(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 101681294 B
(45)授权公告日 2013.12.25

(21)申请号 200880020189.3
(22)申请日 2008.06.18
(30)优先权数据
11/824,379 2007.06.29 US
(85)PCT申请进入国家阶段日
2009.12.14
(86)PCT申请的申请数据
PCT/US2008/067343 2008.06.18
(87)PCT申请的公布数据
WO2009/006023 EN 2009.01.08
(73)专利权人 微软公司
地址 美国华盛顿州
(72)发明人 M·塔耶费尔
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 陈斌 钱静芳
(51)Int.Cl.
G06F 12/00(2006.01)

(56)对比文件
US 6807582 B1,2004.10.19, 全文.
US 6516404 B1,2003.02.04, 全文.

审查员 王可

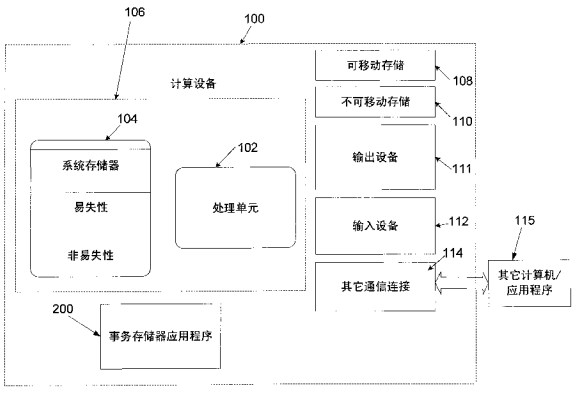
权利要求书3页 说明书5页 附图10页

(54)发明名称

用于存储器事务分组的方法和系统

(57)摘要

公开了用于提供在事务存储器系统下操作的程序中使用的事务分组特征的各种技术和方法。事务分组特征可用于允许创建包含相关事务的事务分组。事务分组可用来增强程序的性能和/或操作。例如,可以对不同的事务分组使用不同的锁定和版本化机制。在运行事务时,硬件事务存储器执行机制可以用于一个事务分组,而软件事务存储器执行机制用于另一事务分组。



CN 101681294 B

1. 一种用于支持事务分组的方法，所述方法包括以下步骤：

提供用于事务存储器系统的事务分组特征，所述事务分组特征可用于允许创建其中已知特定分组内的所有事务只访问与该分组之外的任何事务所访问的数据不相交的数据的事务分组；

向所述事务分组中的至少某一些提供不同的锁定机制，其中各锁定机制彼此在功能上不兼容；以及

向所述事务分组中的至少某一些提供不同的版本化机制，其中各版本化机制彼此在功能上不兼容。

2. 如权利要求 1 所述的方法，其特征在于，一些锁定机制是通过软件方法实现的而其它一些是使用硬件方法实现的。

3. 如权利要求 1 所述的方法，其特征在于，一些版本化机制是使用硬件方法实现的而其它一些是使用软件方法实现的。

4. 如权利要求 1 所述的方法，其特征在于，所述方法进一步包括：

使用所述事务分组提供专用争用管理。

5. 如权利要求 1 所述的方法，其特征在于，所述方法进一步包括：

提供用于允许对事务分组进行命名的命名特征。

6. 如权利要求 5 所述的方法，其特征在于，所述命名特征在调试程序中使用。

7. 如权利要求 5 所述的方法，其特征在于，所述命名特征在剖析程序中使用。

8. 如权利要求 1 所述的方法，其特征在于，所述事务分组中的至少某一些是手动地分配的。

9. 如权利要求 8 所述的方法，其特征在于，所述事务分组特征用于允许程序员在源代码内手动地指定事务分组。

10. 如权利要求 1 所述的方法，其特征在于，所述事务分组中的至少某一些是自动地分配的。

11. 如权利要求 10 所述的方法，其特征在于，所述事务分组特征用于允许编译器自动地标识所述事务分组。

12. 如权利要求 10 所述的方法，其特征在于，所述事务分组特征适用于允许运行时环境自动地标识事务分组。

13. 如权利要求 1 所述的方法，其特征在于，所述事务分组特征适用于改进程序的性能。

14. 一种用于对不同的事务分组使用不同的锁定和版本化机制的方法，所述方法包括以下步骤：

将一组事务分组到多个事务分组中的第一事务分组；以及

提供争用管理机制，所述争用管理机制被配置成检测程序的多个事务的多个并发执行事务之间的冲突；

其中所述该组事务来自所述多个事务，并且

其中所述多个事务中的每个都被实现为被配置成访问共享存储器的代码，以及

其中所述该组事务被配置成访问与其他数据不相交的数据，以及

其中所述多个事务中的不在所述该组事务中的其他事务被配置成访问所述其他数据，

以及

其中所述多个事务分组中的每个都被配置成利用多个锁定和版本化机制之一，该多个锁定和版本化机制之一不与被所述多个事务分组中的其他事务分组利用的所述多个锁定和版本化机制的其他锁定和版本化机制兼容，以及

其中所述多个事务分组中的所述其他事务分组中的至少一个被配置成并发地利用所述多个锁定和版本化机制的其他锁定和版本化机制中的至少一个；

将来自所述程序的所述多个事务的第二组事务分组到所述多个事务分组中的第二事务分组；

由事务存储器执行机制执行所述该组事务；

由所述事务存储器执行机制与所述该组事务并发地执行所述第二组事务。

15. 一种用于对不同的事务分组使用不同的锁定和版本化机制的系统，所述系统包括：

用于将一组事务分组到多个事务分组中的第一事务分组的装置；以及

用于提供争用管理机制的装置，所述争用管理机制被配置成检测程序的多个事务的多个并发执行事务之间的冲突；

其中所述该组事务来自所述多个事务，并且

其中所述多个事务中的每个都被实现为被配置成访问共享存储器的代码，以及

其中所述该组事务被配置成访问与其他数据不相交的数据，以及

其中所述多个事务中的不在所述该组事务中的其他事务被配置成访问所述其他数据，以及其中所述多个事务分组中的每个都被配置成利用多个锁定和版本化机制之一，该多个锁定和版本化机制之一不与被所述多个事务分组中的其他事务分组利用的所述多个锁定和版本化机制的其他锁定和版本化机制兼容，以及

其中所述多个事务分组中的所述其他事务分组中的至少一个被配置成并发地利用所述多个锁定和版本化机制的其他锁定和版本化机制中的至少一个；

用于将来自所述程序的所述多个事务的第二组事务分组到所述多个事务分组中的第二事务分组的装置；

用于由事务存储器执行机制执行所述该组事务的装置；

用于由所述事务存储器执行机制与所述该组事务并发地执行所述第二组事务的装置。

16. 一种用于支持事务分组的方法，所述方法包括以下步骤：

提供事务分组特征，所述事务分组特征被配置成将一组事务置于多个事务分组之一；以及提供争用管理机制，所述争用管理机制被配置成检测程序的多个事务的多个并发执行事务之间的冲突；

其中所述该组事务来自所述多个事务，并且

其中所述多个事务中的每个都被实现为被配置成访问共享存储器的代码，以及

其中所述该组事务被配置成访问与其他数据不相交的数据，以及

其中所述多个事务中的不在所述该组事务中的其他事务被配置成访问所述其他数据，以及

其中所述多个事务分组中的每个都被配置成利用多个锁定和版本化机制之一，该多个锁定和版本化机制之一不与被所述多个事务分组中的其他事务分组利用的所述多个锁定

和版本化机制的其他锁定和版本化机制兼容，以及

其中所述多个事务分组中的所述其他事务分组中的至少一个被配置成并发地利用所述多个锁定和版本化机制的其他锁定和版本化机制中的至少一个。

用于存储器事务分组的方法和系统

[0001] 背景

[0002] 计算机随时间不断变得更加强大，具有更多的处理能力和存储器来处理高级操作。这一趋势最近将焦点从日益增长的单处理器时钟速率移开，并趋向于增加单个计算机中可用处理器的数量。软件开发者想要利用计算机处理能力上改进的优势，以允许他们的软件程序在采用新硬件时更快地执行。然而，对于该新硬件趋势，这要求不同的方法：开发者必须安排特定软件程序的一个或多个任务“并发地”（有时称为“并行”）执行，以使同一逻辑操作可以同时利用多个处理器，并在向该软件在其上运行的计算机添加更多处理器时提供更好的性能。

