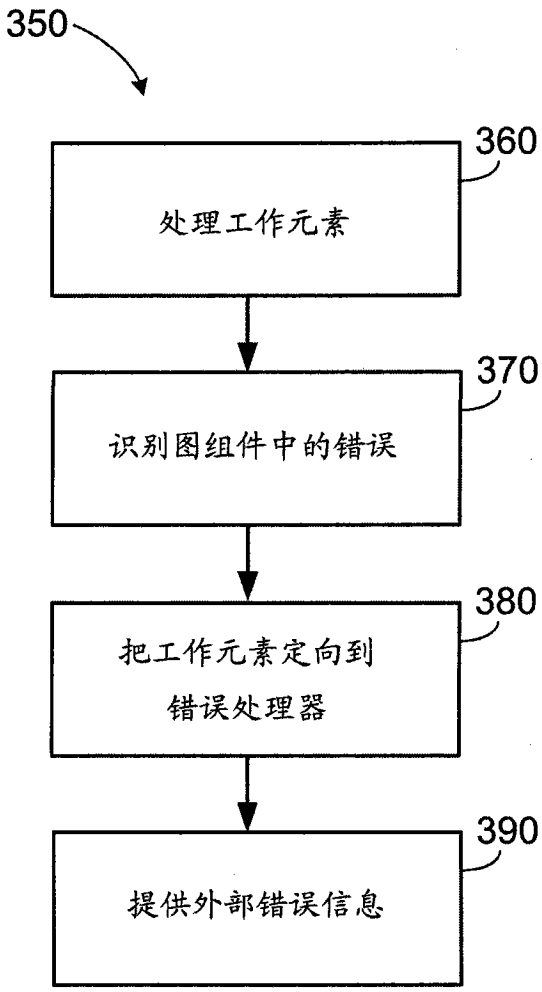

350
360
处理工作元素
370
识别图组件中的错误
380
把工作元素定向到
错误处理器
390
提供外部错误信息

图3B

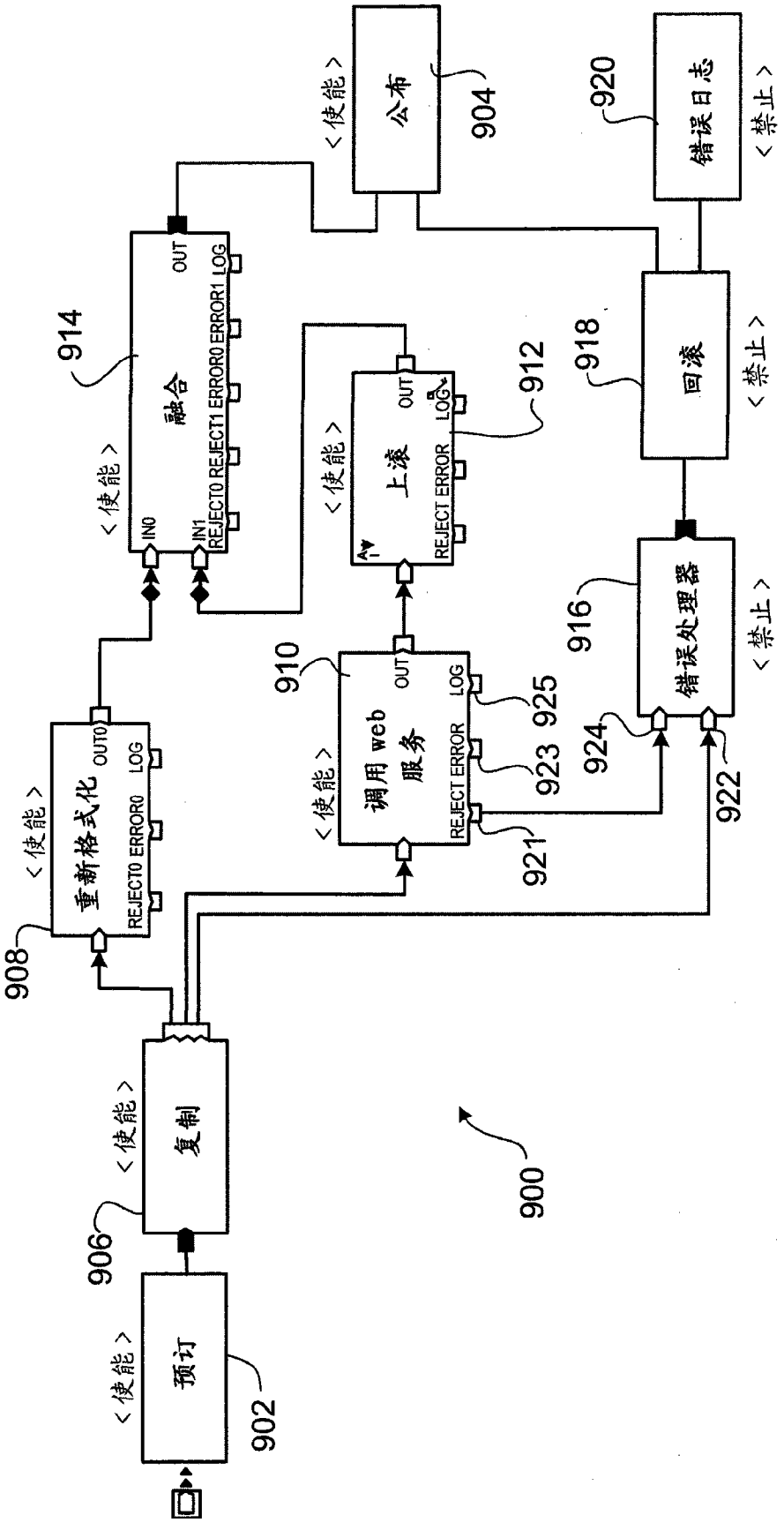

预订
902
复制
906
重新格式化
908
调用 web 服务
910
融合
914
上滚
912
公布
904
回滚
918
错误处理器
916
错误日志
920
900
使能
禁止
OUT
LOG
IN0
IN1
REJECT0
REJECT1
ERROR0
ERROR1
REJECT ERROR
921
922
923
924
925

