(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102419764 A
(43)申请公布日 2012.04.18

(21)申请号 201110339865.7

(22)申请日 2011.10.19

(30)优先权数据
12/908,749 2010.10.20 US

(71)申请人 微软公司
地址 美国华盛顿州

(72)发明人 P-A·拉森 M·茨维林
C·迪亚科努

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 杨洁

(51)Int.Cl.
G06F 17/30(2006.01)

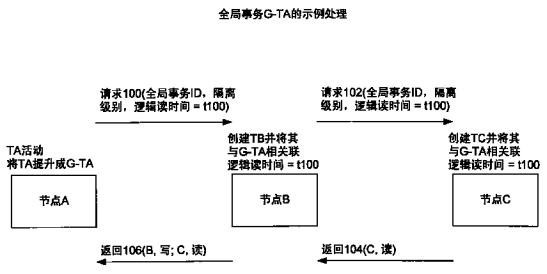
权利要求书 2 页 说明书 15 页 附图 9 页

(54)发明名称

带有多版本化的数据库系统的分布式事务管理

(57)摘要

本发明涉及带有多版本化的数据库系统的分布式事务管理。本发明涉及确保全局或分布式数据库事务中参与节点之间的同步的分布式事务管理技术。该技术利用使用各个参与节点处的本地时钟的提交协议。全局事务中的参与者被配置为利用相同的提交时间戳和逻辑读时间，并且可以提前它们各自的本地时钟以建立该同步。在一个实施例中，分布式提交利用二阶段提交的修改版本，该二阶段提交的修改版本包括从参与者收集提交时间戳投票的额外阶段。另外，心跳机制可用于建立节点之间的松散同步。在另一个实施例中，除了事务结果本身，节点通过返回生成事务结果时所涉及的节点的列表以及由这些节点所使用的访问类型来响应远程事务请求。



CN 102419764 A

1. 一种分布式数据库事务管理系统，包括：

分布式事务管理组件 410，被配置为通过便于对由多个分布式数据库节点 A、B、C 在提交事务时所使用的提交时间戳的同步来实施对所述多个分布式数据库节点 A、B、C 上的数据进行操作的所述事务的原子性；

其中，所述提交时间戳是在未参考对所述多个分布式数据库节点 A、B、C 全局可用的全局时钟的情况下同步的。

2. 如权利要求 1 所述的系统，其特征在于，所述分布式事务管理组件 410 还被配置为对由跨所述多个分布式数据库节点 A、B、C 的所述事务所使用的逻辑读时间进行同步。

3. 如权利要求 1 所述的系统，其特征在于，所述分布式事务管理组件 410 还被配置为传递包括与系统时间有关的数据的心跳消息，并且其中所述心跳消息至少部分地通过依照所述系统时间触发对各个本地时钟的提前来便于所述多个分布式数据库节点 A、B、C 处的各个本地时钟之间的同步。

4. 如权利要求 1 所述的系统，其特征在于，所述多个分布式数据库节点 A、B、C 中的至少一个分布式数据库节点 A 作为所述事务的协调器来操作并且包括所述分布式事务管理组件 410，所述分布式事务管理组件 410 被配置为在每事务的基础上操作，并且多个分布式数据库节点 A、B、C 被配置为经由各个分布式事务管理组件 410 同时地管理各个事务。

5. 如权利要求 1 所述的系统，其特征在于，所述多个分布式数据库节点 A、B、C 中的至少一个分布式数据库节点 A 被配置为经由所述分布式事务管理组件 410 来管理与所述多个分布式数据库节点 A、B、C 相关联的一组事务。

6. 如权利要求 1 所述的系统，其特征在于，所述分布式事务管理组件 410 包括提交时间戳同步器组件，所述提交时间戳同步器组件被配置为从所述多个分布式数据库节点 A、B、C 收集提交时间投票以至少部分地基于所述提交时间投票来便于由所述多个分布式数据库节点 A、B、C 在提交所述事务时所使用的所述提交时间戳的同步，并且便于依照所述提交时间戳来提前分别与所述多个分布式数据库节点 A、B、C 中的至少一个分布式数据库节点 A 相关联的本地时钟。

7. 如权利要求 1 所述的系统，其特征在于，至少部分地基于二阶段提交过程在所述多个分布式数据库节点 A、B、C 提交所述事务。

8. 如权利要求 1 所述的系统，其特征在于，所述分布式事务管理组件 410 与全局事务协调器相关联。

9. 如权利要求 8 所述的系统，其特征在于，所述全局事务协调器管理与所述多个分布式数据库节点 A、B、C 相关联的全部事务。

10. 如权利要求 8 所述的系统，其特征在于，所述全局事务协调器被配置为将系统时间信息合并到传递给所述多个分布式数据库节点 A、B、C 的各个分布式数据库节点的各个消息中，并且其中所述系统时间信息便于与所述多个分布式数据库节点 A、B、C 的所述各个分布式数据库节点相关联的本地时钟的同步。

11. 一种用于管理分布式数据库事务的方法，包括：

从与所述分布式数据库事务相关联的事务协调器接收关于所述分布式数据库事务的信息 700；

至少部分地基于本地时钟来确定所述分布式数据库事务的提交时间投票 710；

将所述分布式数据库事务的所述提交时间投票传递给所述事务协调器 730 ；

从所述事务协调器接收全局提交时间戳 740 ；以及

将所述分布式数据库事务的提交同步到所述全局提交时间戳 750。

12. 如权利要求 11 所述的方法，其特征在于，还包括 ：

如果所述全局提交时间戳指示了比与所述本地时钟相关联的时间更早的时间，则将对所述全局事务的重试请求发送到所述事务协调器。

13. 如权利要求 11 所述的方法，其特征在于，还包括 ：

标识从所述事务协调器接收的一个或多个消息内的系统时间信息 ；以及

根据所述系统时间信息提前所述本地时钟。

14. 如权利要求 11 所述的方法，其特征在于，还包括 ：

从所述事务协调器接收最早活动事务的起始时间戳 ；以及

配置垃圾收集，以使得所述垃圾收集不会进行至超过所述最早活动事务的所述起始时间戳。

15. 如权利要求 11 所述的方法，其特征在于，还包括 ：

从与所述分布式数据库事务相对应的请求数据库节点接收远程请求 ；

创建与所述分布式数据库事务相对应的本地事务 ；

编译生成所述本地事务的结果时所涉及的数据库节点的列表以及由数据库节点的所述列表中的各个数据库节点所使用的访问类型 ；以及

响应于所述远程请求，将数据库节点的所述列表以及所述访问类型返回到所述请求数据库节点。

带有多版本化的数据库系统的分布式事务管理

技术领域

[0001] 本发明涉及数据库系统中的事务管理，并且更具体地，涉及对使用多版本化的数据库系统中的分布式事务的处理。

背景技术

[0002] 数据库系统可以实现各种机制以便确保在数据库系统上执行的事务产生正确的结果。具体而言，数据库系统可以实现并发控制机制以相互隔离多个并发执行的事务，和/或防止这些事务彼此干扰。然而，尽管并发控制机制对正确性是有用的，但它们增加了开销并且负面地影响了系统性能（例如，吞吐量和响应时间）。一类并发控制机制被设计用于带有多版本化的系统，即，数据库系统可以存储一记录的多个版本，不同的版本具有不重叠的有效时间间隔。