[0003] 事务存储器被设计成通过向程序代码区域提供原子性和隔离性来简化并发程序的开发。事务存储器（TM）是类似于数据库事务的、用于在并发计算中控制对共享存储器的访问的并发控制机制。事务存储器的上下文中的事务是对共享存储器执行一系列读取和写入的一段代码。TM 用作传统锁定机制的替换。TM 允许更简单地编写并发程序。事务指定应当好像隔离地执行的代码序列，而实际上它在正常的多线程环境中与许多并发活动一起执行。这一隔离错觉可以通过对象或存储器范围的细粒度锁定，以及通过以在发现事务与某一其它事务相冲突的情况下允许回退该事务的影响的模式执行来实现。如果数据访问受这些锁定和回退机制保护，则可以说该访问被“事务化”。

[0004] 不同的锁定和版本化机制是可能的，包括若干基于软件的和基于硬件的方法。不同的机制具有特征和质量，使得其各自在不同的情况是合适的或优选的。在单个进程中组合不同的机制通常是不可能的，从而导致对通用机制的选择，而通用机制为了实现一般适用性通常使性能受损。

[0005] 概述

[0006] 公开了用于提供在事务存储器系统下操作的程序中使用的事务分组特征的各种技术和方法。事务分组特征可用于允许创建包含相关事务的事务分组。事务分组可用来增强程序的操作。定义各事务分组，以便知道每一分组中的事务操作不相交的数据，这允许每一这样的分组内的不兼容的锁定和版本化机制，转而又允许精细地调节用于每一特定分组的具体机制。

[0007] 提供本概述以便以简化形式介绍将在以下详细描述中进一步描述的一些概念。本概述不旨在标识所要求保护的主题的关键特征或必要特征，也不旨在用于帮助确定所要求保护的主题的范围。

[0008] 附图简述

[0009] 图 1 是一个实现的计算机系统的图示。

[0010] 图 2 是在图 1 的计算机系统上操作的一个实现的事务存储器应用程序的图示。

[0011] 图 3 是图 1 的系统的一个实现的高级处理流程图。

[0012] 图 4 是图 1 的系统的一个实现的处理流程图，其示出在允许程序员对事务进行分组时所涉及的各阶段。

[0013] 图 5 是图 1 的系统的一个实现的处理流程图，其示出在提供基于特定试探法来自

动地对事务进行分组的语言编译器时所涉及的各阶段。

[0014] 图 6 是图 1 的系统的一个实现的处理流程图，其示出在提供自动地对事务进行分组的运行时环境时所涉及的各阶段。

[0015] 图 7 是图 1 的系统的一个实现的处理流程图，其示出在向不同的事务分组提供专用争用管理时所涉及的各阶段。

[0016] 图 8 是图 1 的系统的一个实现的处理流程图，其示出在向不同的事务分组提供专用锁定和版本化机制时所涉及的各阶段。

[0017] 图 9 是图 1 的系统的一个实现的处理流程图，其示出在对相关事务的分组进行命名以增强调试或其它过程时所涉及的各阶段。

[0018] 图 10 是多个事务分组的图示。

[0019] 详细描述

[0020] 此处的技术和方法可以在事务存储器系统的一般上下文中描述，但本技术和方法也用作除此之外的其它目的。在一个实现中，此处所描述的一个或多个技术可被实现为诸如微软®.NET 框架等框架程序内的、或来自为开发者提供开发软件应用程序的平台的任何其它类型的程序或服务的特征。在另一实现中，此处所描述的一个或多个技术被实现为涉及开发在并发环境中执行的应用程序的其它应用程序的特征。

[0021] 在一个实现中，提供事务分组特征以供在事务存储器系统下操作的程序中使用。事务分组特征允许将事务置于各个分组中。如果可以确定一组事务访问可证明与任何其它事务所访问的数据不相交的数据(例如读 / 写数据)，则这一组可被认为是“事务分组”。

[0022] 通过以上定义可知，作为分组的一部分的事务操作与其它分组中的其它事务所访问的读 / 写数据不相交的读 / 写数据。结果，对每一这样的分组实现不同的锁定和版本化机制成为可能，从而允许每一事务分组充分利用特别选择的最适于该分组中的事务所访问的数据的锁定和版本化算法。

[0023] 除分组中的事务所访问的特定数据之外，许多其它因素可以影响对分组内所使用的锁定和版本化算法的具体选择。例如，事务的持续时间或事务内的代码的性质是两个其它这样的因素。在一个实现中，可以在一进程内并发地使用通常不兼容的锁定和版本化机制，导致可能增加性能。

[0024] 确定事务何时可被分组可以通过多种手段来实现。一个实现可以充分利用程序员提供的注释来对分组进行区分，如图 4 所述。另一实现可以使用编译器试探法来自动地推断分组和分组成员资格，如图 5 所述。又一实现可以使用运行时环境来动态且自动地推断分组和分组成员资格，如图 6 所述。应当理解，创建分组和分配分组成员资格时所涉及的具体机制是很多的并可以用各种方式来组合。

[0025] 如图 1 所示，用于实现本系统的一个或多个部分的示例性计算机系统包括诸如计算设备 100 等计算设备。在其最基本的配置中，计算设备 100 通常包括至少一个处理单元 102 和存储器 104。取决于计算设备的确切配置和类型，存储器 104 可以是易失性的(如 RAM)、非易失性的(如 ROM、闪存等)或是两者的某种组合。该最基本配置在图 1 中由虚线 106 来示出。

[0026] 另外，设备 100 还可具有附加特征 / 功能。例如，设备 100 还可包含附加存储(可移动和 / 或不可移动)，包括但不限于磁盘、光盘或磁带。这样的附加存储在图 1 中由可移

动存储 108 和不可移动存储 110 示出。计算机存储介质包括以用于存储诸如计算机可读指令、数据结构、程序模块或其它数据等信息的任何方法或技术来实现的易失性和非易失性、可移动和不可移动介质。存储器 104、可移动存储 108 和不可移动存储 110 都是计算机存储介质的示例。计算机存储介质包括但不限于，RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、数字多功能盘（DVD）或其它光存储、磁带盒、磁带、磁盘存储或其它磁存储设备、或者可用于存储所需信息并且可由设备 100 访问的任何其它介质。任何这样的计算机存储介质都可以是设备 100 的一部分。

[0027] 计算设备 100 包括允许计算设备 100 与其它计算机 / 应用程序 115 进行通信的一个或多个通信连接 114。设备 100 还可以具有诸如键盘、鼠标、笔、语音输入设备、触摸输入设备等输入设备 112。还可以包括诸如显示器、扬声器、打印机等输出设备 111。这些设备在本领域中公知且无需在此处详细讨论。在一个实现中，计算设备 100 包括事务存储器应用程序 200。事务存储器应用程序 200 将在图 2 中更详细地描述。

[0028] 现在转向图 2 并继续参考图 1，示出了在计算设备 100 上操作的事务存储器应用程序 200。事务存储器应用程序 200 是驻留在计算设备 100 上的应用程序中的一个。然而，可以理解，事务存储器应用程序 200 可另选地或另外地被具体化为一个或多个计算机上的计算机可执行指令和 / 或与图 1 所示的不同的变型。另选地或另外地，事务存储器应用程序 200 的一个或多个部分可以是系统存储器 104 的一部分、可以在其它计算机和 / 或应用程序 115 上、或可以是计算机软件领域的技术人员能想到的其它此类变型。