图4A

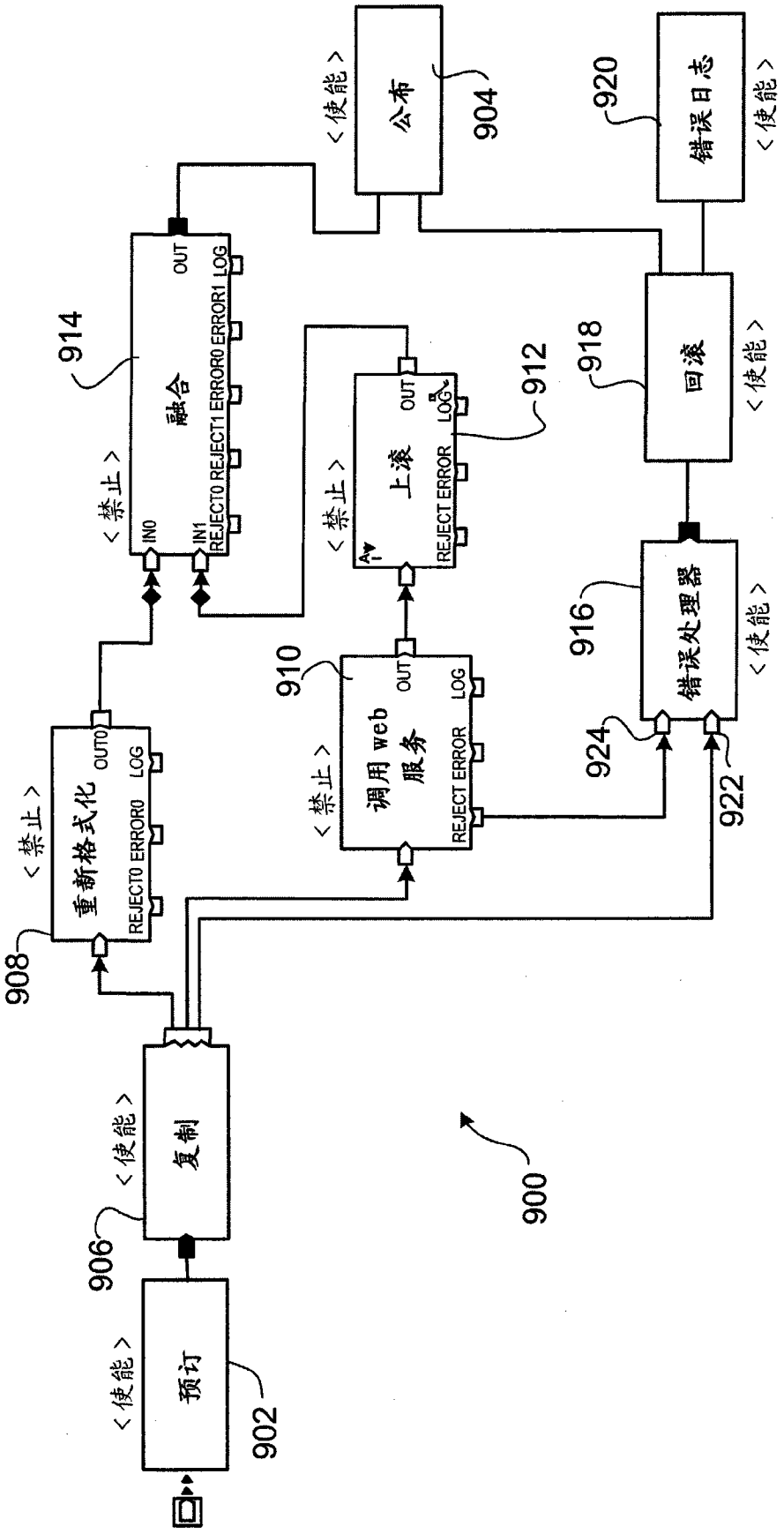

预订
902
<使能>
复制
906
<使能>
重新格式化
908
<禁止>
OUT0
LOG
REJECT0 ERROR0
调用 web 服务
910
<禁止>
OUT
LOG
REJECT ERROR
上滚
912
<禁止>
OUT
LOG
REJECT ERROR
融合
914
<禁止>
IN0
IN1
OUT
LOG
REJECT0 REJECT1 ERROR0 ERROR1
公布
904
<使能>
回滚
918
<使能>
错误处理器
916
<使能>
924
922
错误日志
920
<使能>
900