[0003] 数据库系统可以是分布式数据库系统，其中数据库分布于多个不同的计算机或节点。在分布式数据库中，某些事务可以是本地或全局的。本地事务限于单个节点，而全局事务涉及多个节点。用户通常不知道事务是本地的还是全局的，因为事务是由数据库系统“在幕后”处理的。因此，期望分布式数据库系统实现完整、高效并且可伸缩的用于处理本地和全局事务两者的机制。例如，由分布式数据库系统执行的本地事务应当产生最小的开销，或者相比于与在本地数据库系统中执行的事务相关联的开销，不会产生附加的开销。作为另一个示例，所支持的用于本地事务的基本上全部隔离级别也应当被支持来用于全局事务。作为又一个示例，由分布式数据库系统执行的事务不应需要被频繁地访问的全局节点，并且特别是全局时钟，以避免与使用全局节点和/或时钟相关联的性能和可伸缩性损失。作为又一个示例，由分布式数据库系统执行的事务不应导致分布式死锁，因为检测这样的死锁是困难并且昂贵的。

[0004] 常规上，已经提出了用于分布式数据库系统中的分布式事务管理的各种解决方案。然而，所提出的用于带有多版本化的分布式数据库的解决方案中没有一个能解决全部上述需要，并且期望这样的解决方案显著地改善分布式数据库系统的性能。

[0005] 当今分布式事务管理技术的上述缺点仅旨在提供常规系统的某些问题的概览，并且不旨在是穷尽性的。常规系统的其他问题以及此处描述的各非限制性实施例的对应益处可以在审阅以下描述后变得更显而易见。

发明内容

[0006] 此处提供了简化的发明内容以帮助能够对以下更详细的描述和附图中的示例性、非限制性实施例的各方面有基本或大体的理解。然而，本发明内容并不旨在作为详尽的或穷尽的概观。相反，本发明内容的唯一目的是以简化的形式来提出与一些示例性非限制性实施例相关的一些概念，作为以下各实施例的更为详细的描述的序言。

[0007] 此处提供的各实施例针对实现多版本化的数据库系统，例如，其中数据库可包含同一记录的带有不重叠的有效时间间隔的多个版本。事务指定其逻辑读时间，该逻辑读时

间确定对该事务可见的记录的版本。读时间和有效时间间隔基于实现为例如单调递增计数器的逻辑时钟。

[0008] 此外，此处的实施例可利于带有多版本化的数据库系统的一致性和准确性。例如，令 T1 是在若干节点具有已修改数据的全局事务。在带有多版本化并且没有全局时钟的分布式系统中，在保证另一个全局事务 T2 获得在事务上一致的视图时，例如 T2 或者随处看到 T1 的更新或者看不到 T1 的更新，产生问题。

[0009] 因此，如此处各实施例中描述的数据库系统中使用的提交协议确保在事务 T1 接触的全部节点上使用相同的提交时间戳。因此，假如随后的事务 T2 在全部节点上使用相同的逻辑读时间，那么 T2 可被配置为看到由 T1 生成的更新，在系统内要么随处可见要么看不到。

[0010] 在其他实施例中，采用了各种优化和 / 或其他技术。这些包括，例如，(1) 使用通过批处理与参与者的通信来减少提交处理的开销的全局事务协调器，(2) 通过将信息承载到由全局事务协调器发送的消息上来保持本地时钟基本同步，以及 (3) 通过将信息承载到由全局事务协调器发送的消息上来防止节点处过早的垃圾收集。

[0011] 在一个或多个实施例中，分布式事务管理是通过便于与全局事务的参与者相关联的本地时钟之间的同步来实现的。在一个非限制性实现中，参与全局事务的各个数据库节点就将要用于该事务的提交时间戳达成协定。这样的协定是单独地或在事务协调器的辅助下达成的。此外，参与全局事务的一个或多个节点可以根据达成协定的提交时间戳提前其各自的本地时钟。

[0012] 在一实施例中，提交分布式事务是经由二阶段提交 (2PC) 的修改版本来实现的，在该二阶段提交的修改版本中，数据库系统中的此处被称为事务协调器的指定节点从参与事务的节点收集提交时间戳投票。提交时间戳投票在给定的参与节点处基于该参与节点的本地时钟而生成。随后确定全局提交时间戳，例如基于所收集的投票来计算，并且全局提交时间戳随后被提供给参与节点以用于对应的分布式提交。为了符合全局提交时间戳，一个或多个参与节点可以提前它们的本地时钟。

[0013] 此处描述的系统和方法还包括分布式数据库系统中的节点可以独立于全局事务同步其本地时钟的规定。作为非限制性的示例，指定的数据库节点便于传递“心跳”信息。基于该信息或任何其他合适的信息，各个数据库节点调整其本地时钟。以规则或不规则的间隔传送心跳信息或其他同步信息，从而确保各个数据库节点处的本地时钟保持基本同步。

[0014] 此处描述的其他系统和方法包括数据库节点返回对来自给定节点的请求的响应以及生成该请求的结果所涉及的节点的身份的机制。以此方式，可使全局事务的根节点知道该事务所涉及的节点，而无需来自全局实体的干预。此外，由生成事务的结果所涉及的各个节点执行的访问类型（例如读或写）同该各个节点的身份一起被返回，从而利于作出请求的节点处的更大深度的信息。

[0015] 这些和其他实施例在下面将更详细地描述。

附图说明

[0016] 各非限制性实施例参考附图来进一步描述，附图中：

[0017] 图 1 是示例性全局事务处理的说明性概览；

[0018] 图 2 是分布式数据库系统中示例性事务启动的说明性概览；

[0019] 图 3 是分布式数据库系统中全局事务的示例性协调的说明性概览；

[0020] 图 4 是示出根据一个或多个实施例的分布式事务管理机制的示例性功能的框图；

[0021] 图 5 是示出用于分布式数据库系统中事务管理和节点同步的示例性、非限制性过程的流程图；

[0022] 图 6 是示出用于协调数据库系统中关于分布式事务的一组节点的示例性、非限制性过程的另一个流程图；

[0023] 图 7 是示出在分布式事务的上下文中便于与数据库系统的协调和同步的示例性、非限制性过程的又一个流程图；

[0024] 图 8 是表示其中可实现此处所描述的各实施例的示例性、非限制性联网环境的框图；以及

[0025] 图 9 是表示其中可实现此处所描述的各实施例的一个或多个方面的示例性、非限制性计算系统或操作环境的框图。

具体实施方式

[0026] 概览

[0027] 作为介绍，数据库系统可以通过诸如但不限于数据创建和维护、信息检索、经由搜索或其他机制的数据访问等操作来便于对各个相关联的数据项的管理。在某些实施例中，数据库系统可以提供各种机制以确保相关联的数据的可恢复性和完整性。例如，数据库系统可以管理在系统内执行的（例如，与由数据库系统执行的各个事务相关联的）操作从而确保这样的操作产生正确的结果。作为示例而非限制，数据库系统可以包括并发控制机制以确保由数据库系统并发地执行的各个事务不会彼此干扰。此外，数据库系统可包括各种其他机制以确保正确的操作。作为非限制性的示例，数据库系统可以利用确保与系统相关联的数据的一致性、完整性、和 / 或可恢复性的各种机制。