[0029] 事务存储器应用程序 200 包括负责执行在此描述的技术中的一些或全部的程序逻辑 204。程序逻辑 204 包括用于提供允许将特定程序中的相关事务分组在一起的事务分组特征的逻辑 206（如以下参考图 3-6 描述的）；用于使用事务分组来提供专用争用管理的逻辑 210（如以下参考图 7 描述的）；用于向不同的事务分组提供不同的锁定和版本化机制的逻辑 212（如以下参考图 8 描述的）；用于对事务分组进行命名以增强调试或其它过程的逻辑 214（如以下参考图 9 描述的）；以及用于操作该事务存储器应用程序的其它逻辑 220。

[0030] 现在转向图 3-10 并继续参考图 1-2，更详细地描述了用于实现事务存储器应用程序 200 的一个或多个实现的各阶段。在某些实现中，图 3-10 的过程至少部分地在计算设备 100 的操作逻辑中实现。图 3 是事务存储器应用程序 200 的高级处理流程图。该过程在起始点 240 处开始，在那里使用软件、硬件、和 / 或其组合来提供事务存储器系统（阶段 242）。系统提供允许特定程序中的相关事务被手动地（如在图 4 中描述的）或通过程序（如在图 5 和图 6 中描述的）被分组在一起的事务分组特征（阶段 244）。系统使用事务分组来改进程序性能或以其它方式增强程序操作，如在图 7-10 所述（阶段 246）。该过程在结束点 248 处结束。

[0031] 图 4 示出在允许程序员对事务进行分组时所涉及的各阶段的一个实现。该过程在起始点 270 处开始，在那里从程序员接收输入以访问在事务存储器系统下执行的特定程序的源代码（阶段 272）。程序员向事务添加说明或以其它方式向事务分配分组（阶段 274）。系统使用指定分组来增强程序操作，如在图 7-9 更详细描述的（阶段 276）。该过程在结束点 278 处结束。

[0032] 图 5 示出在提供对事务自动地进行分组的语言编译器时所涉及的各阶段的一个实现。该过程在起始点 290 处开始，在那里提供用于编译在事务存储器系统下执行的程序

的语言编译器(阶段292)。在特定程序的编译时间,使用逻辑来确定是否有任何事务可以分组在一起(阶段294)。作为几个非限制性示例,编译器可以通过充分利用程序设计语言的特定语义或通过论证分组是可能的全局分析来将事务分组在一起。系统在程序中创建所标识的分组(阶段296)并随后使用指定分组来增强程序操作(阶段298)。该过程在结束点300处结束。

[0033] 图6示出在提供对事务进行分组的运行时环境时所涉及的各阶段的一个实现。该过程在起始点310处开始,在那里提供用于在事务存储器系统下运行程序的运行时环境(阶段312)。在运行特定环境时,使用逻辑来标识应被分组在一起的事务(阶段314)。作为几个非限制性示例,运行时环境可以通过标识按构造肯定操作不相交的读/写数据的事务的集合来将事务分组在一起。运行时可以按编译器所提供的提示来操作以训练其分析过程。运行时将所标识的事务分组在一起(阶段316)并使用分组的事务来改进特定程序的操作(阶段318)。该过程在结束点320处结束。

[0034] 现转向图7-9,将使用各示例来描述如何使用分组的事务来增强程序操作。图7示出在使用分组的事务提供专用争用管理时所涉及的各阶段的一个实现。该过程在起始点340处开始,在那里允许程序员定义一个或多个专用争用管理策略(阶段342)。对于每一策略,系统允许指定事务执行调度设置、事务异常中止处理设置、和/或其它设置(阶段344)。随后可以在进程中向每一事务分组分配先前定义的策略(阶段346)。系统使用这些策略来实现用于分组的事务的专用争用管理(阶段348)。

[0035] 争用管理是运行时系统用来每当在多个并发执行事务之间检测到冲突时选择适当的行为的机制。争用管理决定要保留哪一冲突事务(如果有的话)以及要中止哪一冲突事务。其还决定如何重新调度各个事务的执行以使它们可以运行完成。此处使用的专用争用管理指的是应用与给定运行时的默认试探法不同的争用管理试探法的能力。

[0036] 在一个实现中,通过对不同的分组应用不同的策略,可以得到程序的增强性能。例如,可以向特定事务分组分配给出这些事务所包含的那些类型的操作的最佳性能的一种类型的专用争用管理策略。可以向另一事务分组分配得到该另一事务所包含的那些类型的操作的最佳性能的另一专用争用管理策略。该过程在结束点350处结束。

[0037] 图8示出在向不同的事务分组提供不同的锁定和版本化机制时所涉及的各阶段的一个实现。该过程在起始点370处开始。系统使用逻辑来确定对每一事务分组使用什么类型的锁定和版本化(阶段372)并随后视需要调整用于每一事务分组的锁定和版本化(阶段374)。该过程在结束点376处结束。

[0038] 作为示例,一个事务存储器锁定和版本化机制可以与一个特定事务分组一起使用,而另一个可能不兼容的事务存储器锁定和版本化机制与另一个事务分组一起使用。

[0039] 查看非限制性示例来进一步说明在存在多个事务分组的情况下可以如何将不同的事务存储器机制组合在一起。一个事务分组可以使用缓冲更新方案来用于版本化,而另一分组可以用使用撤消记录的就地更新方案。通过对事务进行分组,可以用允许使用这些不兼容的事务存储器锁定和版本化机制的组合的方式来隔离数据,这可以使改进的总体性能成为可能。作为另一示例,可以在某些事务分组中使用快速但有限制的基于硬件的事务存储器机制,而在硬件限制对于其它事务分组中的事务而言是不可接受的情况下,可以在这些事务分组中使用不兼容且较慢的软件事务存储器机制。

[0040] 尽管先前刚刚提及的假想示例将该技术用于两个事务分组，但该概念可以用于超过两个分组并用于事务存储器锁定和版本化机制的各种组合，包括硬件和软件方法两者。

[0041] 图 9 示出在对相关事务的分组进行命名以增强调试或其它过程时所涉及的各阶段的一个实现。该过程在起始点 450 处开始，在那里允许用户和 / 或通过程序将相关事务分组在一起(阶段 452)。提供允许用户和 / 或通过程序向每一事务分组给出名称的命名特征(阶段 454)。系统随后显示或使用分组名称来增强调试、剖析、或其它过程(阶段 456)。例如，可以在调试程序或剖析程序中显示分组名称以允许用户更容易地标识该特定事务分组。该过程在结束点 458 处结束。

[0042] 图 10 是多个事务分组的图示。在所示示例中，第一事务分组 500 包含四个事务，其中一个嵌套在另一个内。第二事务分组 502 只包含单个事务。包含不同数量的事务分组和 / 或每一分组内包含不同数量的事务的多个其它事务分组场景也是可能的。

[0043] 尽管用对结构特征和 / 或方法动作专用的语言描述了本主题，但可以理解，所附权利要求书中定义的主题不必限于上述具体特征或动作。相反，上述具体特征和动作是作为实现权利要求的示例形式公开的。落入在此所述和 / 或所附权利要求所描述的实现的精神的范围内的所有等效方案、更改和修正都期望受到保护。