图4B

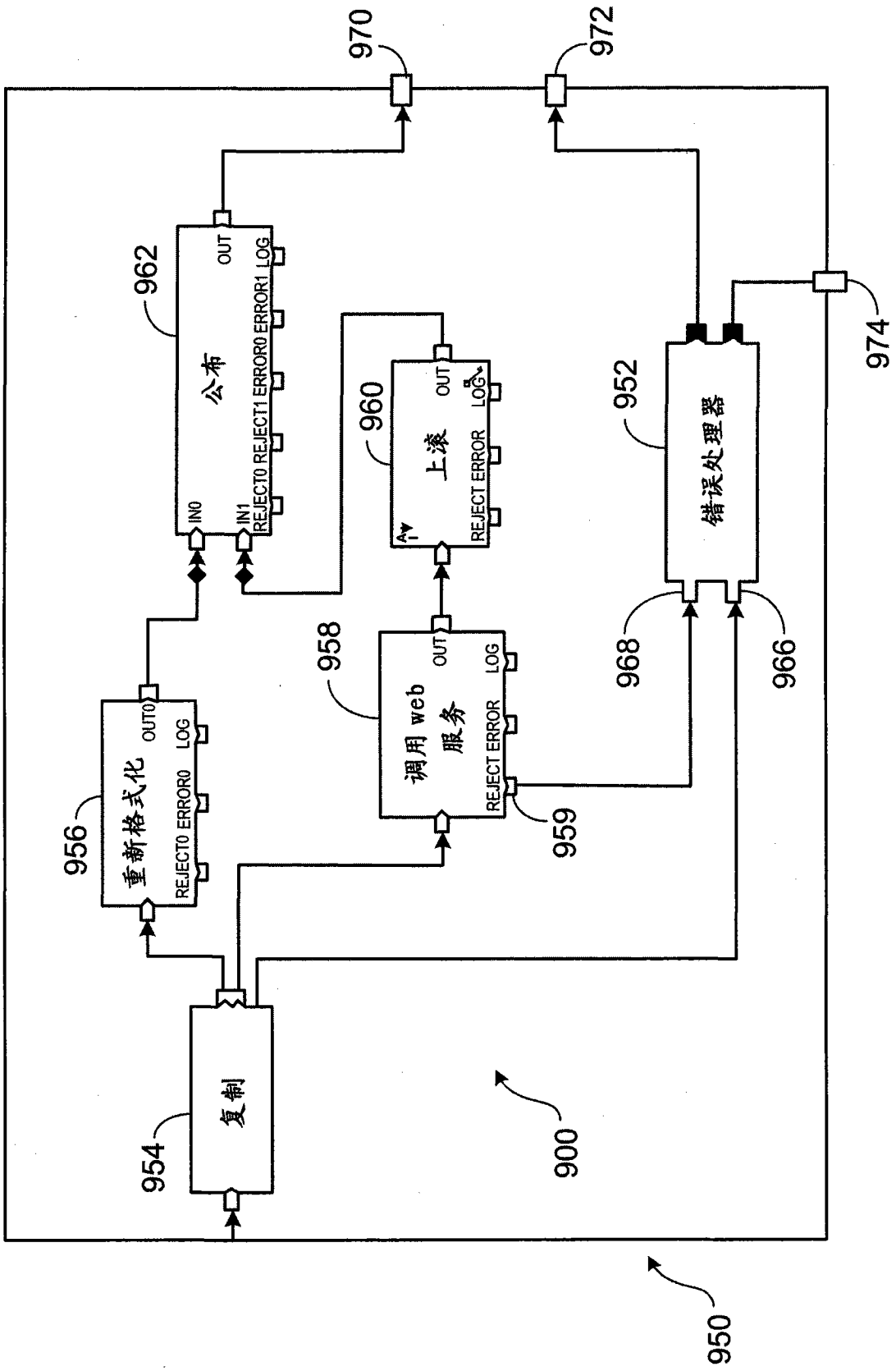

970
972
962
公布
OUT
IN0
IN1
REJECT0 REJECT1 ERROR0 ERROR1 LOG
960
上滚
OUT
LOG
REJECT ERROR
952
错误处理器
974
958
调用 web 服务
OUT
LOG
REJECT ERROR