[0028] 在一个示例中，数据库系统可用于管理存储在至少一个数据存储中的信息。作为示例而非限制，可设计并实现存储器内数据库来管理存储在对应的计算机存储器或任何其他合适的非瞬态计算机存储介质中的数据。存储器内数据库系统的各种非限制性实施例或者可用于提供类似功能的其他数据库系统通常可提供关于相关联的数据存储的低级功能。这样的功能可包括，例如，支持事务和 / 或查找、光标等。此外，这样的数据库系统可以直接地或间接地通过与提供这样的功能的一个或多个不同的系统的合作来提供对各种更高级功能的支持。可由数据库系统直接地或间接地支持的更高级功能的示例包括但不限于，相关联的用户界面的生成或控制、支持基于各种查询语言的查询（例如，美国国家标准协会 (ANSI)/ 国际标准组织 (ISO) 结构化查询语言 (SQL)、Xquery 等）等。

[0029] 某些数据库系统被设计并优化以在单个计算机上使用，例如，以使得与该数据库系统相关联的数据存储限于一个设备。然而，由于本地存储限制或其他问题，将数据库系统限于单个机器限制了数据存储的大小，或限制了系统可处理的总体工作负载。因此，在某些情形中，数据库系统分布在多个计算设备上是合乎需要的。例如，可以观察到，希望实现数据库系统以管理极大量数据的数据库系统的企业用户可能由于每个机器的存储限制、冗余问题、与大型服务器相关联的维护成本的减少等而想要将数据的存储分布在多个机器上。

[0030] 由此，可以管理分布式数据存储组的数据库系统是合乎需要的。然而，为了提供与本地系统可以相比的功能和性能，分布式事务的管理应当满足各种必要条件。作为具体的示例而非限制，这些必要条件可包括以下各项。第一，系统不应导致本地事务（例如，仅接触单个节点的事务）上的附加开销。第二，所支持的用于本地事务的全部隔离级别也应当被支持来用于全局事务，例如接触多个节点的事务。第三，不应存在被频繁地访问的全局节点，并且特别是没有全局时钟。第四，系统应当能够操作而没有分布式死锁。尽管存在各种常规的分布式数据库管理解决方案，但不存在满足以上全部必要条件的解决方案。考虑到现有分布式数据库事务管理系统的这些和其他缺点，本发明呈现旨在获得高性能和低等待时间的分布式事务管理的多个实施例。

[0031] 就此方面，在此处说明的各实施例中，提出了解决上述必要条件中的每一个的分布式事务管理方案，由此能够在比较小和便宜的服务器上支持更大的数据库系统和 / 或分布式数据库系统。此外，此处提供事务管理方案的其他益处。可以理解的是，此处描述的各实施例的益处仅旨在作为经由使用这样的实施例而实现的益处的非限制性示例。此外，此处提供的各个方案不旨在关于所声明的益处中的任一个和 / 或可从此类方案中实现的任何其他益处而被认为优于彼此。

[0032] 在某些实施例中，提交协议可以被分布式事务管理机制所利用，该分布式事务管理机制仅使用本地时钟，例如本地事务事件计数器、或各个数据库节点处的任何其他合适的跟踪度量。本地时钟可以被同步以作为提交全局事务的一部分。作为非限制性的示例，可在提交事务期间提前本地时钟的值以确保该提交中的某些或全部参与者具有相同的提交时间戳。此外，可以理解的是在此处的各个实施例中，时间戳不限于实际的时间。计数器或计时单元计数也可用于表示时间戳，例如，表示进展的任何数据。

[0033] 在其他实施例中，“心跳”机制或其他手段可用于同步节点，从而防止各个数据库节点的本地时钟彼此漂移得过于远离。此处详细地描述了用于以此方式实现同步的技术。

[0034] 此处的各个实施例是基于全局事务行为的各方面。以下是这些方面的具体的、非限制性示例。可以理解的是，此处描述的实施例中的某些可以考虑这些方面中的某些、全部或者一个也不考虑。此外，可以理解的是，此处呈现的各个实施例不旨在限于以下方面中的任一个，除非以其他方式明确声明。因此，作为第一个示例方面，全局事务中的全部参与者使用相同的逻辑读时间。在第二个方面中，当节点返回远程请求的结果时，它还包括生成结果时所涉及的节点的列表以及所列节点上的访问类型（例如读或写）。这使得事务的始发节点或根节点能够跟踪事务的参与者。在第三个方面中，全局事务的参与者使用相同的提交时间戳，并且在某些情形中可以提前它们各自的本地时钟以确保参与者之间的提交时间戳是相同的。在第四个方面中，分布式提交由二阶段提交的修改版本来处理，该二阶段提交的修改版本包括从参与者收集提交时间戳投票的额外阶段。

[0035] 在一个实施例中，分布式数据库事务管理系统包括分布式事务管理组件，该分布式事务管理组件被配置为通过便于由多个分布式数据库节点在提交事务时使用的提交时间戳的同步来实施对多个分布式数据库节点上的数据进行操作的事务的原子性，其中提交时间戳是在未参考对多个分布式数据库节点全局可用的全局时钟的情况下同步的。

[0036] 分布式事务管理组件还可被配置为对由跨多个分布式数据库节点的事务所使用的逻辑读时间进行同步。另外地或另选地，分布式事务管理组件可被配置为传递心跳消息，

该心跳消息包括与系统时间有关的数据。该心跳消息可以至少部分地通过依照系统时间触发对各个本地时钟的提前来便于多个分布式数据库节点处的各个本地时钟之间的同步。

[0037] 在某些实现中，至少一个分布式数据库节点作为给定事务的协调器运行，并且包括分布式事务管理组件。在该实现中，分布式事务管理组件可被配置为在每事务的基础上操作。此外，多个分布式数据库节点可被配置为经由各个分布式事务管理组件来同时地管理各个事务。