[0044] 例如，计算机软件领域普通技术人员将认识到，此处所讨论的示例可以在一个或多个计算机上不同地组织来包括比这些示例中所描绘的更少或更多选项或特征。

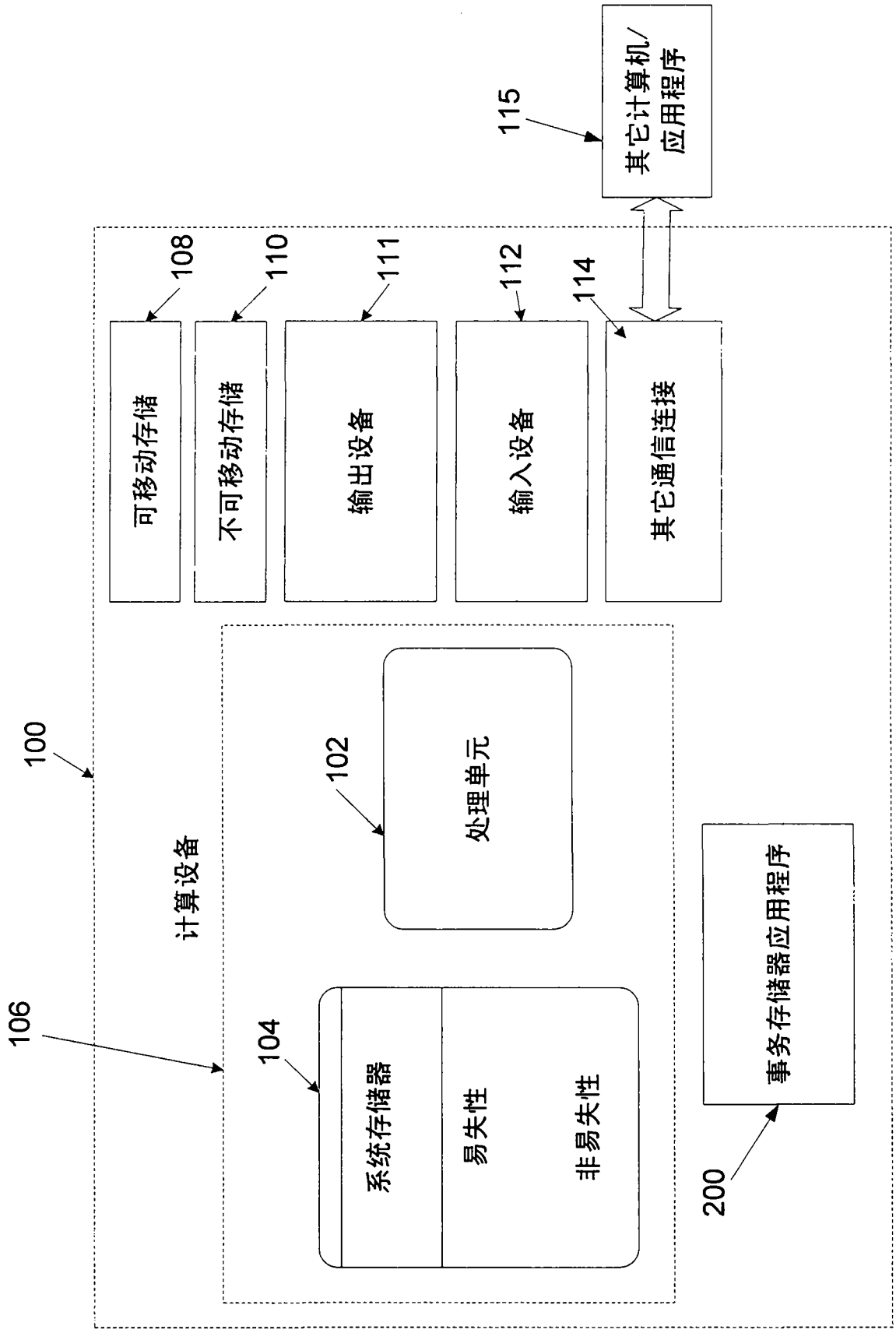

其它计算机/应用程序
115
可移动存储
不可移动存储
输出设备
输入设备
其它通信连接
108
110
111
112
114
100
计算设备
处理单元
102
106
104
系统存储器
易失性
非易失性
事务存储器应用程序
200

图 1

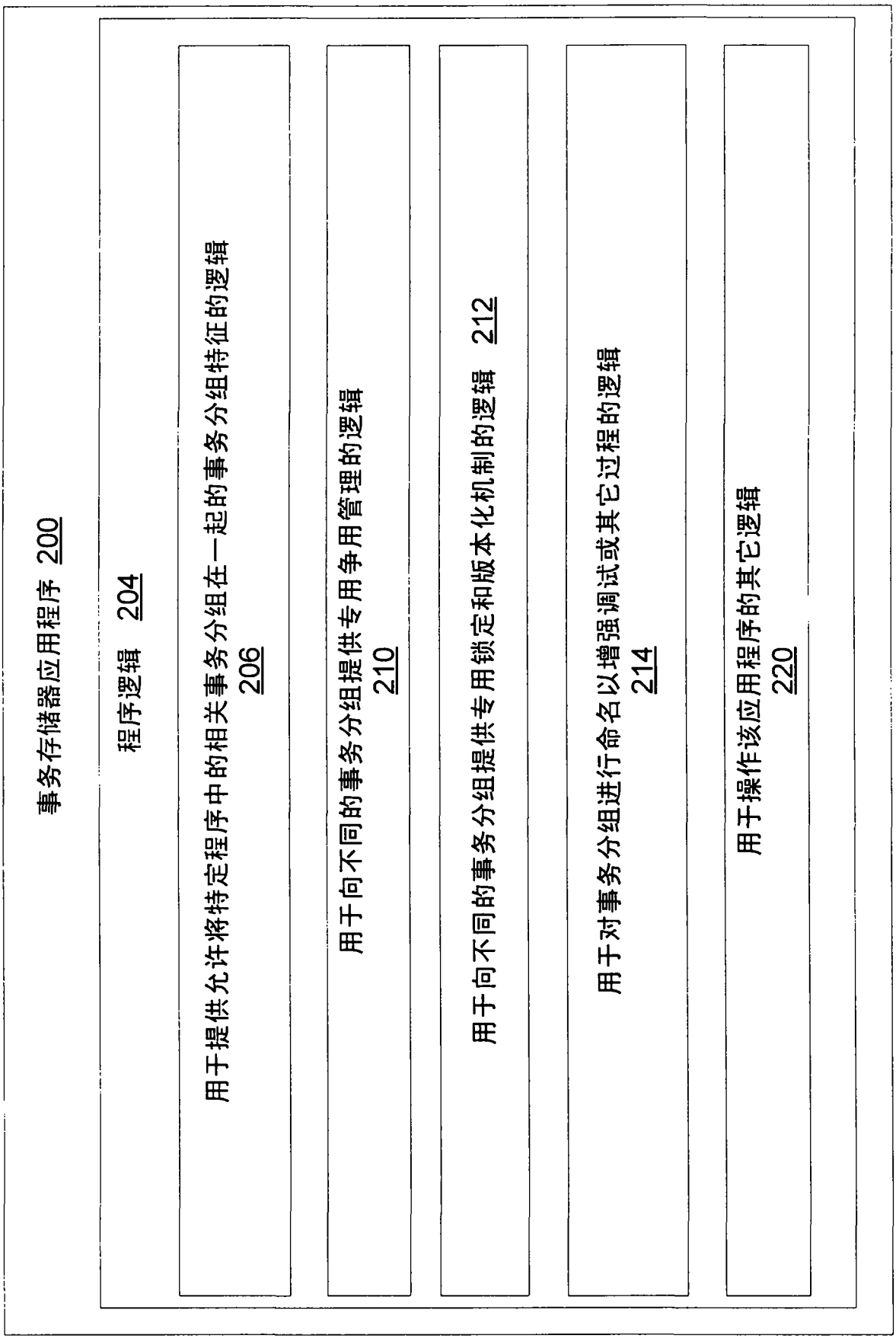

事务存储器应用程序 200
程序逻辑 204
用于提供允许将特定程序中的相关事务分组在一起的事务分组特征的逻辑 206
用于向不同的事务分组提供专用争用管理的逻辑 210
用于向不同的事务分组提供专用锁定和版本化机制的逻辑 212
用于对事务分组进行命名以增强调试或其它过程的逻辑 214
用于操作该应用程序的其它逻辑 220