959
956
重新格式化
OUT0
LOG
REJECT0 ERROR0
968
966
954
复制
900
950

图5

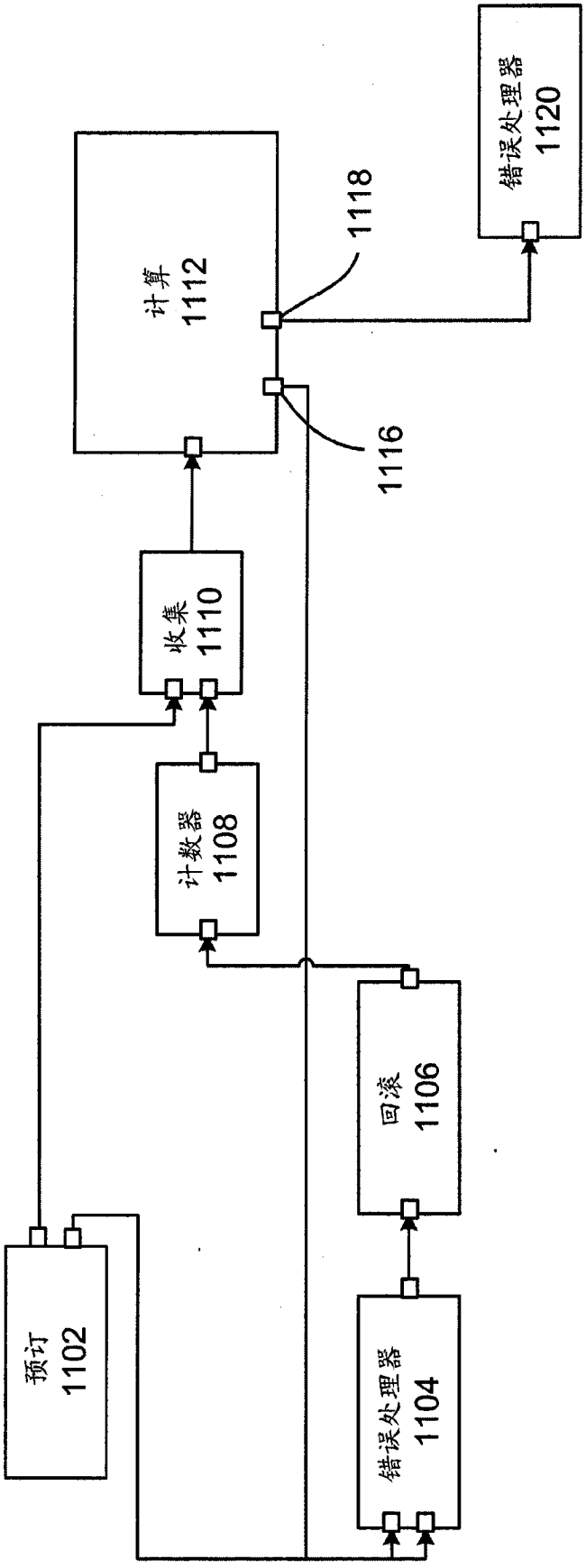

错误处理器
1120
1118
计算
1112
1116
收集
1110
计数器
1108
回滚
1106
预订
1102
错误处理器
1104

图6