[0038] 在其他实现中，多个分布式数据库节点中的至少一个分布式数据库节点被配置为经由分布式事务管理组件来管理与该多个分布式数据库节点相关联的一组事务。

[0039] 此外，分布式事务管理组件可包括提交时间戳同步器组件，该提交时间戳同步器组件被配置为从多个分布式数据库节点收集提交时间投票，并且至少部分地基于提交时间投票来便于对由多个分布式数据库节点在提交事务时使用的提交时间戳的同步。另外地或另选地，提交时间戳同步器组件可被配置为便于依照提交时间戳来提前分别与多个分布式数据库节点中的至少一个分布式数据库节点相关联的本地时钟。

[0040] 此处通常描述的所进行的事务可在多个数据存储处至少部分地基于二阶段提交过程来提交，该二阶段提交过程可以如此处描述的和 / 或以任何其他合适的方式来修改。

[0041] 在其他实现中，分布式事务管理组件与全局事务协调器相关联。在一个示例中，全局事务协调器管理与多个分布式数据库节点相关联的全部事务。在另一个示例中，全局事务协调器可被配置为将系统时间信息合并到被传递给多个分布式数据库节点中的各个分布式数据库节点的各个消息中。在这样的示例中，系统时间信息便于与多个分布式数据库节点中的各个分布式数据库节点相关联的本地时钟的同步。另外地或另选地，全局事务协调器可被配置为将活动事务信息合并到被传递给多个分布式数据库节点中的各个分布式数据库节点的各个消息中。在这样的示例中，活动事务信息便于对由多个分布式数据库节点中的各个分布式数据库节点所执行的垃圾收集的控制。

[0042] 在另一个实施例中，用于管理分布式数据库事务的方法包括接收同对与关联于该事务的各个参与节点相关联的数据存储中的数据进行操作的事务有关的信息；从各个参与节点请求提交时间投票；响应于该请求从各个参与节点接收（例如收集）提交时间投票；至少部分地基于提交时间投票来计算事务的全局提交时间戳；以及将各个参与节点处事务的提交同步到全局提交时间戳。

[0043] 在某些实现中，该方法可包括获得与各个参与节点中的一个或多个节点相对应的本地时钟值，至少部分地基于本地时钟值来生成系统时间信息，以及将系统时间信息传递给各个参与节点。在其他实现中，该方法可包括接收各个参与节点的各个最早活动事务的起始时间戳，从各个参与节点之间标识全局最早活动事务，并且将全局最早活动事务的起始时间戳传递到各个参与节点。

[0044] 在其他实现中，根据该方法处理的事务的提交可以在各个参与节点处根据例如二阶段提交过程、三阶段提交过程或 Paxos 协议来同步。

[0045] 在另一个实施例中，用于管理分布式数据库事务的方法包括从与分布式数据库事务相关联的事务协调器接收关于分布式数据库事务的信息，至少部分地基于本地时钟来确定分布式数据库事务的提交时间投票，将分布式数据库事务的提交时间投票传递给事务协调器，从事务协调器接收全局提交时间戳，以及将对分布式数据库事务的提交同步到全局

提交时间戳。

[0046] 如果以上方法中所描述的全局提交时间戳指示了比与本地时钟相关联的时间更早的时间，则以上方法可包括将对全局事务的重试请求发送到事务协调器。

[0047] 在某些实现中，以上方法还可包括标识从事务协调器接收到的一个或多个消息内的系统时间信息，以及将本地时钟提前至该系统时间信息。

[0048] 在其他实现中，该方法还可包括从事务协调器接收最早活动事务的起始时间戳，以及配置垃圾收集以使得垃圾收集不会进行至超出最早活动事务的起始时间戳。

[0049] 在其他实现中，以上方法可包括从与分布式数据库事务相对应的请求数据库节点接收远程请求，创建与分布式数据库事务相对应的本地事务，编译在生成本地事务的结果时所涉及的数据库节点的列表以及由数据库节点的列表中的各个数据库节点所利用的访问类型，以及响应于远程请求将数据库节点的列表和访问类型返回到请求数据库节点。

[0050] 此处，以上已呈现了用于实现分布式事务管理的实施例中的某些的概览。作为接下来的内容的向导，更详细地描述分布式事务管理的各种示例性、非限制性实施例和特征。然后，为了附加说明给出一些非限制行的实现和示例，接着是可在其中实现这样的实施例和 / 或特征的代表性网络和计算环境。

[0051] 分布式事务管理

[0052] 作为对关于进行正常事务处理的一个或多个非限制性方式的进一步描述，考虑访问三个节点 A、B 和 C 上的数据并且运行在可串行化隔离级别下的示例全局事务，如图 1 概括地示出。事务以节点 A 上的本地事务 TA 开始，并且使用节点 A 的本地时钟来获得起始时间戳，例如 t100。在图 1 所示的示例中，t100 是将由参与到该事务中的所有节点所使用的逻辑读时间。

[0053] 在其处理中的某点处，TA 可能寻求访问节点 B 上的数据。因此，TA 被提升成以 A 为其根节点的全局事务 G-TA。节点 A 随后向节点 B 发送包括全局事务 ID（例如，节点 ID 加上本地事务 ID）、该事务的隔离级别以及将要使用的逻辑读时间 (t100) 的请求 100。作为响应，节点 B 创建本地事务 TB，将其与该全局事务相关联，并且将 TB 的逻辑读时间设置为 t100。根据本领域公知和理解的技术，与节点 B 上的事务相关联的处理可随后继续。

[0054] 在节点 B 寻求访问节点 C 作为其处理的一部分的情形中，节点 B 可向节点 C 发送包括与请求 100 基本类似的信息的请求 102，例如全局事务 ID、隔离级别以及逻辑读时间。作为响应，节点 C 创建本地事务 TC 并如以上概括地描述地继续。

[0055] 当节点 C 将其结果返回给节点 B 时，它可向节点 B 提供返回报告 104，该返回报告还包括指示哪些节点参与计算该结果以及每个节点所利用的访问类型（例如，读或写）的信息。作为非限制性示例，TC 可以进行对 C 上某些数据的只读访问。因此，节点 C 提供返回报告 104(C，读）。此外，在事务 TB 更新 B 上某些数据的情形中，节点 B 可以向节点 A 提供返回报告 106(B，写 ；C，读）。以此方式，可以理解的是，关于哪些节点参与全局事务以及这些节点上的访问类型的信息流回到该事务的根节点。

[0056] 事务的参与者记录哪个本地事务与全局事务相对应，以使得它可以通过其对应的本地事务来便于对全局事务的处理。参与者还记录事务的隔离级别，以使得它可以决定在处理期间记录哪些信息以及当事务终止时，如果有的话，将要执行哪些动作。

[0057] 尽管以上示例涉及带有可串行化隔离的事务，但可以理解的是，也可以支持其他

级别的隔离。关于对其他级别的隔离的支持，存在不同级别的一致性保证，包括脏读 (DR)、已提交读 (RC)、可重复读 (RR)、可串行化隔离以及快照隔离 (SI)，这些可能是或不是合乎需要的，这取决于正为数据进行服务的特定的应用程序。