图 2

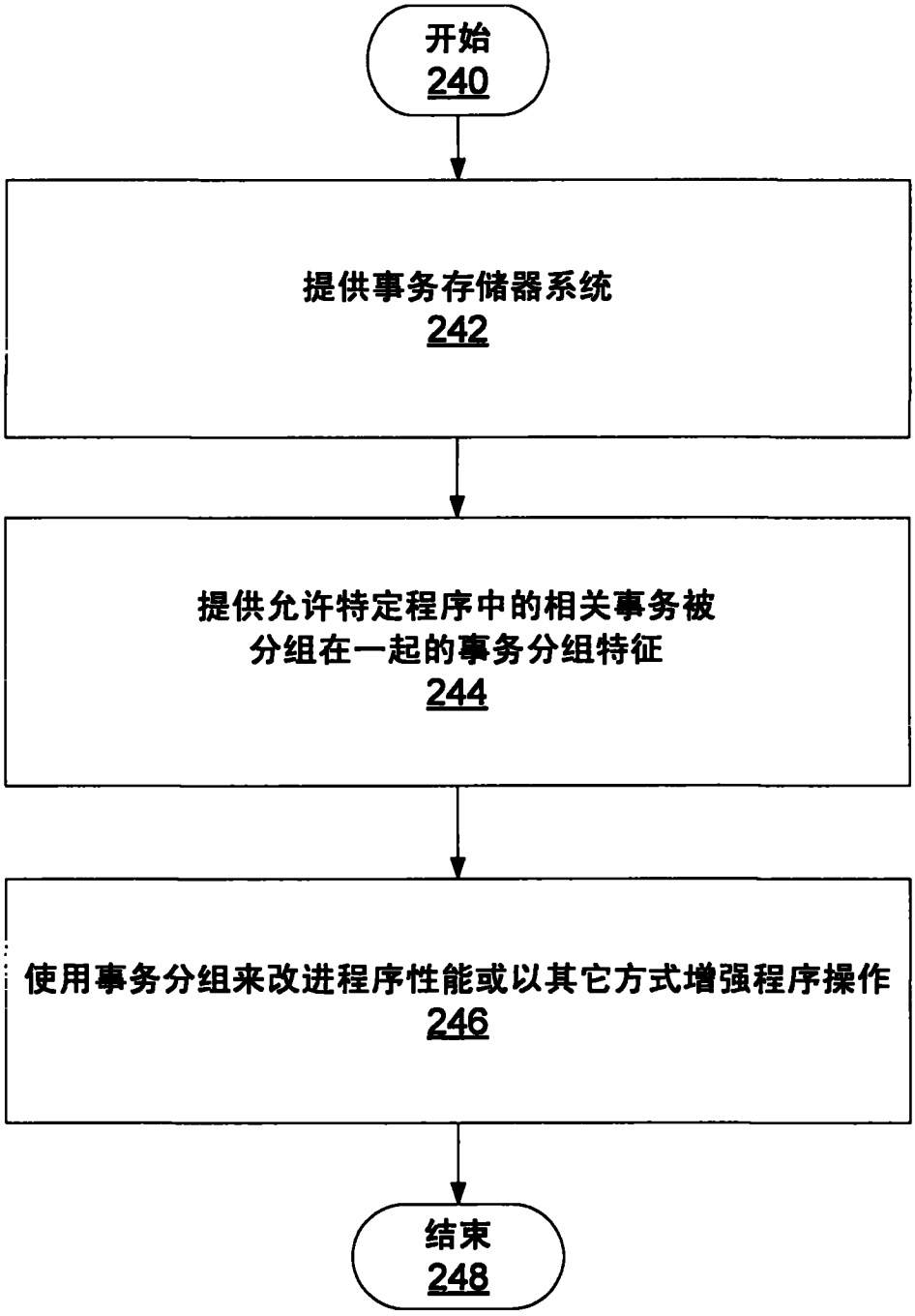

开始
240
提供事务存储器系统
242
提供允许特定程序中的相关事务被分组在一起的事务分组特征
244
使用事务分组来改进程序性能或以其它方式增强程序操作
246
结束
248

图 3

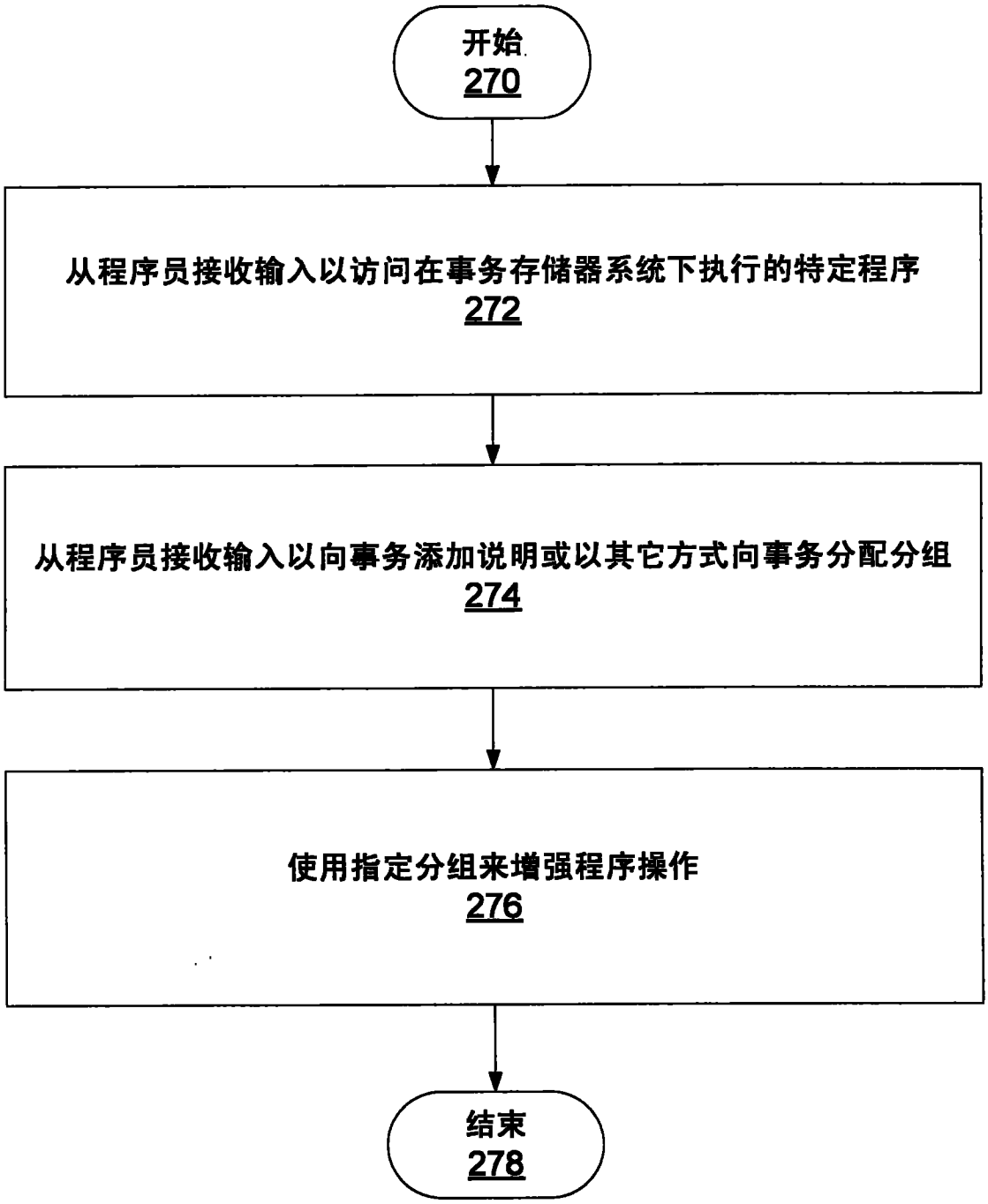

开始
270
从程序员接收输入以访问在事务存储器系统下执行的特定程序
272
从程序员接收输入以向事务添加说明或以其它方式向事务分配分组
274
使用指定分组来增强程序操作
276
结束
278

图 4

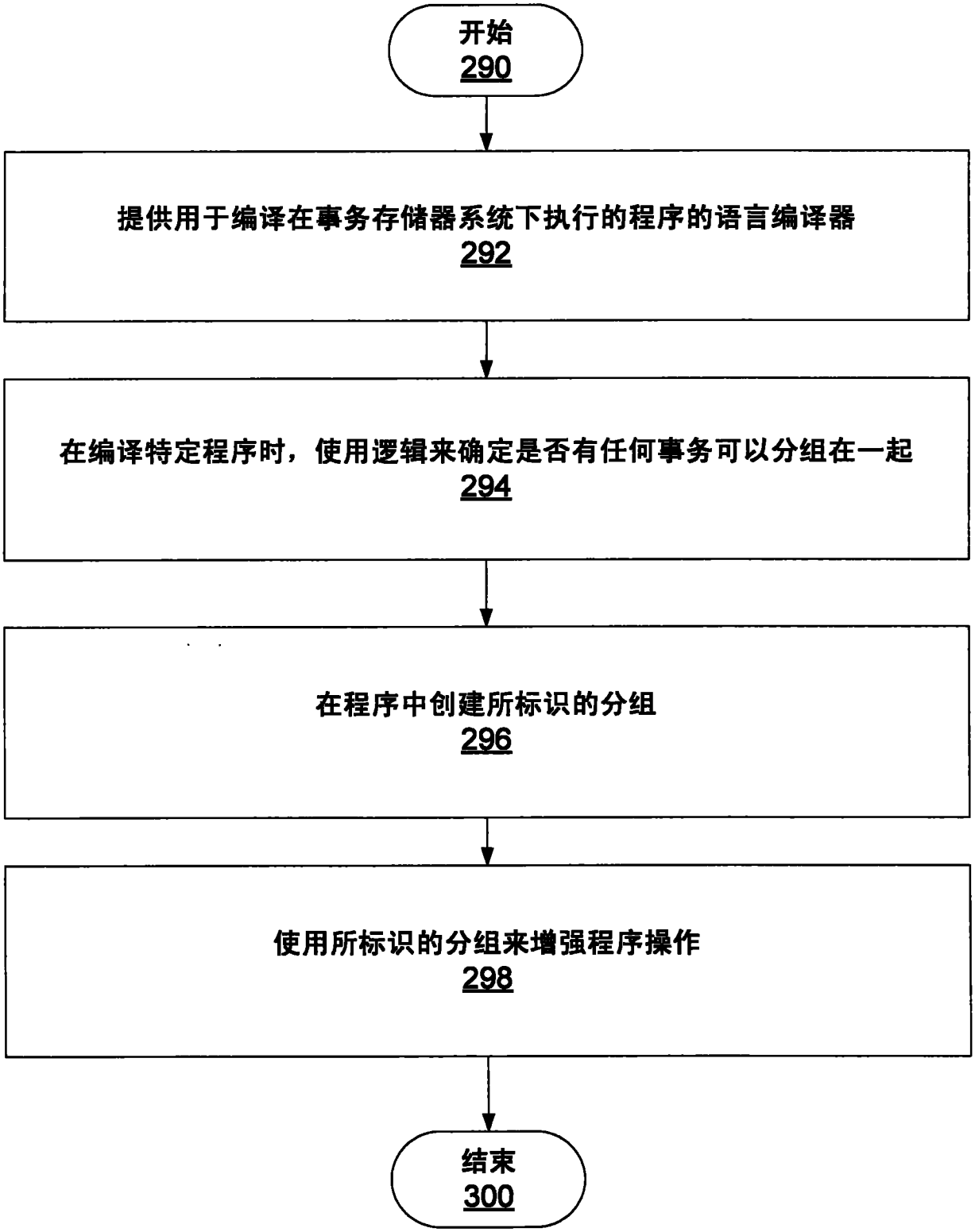

开始
290
提供用于编译在事务存储器系统下执行的程序的语言编译器
292
在编译特定程序时，使用逻辑来确定是否有任何事务可以分组在一起
294
在程序中创建所标识的分组
296
使用所标识的分组来增强程序操作
298
结束
300

图 5

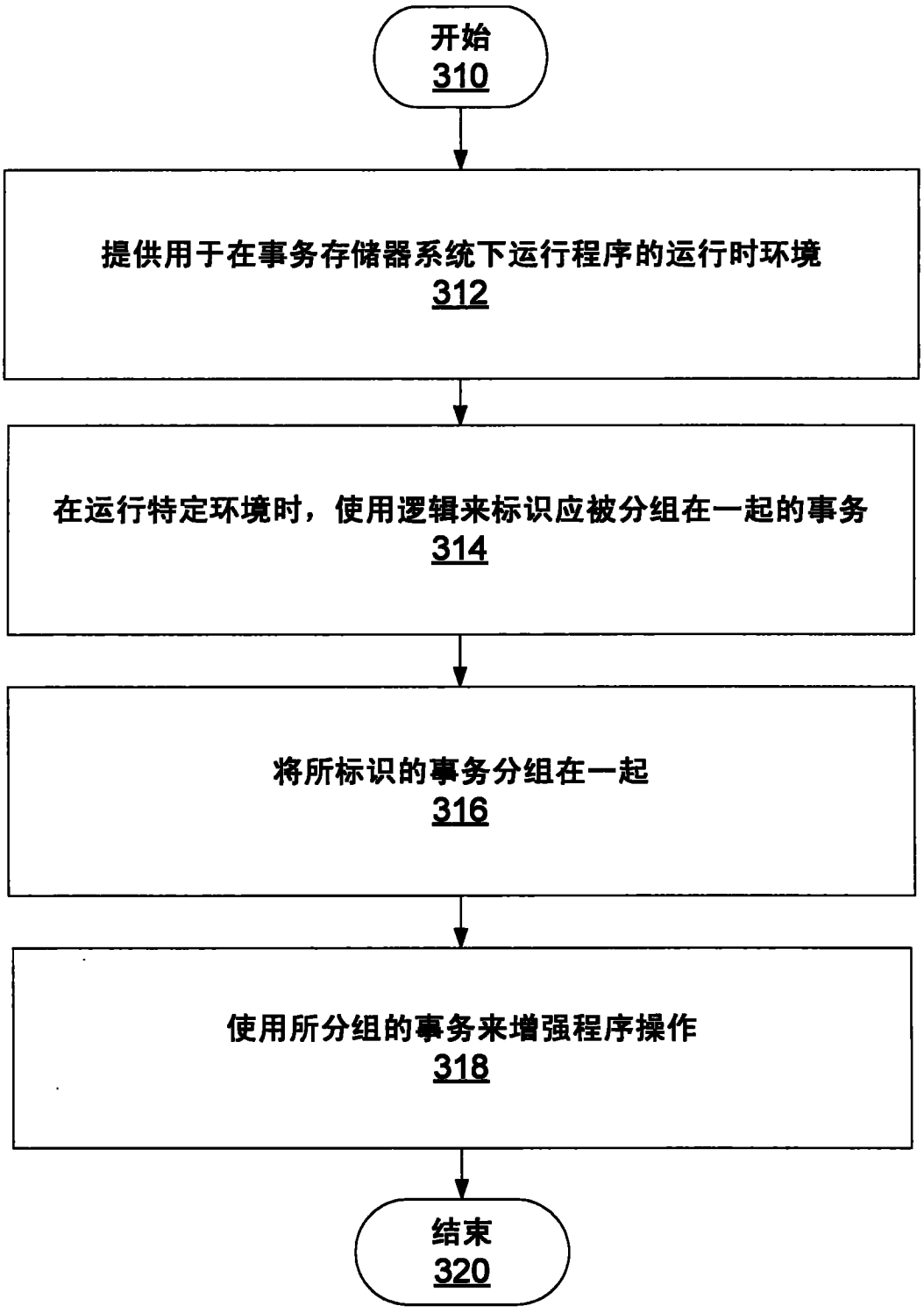

开始
310
提供用于在事务存储器系统下运行程序的运行时环境
312
在运行特定环境时，使用逻辑来标识应被分组在一起的事务
314
将所标识的事务分组在一起
316
使用所分组的事务来增强程序操作
318
结束
320