[0058] 在 DR 隔离级别下，不管记录的最新版本是否已提交，事务 T1 总是读取该版本。如果 T1 读取由事务 T2 创建的随后中止的版本，则 T1 已经看到了逻辑上从未存在的数据。然而，对于涵盖大量数据的某些报告或监控应用程序以及在确切值不如数据中的全局趋势重要的情况下，由这样的读引入的小误差是可以接受的。

[0059] 在 RC 隔离级别下，事务 T1 读取最近提交的版本，而未提交的版本被忽略。这具有以下效果，T1 可以看到在 T1 的生存期期间所提交的来自事务 T2 的更新中的某些，但也会错过 T2 更新中的某些。换言之，T1 不具有数据的事务一致的视图。

[0060] 在 RR 隔离级别下，系统保证事务 T1 被允许提交，只要 T1 读取的全部版本在该事务的结束时仍然有效。

[0061] 可串行化隔离可被视为进一步处理幻影问题的一种形式的可重复读。事务 T1 的读操作的执行在另一个事务 T2 引入并提交实现该读操作的选择准则的各版本时示出了幻影问题。在 T1 的结束处，该版本对 T1 是可见的，但取决于读操作的计时，该版本在正常处理期间可能尚未被读取。由此，事务开始时不存在的数据可能在事务期间出现，并且由此术语“幻影”用于暗示其出现“出乎意料”。因此，SR 不但将保证在事务期间读取的项在事务结束时不会改变，SR 还另外地保证在事务读的范围内不会引入新数据直到事务结束。

[0062] SI 也是另一种形式的隔离。在 SI 下，事务在逻辑上在事务起始时间获取它自己的对该数据的快照，这保证了读操作不会阻塞并且事务具有该数据的一致视图。

[0063] 关于分布式提交处理，根据使用二阶段提交的实施例来执行提交全局事务，该二阶段提交被修改为包括用于确定事务的提交时间戳的一轮附加消息。为了便于解释并且作为特定的、非限制性的示例，在以下讨论中假设未以其他方式参与到事务的节点担任事务协调器 (TxnC)。这仅是可由此处描述的技术所处理的一种使用场景，因为 TxnC 当然可以还是事务中的参与节点。此外，以下讨论利用基本的二阶段提交协议；然而，可以理解的是，可以使用其他合适的分布式提交协议或协议的组合，诸如三阶段提交过程、Paxos 协议、和 / 或任何其他合适的提交过程或协议。

[0064] 在一个示例中，提交处理由事务的根节点 (例如，节点 A) 通过向 TxnC 发送消息来启动。图 2 概括地示出提交处理的示例启动，其中节点 A 向 TxnC 提供事务提交请求消息 200。在事务的启动时向 TxnC 提供的消息包含其事务中涉及的参与者的列表及其各自的访问类型、以及其他可能的信息。

[0065] 在事务提交处理中，事务中的参与节点与 TxnC 可以彼此通信，如图 3 概括地示出。随后是协调器和参与者的示例协议。虽然以下描述忽略了超时，但可以理解的是，以下描述的协议可被修改来处理超时和 / 或任何其他适合的场景。

[0066] 一般而言，分布式或全局事务中的参与节点利用相关联的本地时钟来维护与所维护的记录相关联的时间戳。在多版本化系统中，这些时间戳确立记录的各个版本的有效时间。例如，如果记录 R 在时间 t10 被创建并且在 t25 被更新，则记录 R 的一个版本的生存期是从 t10 到 t25。因此，保持各个节点的本地时钟之间的同步从而防止与访问在一个或多个节点处的记录的不正确版本相关联的差错是合乎需要的。例如，如果节点 A 的本地时钟与

节点 B 的本地时钟不同步，则对节点 A 和节点 B 的记录进行操作的各个事务可能在节点 A 和节点 B 的记录之间遇到不一致，这进而可导致事务产生不正确的结果。因此，此处描述的协议提供对与事务的参与节点相关联的本地时钟进行同步的机制，从而减轻此类差错。稍后提供关于时间戳和记录版本化的其他细节。

[0067] 现在参考示例协议，可以理解的是，以下协议旨在作为非限制性的示例，并且各个实施例不旨在限于任何特定的协议，除非以其他方式明确地声明。

[0068] 首先参考事务协调器协议，所述协议经由以下步骤来进行。

[0069] (1)TxnC 向参与者发送预提交请求，请求它们对事务的提交时间的投票。

[0070] (2)TxnC 从参与者收集提交时间投票。在某些情形中，参与者可以在该阶段投票以中止事务。在此情形中，TxnC 选择中止事务并进行至下面的步骤 (8)。否则，TxnC 将全局提交时间计算为投票的最大值，在某些情形中添加增量值。

[0071] (3)TxnC 将准备请求发送到参与者，并且将全局提交时间包括在该消息中。

[0072] (4)TxnC 从参与者收集投票（例如提交、重试或中止）。

[0073] (5) 如果参与者中的任一个投票以中止事务，则 TxnC 选择中止并继续至下面的步骤 (8)。

[0074] (6) 如果参与者中的任一个投票以重试事务，则 TxnC 返回到上面的步骤 (1) 并且再次尝试提交。在一个实施例中，重试的次数可被限制为最大数目。

[0075] (7) 由于基于上面的步骤 (5) 和 (6) 全部参与者已经投票以提交事务，因此 TxnC 选择提交。

[0076] (8)TxnC 方将其决定写入日志 300（例如，维护为持久存储的日志）。

[0077] (9) 基于 TxnC 中止或提交事务的决定，TxnC 将中止或提交请求发送到参与者。

[0078] (10)TxnC 从参与者收集返回消息。

[0079] (11)TxnC 将事务结束记录写入其日志 300，并且停止对事务的参与。

[0080] 接下来参考参与者协议，所述协议经由以下步骤来进行。

[0081] (1) 参与者 P 从 TxnC 接收对全局事务 GT 的预提交请求。

[0082] (2) 如果 P 处对应的本地事务 LT 已被中止，则 P 将中止投票发送到 TxnC。

[0083] (3) 否则，P 计算其对提交时间的投票，例如本地时钟值加上增量值，并且将其投票发送到 TxnC。在一个实施例中，可选择增量来为本地事务活动留出足够的时间直到 GT 预提交。

[0084] (4)P 从 TxnC 接收中止请求或带有提交时间戳的准备请求。

[0085] (5) 如果请求是中止，则 P 中止 LT 并且继续至下面的步骤 (11)。

[0086] (6) 如果请求是准备，则 P 检查事务提交时间戳。如果提交时间戳少于 P 的本地时钟，则 P 无法提交事务并且将重试投票发送到 TxnC.

[0087] (7) 否则，P 将其本地时钟提前至提交时间戳后的下一个时钟值，将 LT 的结束时间戳设置为提交时间戳，并且根据系统使用的用于本地事务的规则来确定 LT 是否可以被提交。如果 LT 无法被提交，则 P 中止 LT。

[0088] (8) 如果 LT 准备提交并且这是提交 GT 的第一次尝试，则 P 方将 LT 创建的全部新版本和准备记录写入日志 300（例如，维护为持久存储的日志），并且将提交投票发送到 TxnC。否则，P 发送中止投票。

[0089] (9)P 从 TxnC 接收提交或中止请求。

[0090] (10) 如果请求是提交请求，则 P 提交 LT，并且如果 LT 是读 - 写事务，则 P 将提交记录写入日志 300。在一个实施例中，写不必是强制的。

[0091] (11) 如果请求是中止请求，则 P 中止 LT，并且如果 LT 是读 - 写事务，则 P 任选地将中止记录写入日志 300。在一个实施例中，写不必是强制的。

[0092] (12)P 向 TxnC 发送指示事务已被如上决定地提交或中止的消息。

[0093] 如以上示例协议所示，在提交第一次尝试就成功的情形中，事务协调器将三个消息（例如预提交、准备和提交 / 中止）发送到参与者，并且执行两个日志写，其中一个是强制的。参与者也发送三个消息（提交时间投票、提交 / 中止投票、和已提交 / 已中止），并且在本地事务是读 - 写事务的情形中执行两个日志写，其中一个是强制的。另选地，如果本地事务是只读，则可以不执行日志写。

[0094] 以上提交协议确保了如果给定的隔离级别这样要求，则无论是本地事务还是全局事务都可获得数据库的事务一致的视图。换言之，事务要么将看到另一个事务的更新的全部，要么看不到另一个事务的更新。

[0095] 作为具体的、非限制性的示例，令 T1 为修改节点 A、B 和 C 上的数据并且在时间 t100 提交的全局事务。当第二个事务 T2 开始时，不清楚 T2 是本地事务还是全局事务。作为又一个示例，T2 可以始于节点 D 并且使用本地时钟来获得起始时间戳 t120。T2 使用该值作为其在所有节点上的读时间戳，这确保了它具有跨节点的事务一致的视图。由此，如果 T2 从节点 A 和 B 读取，则这两个节点上的 T1 的更新对 T2 都将是可见的。另一方面，可以理解的是，如果 T2 的起始时间戳早于 t100（例如 t80），则 T2 在任一节点上都将看不到 T1 的更新。

[0096] 注意到，如果 T2 在 T1 提交之后始于节点 A、B 或 C 中之一，则保证 T2 能够看到 T1 的改变，因为它的起始时间戳高于 T1 的提交时间戳。然而，如果 T2 始于某些不相关的节点则不能作出保证，因为本地时钟没有被完全地同步。在一实施例中，为了避免时钟漂移得过于远离的情形，可以按规则的间隔运行时钟同步循环。这样的时钟同步循环可以经由传递“心跳”消息和 / 或任何其他合适的机制来执行。以下进一步详细地描述根据各个实施例的可利用的同步过程。

[0097] 可以理解的是，以上方法满足了如上所述的分布式事务管理方案的所需属性。具体而言，本地事务没有开销。此外，支持所有隔离级别。另外，不需要全局时钟或全局协调器；任意节点可以担任事务协调器，并且多个全局提交可以并发地进行。最后，如本领域中公知的，可以使用提交依赖性以避免等待（例如，事务无法完成其准备直到其全部待决的依赖性已被解决时）。由此，不会发生本地死锁或全局死锁。

[0098] 关于对各种隔离级别的处理，可以理解的是，参与节点的提交处理的细节可依赖于所使用的并发控制方法、以来于该隔离级别、以及事务在哪里执行写。例如，如果使用乐观的并发控制方法，则对于已提交读和快照隔离级别，不需要读确认，但由于例如推测性读，本地事务可能具有待决的提交依赖性。在一个示例中，不带写的节点可由事务协调器在投票和提交 / 中止阶段期间忽略但不在准备阶段期间忽略。此外，这样的节点不需要向日志写。因此，可以理解的是，纯只读事务可以仅需要一个阶段，例如，分发提交请求并且收集答复。

[0099] 在某些情形中，降低与此处描述的一个或多个协议相关联的消息收发的开销是合乎需要的。可以理解的是，如果个别地提交每个相关联的事务，则在某些情形中提交协议可展示较高的消息开销。例如，涉及 N 个节点的事务需要 6N 个消息。在一个实施例中，通过利用单个全局协调器、消息汇聚和心跳的技术，可以显著地降低该开销，如此处概括地描述的。

[0100] 在一个实施例中，全局协调器以规则的间隔（例如，每 10ms）将消息广播到所有节点。该消息可包括自从前一消息被发送以来积累的预提交、准备和提交 / 中止请求。当节点接收该消息时，该节点提取对其参与的事务的请求并采取任何适合的动作。参与者可随后将其响应和任何新的提交请求收集到返回消息中，该返回消息在合适的间隔之后、当参与者已经完成各个所需动作时和 / 或任何其他适合的时间被发送回到协调器。当返回消息到达时，协调器处理响应，并且过程继续。一般而言，可以理解的是，以上过程可用于实现全局事务的组提交。例如，以与常规组提交类似的方式，以上过程牺牲了等待时间以利于更低的开销和更高的吞吐量。

[0101] 在另一个实施例中，该心跳机制可用于保持本地时钟松散地同步。当参与者将返回消息发送到协调器时，参与者还可包括其本地时钟的当前值。协调器跟踪其从系统中的各节点接收的最大的时钟值。随后，当协调器广播消息时，它将那些值中的最大值包括在该消息中。因此，参与者在接收消息之后可将其本地时钟提前至新确定的时钟值。通过经由心跳机制和 / 或其他手段加紧时钟同步，降低了用户经历“时间扭曲 (time warping)”的风险。

[0102] 关于不再需要的版本的垃圾收集，可以观察到的是，除非垃圾收集是跨节点同步的，否则由于过于靠前的节点以及事务需要读的垃圾收集版本，全局事务可能必须被中止。例如，一个示例场景中的节点可能已经放弃了在时间 t100 之前期满的全部版本。如果带有小于 t100 的逻辑读时间的全局事务到达该节点，则该事务必须被中止，因为无法保证该事务应当看到的全部版本都是可用的。为了避免这样的问题，可以配置节点上的垃圾收集，以使得它不会进行至超出系统中最早活动事务的起始时间。在一实施例中，事务协调器和 / 或其他数据库实体可以通过将活动事务信息合并到传递给各个数据库节点的消息中来通过以上方式便于对垃圾收集的控制。

[0103] 在一个实施例中，可以通过与时钟同步相同的方式来解决这个问题。例如，参与者可以将节点上最早活动事务的起始时间包括在其返回消息中。事务协调器跟踪每个节点的最新报告的值。当它广播消息时，它将各值中的最小值包括在该消息中。参与者随后使用该值来限制其允许垃圾收集进行的程度。通过使用该方法，可以确保版本不会从全局事务中消失。