图 6

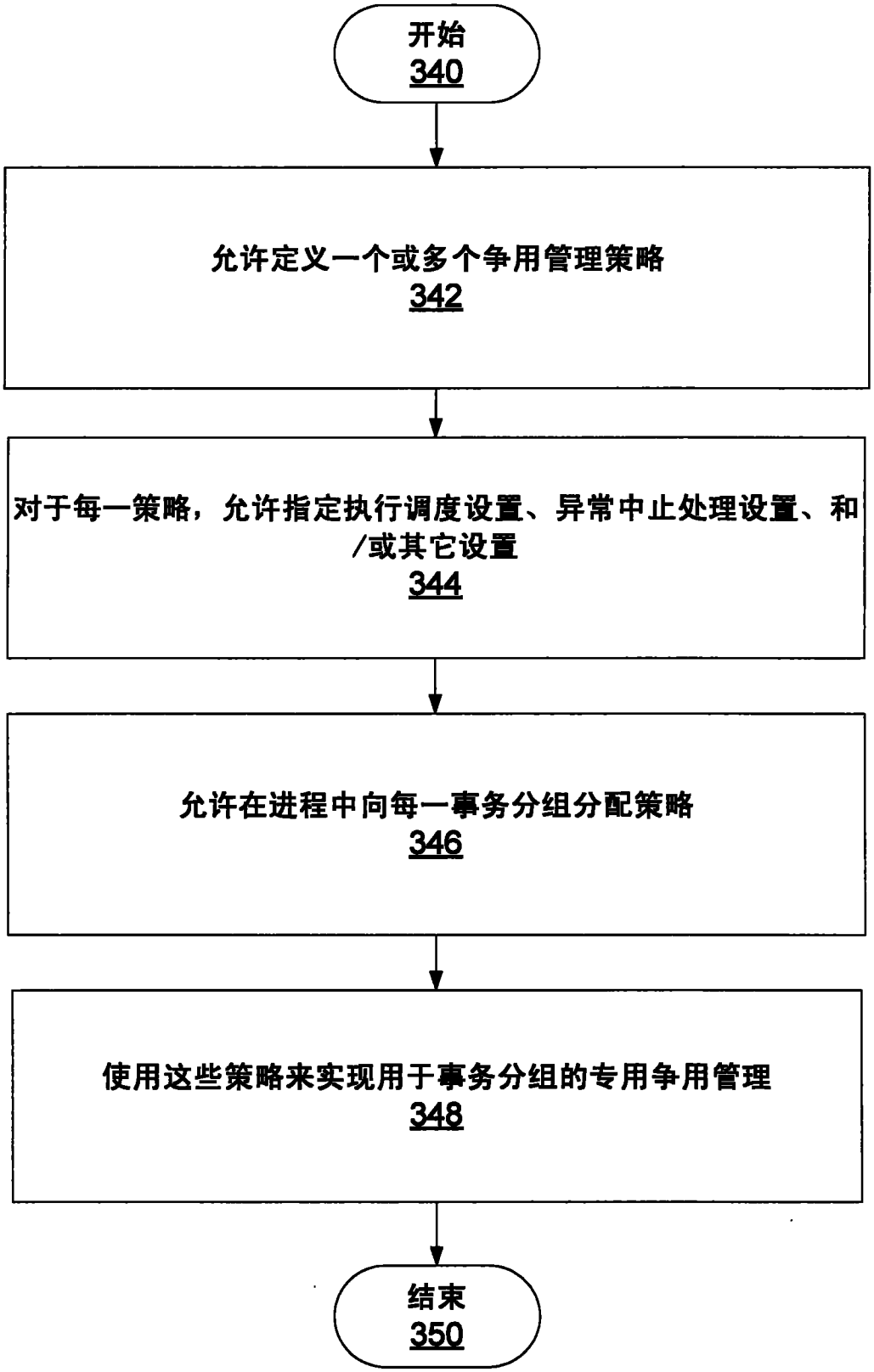

开始
340
允许定义一个或多个争用管理策略
342
对于每一策略，允许指定执行调度设置、异常中止处理设置、和/或其它设置
344
允许在进程中向每一事务分组分配策略
346
使用这些策略来实现用于事务分组的专用争用管理
348
结束
350

图 7

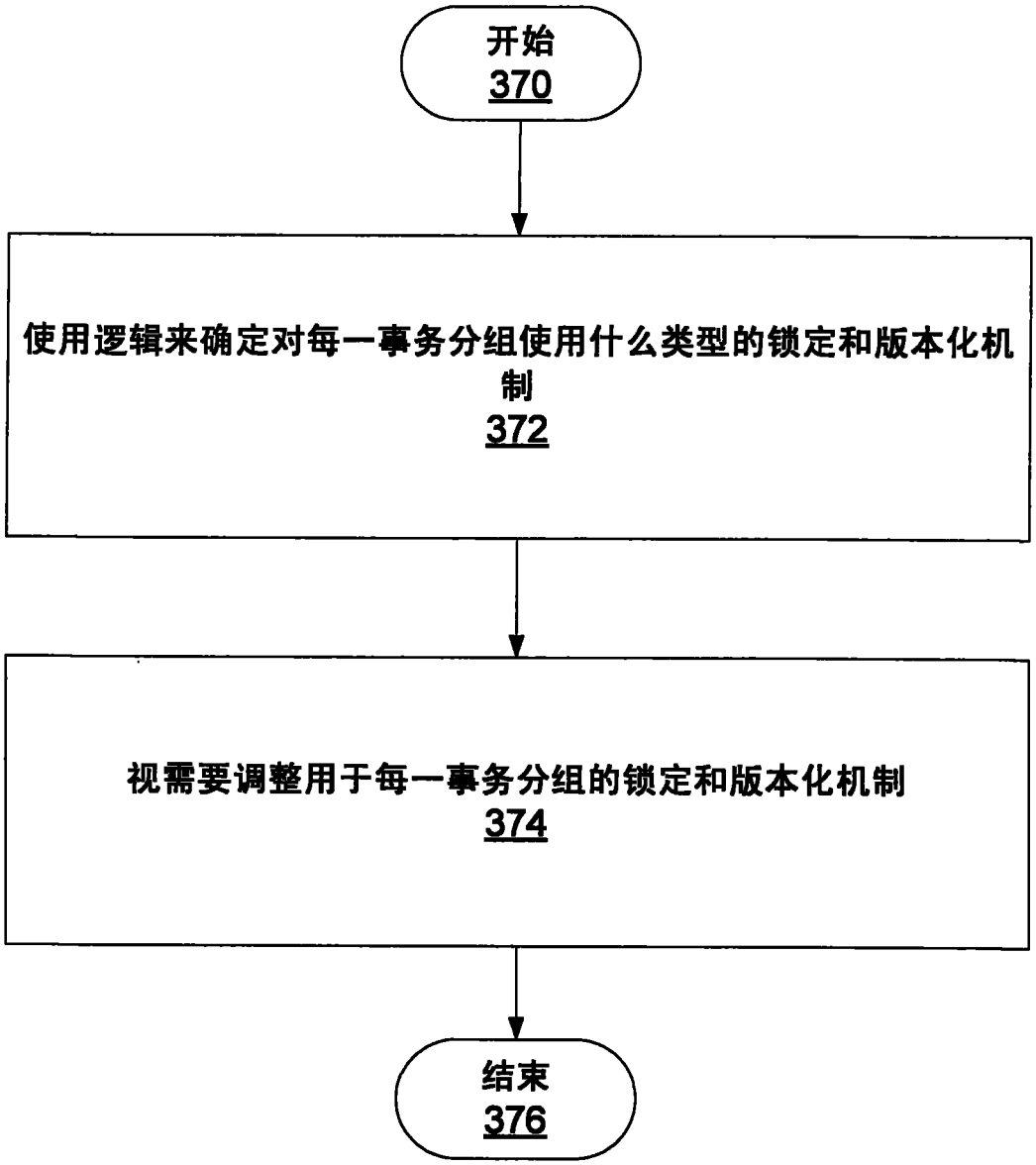

开始
370
使用逻辑来确定对每一事务分组使用什么类型的锁定和版本化机制
372
视需要调整用于每一事务分组的锁定和版本化机制
374
结束
376

图 8

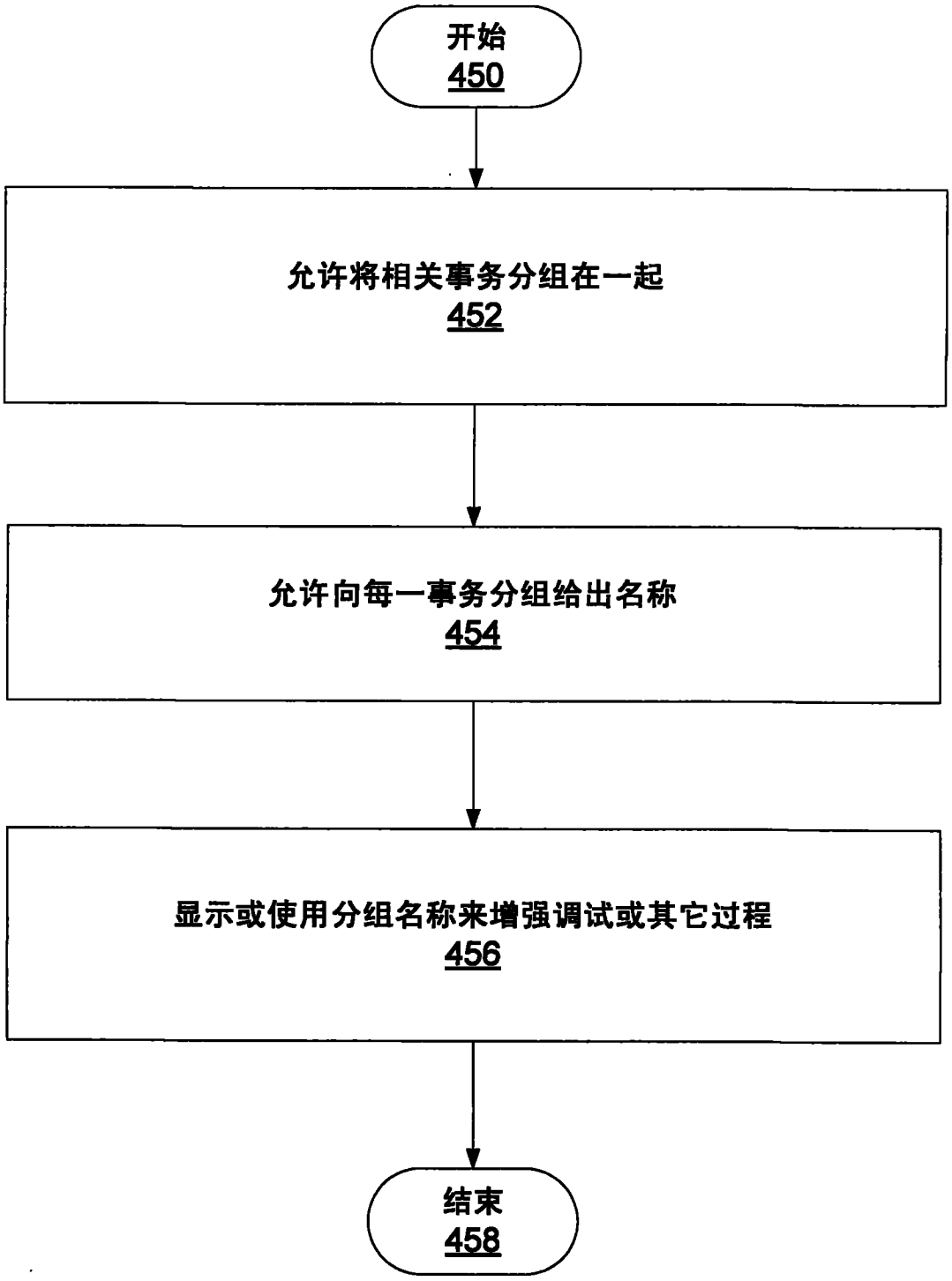

开始
450
允许将相关事务分组在一起
452
允许向每一事务分组给出名称
454
显示或使用分组名称来增强调试或其它过程
456
结束
458

图 9

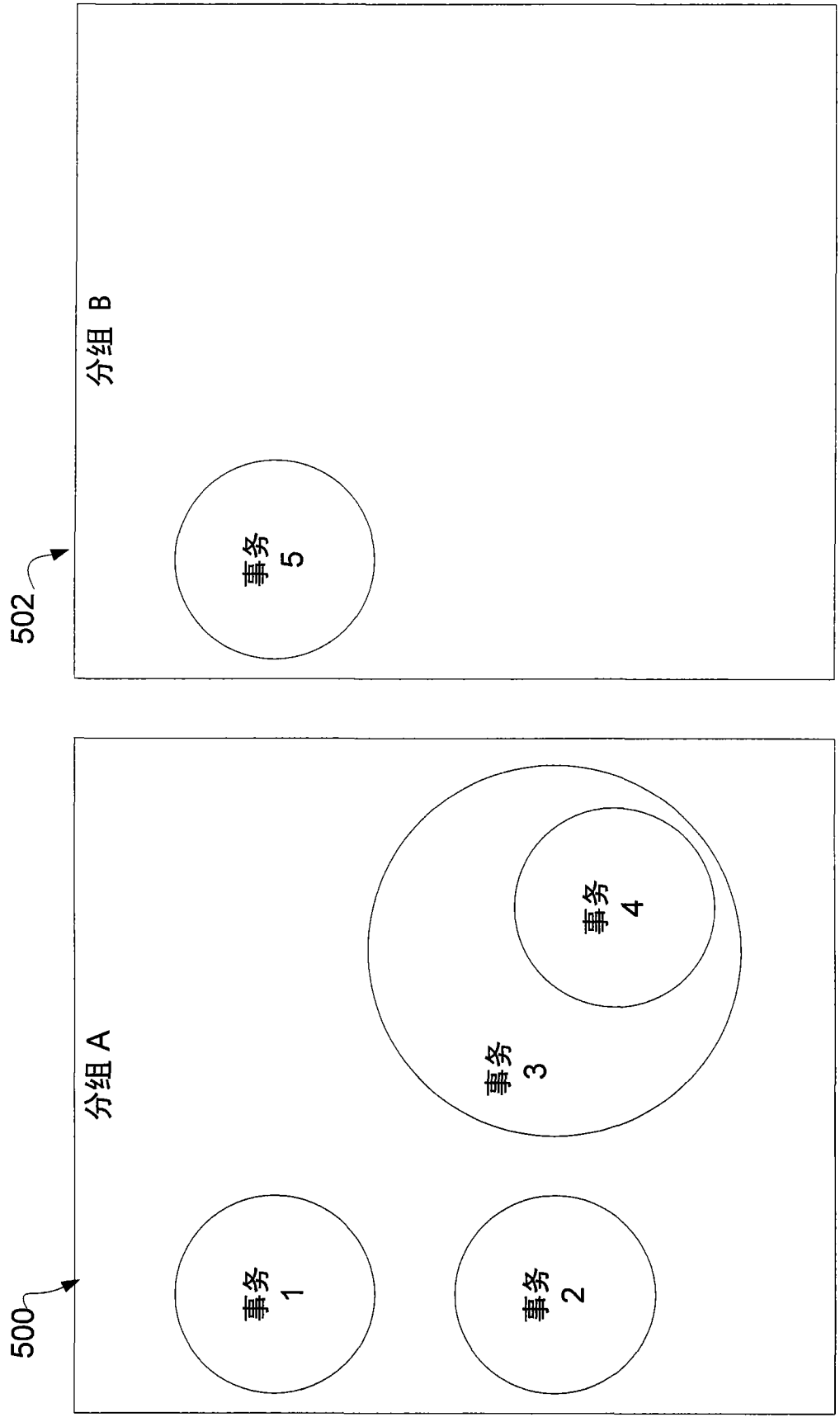


图 10