[0104] 虽然上面在全局事务协调器的上下文中提供了各个实施例，但是可以理解的是，事务协调可以通过任何合适的方式来实现。例如，作为（例如，管理与系统相关联的全部事务的）全局事务协调器的补充或替换，事务可以由各个数据库节点在每事务的基础上来协调。在这样的示例中，多个数据库节点可以实现此处提供的分布式事务管理技术和组件。此外，某些情形中的各个数据库节点可以同时地或以其他方式重叠时间间隔地操作以便于一次对多个事务的协调。

[0105] 在其他实现中，一个或多个数据库节点可以作为事务协调器操作以管理一组事务

（例如，对应于与系统相关联的全部事务的子集）。在以此方式配置多个数据库节点的情形中，事务可以经由随机选择和 / 或任何其他合适的分发方法分布在各个节点之间。此外，这样的数据库节点可以同时地操作，如以上概括地描述的。

[0106] 图 4 是示出分布式事务管理组件 410 的示例性实现的框图。如图 4 所示，分布式事务管理组件 410 可以利用日志组件 420、提交时间戳同步器组件 430、和 / 或重试管理器组件 440 来便于对分布式事务 400 的处理。在此处的各个实施例中，日志组件 420 将与分布式事务 400 相关联的读 / 写事件记录到持久存储的日志和 / 或另一个合适的数据存储中。

[0107] 提交时间戳同步器组件 430 便于参与分布式事务 400 的各个节点之间的提交时间戳的同步。例如，在事务协调器处操作的提交时间戳同步器组件 430 可以计算分布式事务 400 将要使用的全局提交时间戳，并且将该时间戳传递给分布式事务 400 中的参与节点。在另一个示例中，在分布式事务 400 中参与节点处操作的提交时间戳同步器组件 430 可以提交时间投票、或与分布式事务 400 的所需提交时间戳有关的其他信息，并且接收返回的全局提交时间戳，该节点基于该时间戳可提前其本地时钟以符合该提交时间戳。

[0108] 如果例如分布式事务 400 的指定的提交时间对于分布式事务中的一个或多个参与节点来说是不够的，则重试管理器组件 440 被配置为重复对分布式事务 400 的已尝试的启动。例如，接收到分布式事务的早于其本地时钟值的全局提交时间戳的节点可以利用重试管理器组件 440 来请求对事务的重试。因此，接收到请求重试一事务的事务协调器可以利用重试管理器组件 440 来重试该事务。

[0109] 虽然图 4 中分布式事务管理组件 410 被示为包括组件 420-440，但是可以理解的是，分布式事务管理系统不需要包括组件 420-440 中的每一个，如此处参考各个实施例所概括地描述的。此外，可以理解的是，在适当时分布式事务管理组件 410 的功能及其相关联的组件 420、430 和 440 可被组合或被进一步细分。

[0110] 图 5 是示出用于启动数据库系统中的分布式事务的示例性、非限制性过程的流程图。在 500，标识对与各个节点相对应的一个或多个数据存储中的数据进行操作的事务。在 510，尝试确立事务的全局提交时间戳。在 520，如果该尝试导致作出对中止的请求，则过程继续至 560。否则，如果尝试是成功的，则在 530，尝试基于全局提交时间戳来准备事务。在 540，如果作为该尝试的结果，作出对重试的请求，则过程返回到 510。此外，如果作为 530 处尝试的结果来请求中止事务，则过程继续至 560。否则，如果在 540，事务被视为准备好用于提交，则在 550 提交该事务。在 560，提交或中止事件（例如，如在该过程的之前动作中确定的）被任选地记录到日志。

[0111] 图 6 是示出用于协调数据库系统中关于分布式事务的一组节点的示例性、非限制性过程的流程图。在 600，接收与对数据存储中的数据进行操作的事务有关的信息，数据存储与该事务中涉及的各个参与节点相关联。在 610，从各个参与节点请求提交时间投票。在 620，响应于 610 处的请求，从各个参与节点接收提交时间投票。在 630，如果 620 处获得的提交时间投票中的一个或多个指示事务已被参与节点中的一个或多个所中止，则该事务被中止。否则，在 640，至少部分地基于提交时间投票来计算全局提交时间戳。在 650，在各个参与节点处将事务的提交同步到全局提交时间戳。

[0112] 图 7 是示出用于在分布式事务的上下文中与数据库系统的协调和同步的示例性、非限制性过程的流程图。在 700，从与分布式事务相关联的事务协调器接收关于该分布式事

务的信息。在 710，至少部分地基于本地时钟来确定分布式事务的提交时间投票。在 720，如果与全局事务相对应的本地事务已被中止，则分布式事务也被中止。否则，在 730，将分布式事务的提交时间投票传递给事务协调器。在 740，从事务协调器接收全局提交时间戳。在 750，将分布式事务的提交同步到全局提交时间戳。

[0113] 示例性联网和分布式环境

[0114] 本领域技术人员可以理解，此处所描述的分布式事务管理系统和方法的各实施例可以结合任何计算机或其它客户机或服务器设备来实现，该任何计算机或其它客户机或服务器设备可作为计算机网络的一部分来部署或者被部署在分布式计算环境中，并且可以连接到任何种类的数据存储。就此，此处所描述的各实施例可以在具有任意数量的存储器或存储单元以及出现在任意数量的存储单元上的任意数量的应用程序和进程的任何计算机系统和环境中实现。这包括但不限于具有部署在具有远程或本地存储的网络环境或分布式计算环境中的服务器计算机和客户计算机的环境。

[0115] 分布式计算通过计算设备和系统之间的通信交换提供了计算机资源和服务的共享。这些资源和服务包括信息的交换、对于诸如文件等对象的高速缓存存储和盘存储。这些资源和服务还包括多个处理单元之间的处理能力共享以便进行负载平衡、资源扩展、处理专门化等。分布式计算利用网络连接，从而允许客户机利用它们的集体力量来使整个企业受益。就此，各种设备可具有可如参考本发明的各实施例描述地参与事务管理机制的应用程序、对象或资源。

[0116] 图 8 提供了示例性的联网或分布式计算环境的示意图。该分布式计算环境包括计算对象 810、812 等以及计算对象或设备 820、822、824、826、828 等，这些计算对象或设备可包括如由应用程序 830、832、834、836、838 表示的程序、方法、数据存储、可编程逻辑等。可以理解的是，计算对象 810、88 等以及计算对象或设备 820、822、824、826、828 等可包括不同的设备，诸如 PDA、音频 / 视频设备、移动电话、MP3 播放器、个人计算机、膝上型计算机等。

[0117] 每个计算对象 810、812 等以及计算对象或设备 820、822、824、826、828 等可经由通信网络 828 或直接或间接地与一个或多个其他计算对象 810、812 等以及计算对象或设备 820、822、824、826、840 等通信。即使在图 8 中被示为单个元件，但通信网络 840 可包括向图 8 的系统提供服务的其他计算对象或解释设备，和 / 或可表示未示出的多个互连网络。每个计算对象 810、812 等或计算对象或设备 820、822、824、826、828 等还可以包含应用程序，诸如可以利用 API 或其他对象、软件、固件和 / 或硬件的、适于实现或与根据本发明的各实施例所提供的事务管理进行通信的应用程序 830、832、834、836、838。

[0118] 存在支持分布式计算环境的各种系统、组件和网络配置。例如，计算系统可以由有线或无线系统、本地网络或广泛分布的网络连接在一起。当前，许多网络被耦合至因特网，后者为广泛分布的计算提供了基础结构并包含许多不同的网络，但任何网络基础结构可用于变得与如各实施例中所描述的可串行化快照隔离系统相关联的示例性通信。

[0119] 由此，可以利用诸如客户机 / 服务器、对等、或混合体系结构的大量的网络拓扑结构和网络基础结构。“客户机”是使用与它无关的另一类或组服务的一个类或组中的成员。客户机可以是进程，即大致上是请求由另一程序或进程提供的服务的一组指令或任务。客户机进程利用所请求的服务，而不必“知道”有关其他程序或服务本身的任何工作细节。

[0120] 在客户机 / 服务器体系结构中，尤其在联网系统中，客户机通常是访问由例如服

务器的另一计算机提供的共享的网络资源的计算机。在图 8 的图示中，作为非限制性示例，计算对象或设备 820、822、824、826、828 等可被认为是客户机，而计算对象 810、812 等可被认为是服务器，其中计算对象 810、812 等担任提供数据服务的服务器，诸如从客户机计算对象或设备 820、822、824、826、828 等接收数据，存储数据，处理数据，向客户机计算对象或设备 820、822、824、826、828 等传送数据，但任何计算机都可取决于环境而被认为是客户机、服务器或两者。这些计算设备中的任一个都可以处理数据，或请求可包含此处一个或多个实施例所描述的事务管理和多版本化技术的事务服务或任务。

[0121] 服务器通常是可通过诸如因特网或无线网络基础结构的远程网络或本地网络可访问的远程计算机系统。客户机进程在第一计算机系统中可以是活动的，而服务器进程在第二计算机系统中可以是活动的，它们通过通信介质彼此通信，从而提供分布式功能并允许多个客户机利用服务器的信息收集能力。按照此处描述的技术来利用的任何软件对象可以单独提供或分布在多个计算设备或对象上。

[0122] 在通信网络 840 或总线是因特网的网络环境中，例如，计算对象 810、812 等可以是其他计算对象或设备 820、822、824、826、828 等经由诸如超文本传输协议 (HTTP) 等多种已知协议中的任一种与其通信的 web 服务器。担任服务器的计算对象 810、812 等还可担任诸如计算对象或设备 820、822、824、826、828 等的客户机，这可以是分布式计算环境的特性。

[0123] 示例性计算设备

[0124] 如上所述，有利的是，此处所描述的技术可适用于期望执行分布式事务管理的任何设备。因此，应当理解的是，构想了所有种类的手持式、便携式和其他计算设备和计算对象来用于各实施例，即，在设备可能期望从数据存储读事务或向数据存储写事务的任何地方。因此，以下在图 9 中描述的通用远程计算机只是计算设备的一个示例。此外，数据库服务器可包括以下通用计算机或其他数据库管理服务器组件的一个或多个方面。

[0125] 尽管并非所需，但各实施例可以部分地经由操作系统来实现，以供设备或对象的服务开发者使用，和 / 或被包括在用于执行此处所描述的各实施例的一个或多个功能方面的应用程序软件中。软件可以在由诸如客户机工作站、服务器或其他设备等一个或多个计算机执行的诸如程序模块等计算机可执行指令的通用上下文中描述。本领域的技术人员可以理解，计算机系统具有可用于传递数据的各种配置和协议，并因此没有特定配置或协议应被认为是限制性的。

[0126] 因此，图 9 示出了其中可实现各实施例的一个或多个方面的合适的计算系统环境 900 的一个示例，尽管如上所述，计算系统环境 900 仅为合适的计算环境的一个示例，并非对使用范围或功能提出任何限制。也不应将计算系统环境 900 解释为对在示例性计算系统环境 900 中示出的组件中的任何一个或其组合有任何依赖或要求。

[0127] 参考图 9，用于实现一个或多个实施例的示例性远程设备包括计算机 910 形式的通用计算设备。计算机 910 的组件可以包括，但不仅限于，处理单元 920、系统存储器 930，以及将包括系统存储器的各种系统组件耦合到处理单元 920 的系统总线 922。

[0128] 计算机 910 通常包括各种计算机可读介质，并可以是可由计算机 910 访问的任何可用介质。系统存储器 930 可以包括诸如只读存储器 (ROM) 和 / 或随机存取存储器 (RAM) 等易失性和 / 或非易失性存储器形式的计算机存储介质。作为示例而非限制，系统存储器 930 还可包括操作系统、应用程序、其他程序模块、和程序数据。

[0129] 用户可以通过输入设备 940 向计算机 910 输入命令和信息。监视器或其他类型的显示设备也经由接口，诸如输出接口 950 连接至系统总线 922。除监视器之外，计算机还可以包括其他外围输出设备，如扬声器和打印机，它们可以通过输出接口 950 连接。

[0130] 计算机 910 可使用至一个或多个远程计算机，诸如远程计算机 970 的逻辑连接在联网或分布式环境中操作。远程计算机 970 可以是个人计算机、服务器、路由器、网络 PC、对等设备或其他常见网络节点、或任何其他远程媒体消费或传输设备，并且可以包括上面关于计算机 910 所描述的任何或全部元件。图 9 所描绘的逻辑连接包括诸如局域网 (LAN) 或广域网 (WAN) 等的网络 972，但也可以包括其他网络 / 总线。这样的联网环境在家庭、办公室、企业范围计算机网络、内联网和因特网中是常见的。

[0131] 如上所述，尽管结合各种计算设备和网络体系结构描述了各示例性实施例，但底层概念可被应用于任何网络系统和任何计算设备或系统。

[0132] 而且，存在实现相同或相似功能的多种方法，例如适当的 API、工具箱、驱动程序代码、操作系统、控件、独立或可下载软件对象等，它们使得应用程序和服务能够使用此处提供的技术。由此，从 API（或其他软件对象）的观点以及从实现包括此处描述的确认测试的并发控制的一个或多个方面的软件或硬件对象构想此处的各实施例。因此，此处描述的各实施例可以具有完全采用硬件、部分采用硬件并且部分采用软件、以及采用软件的方面。

[0133] 此处使用的词语“示例性”意味着用作示例、范例或说明。为避免疑惑，此处公开的主题不受限于这样的示例。此外，此处描述为“示例性”的任何方面或设计不必解释成优于其他方面或设计或比其他方面或设计有利，它也不旨在排除本领域的普通技术人员所知的等效示例性结构和技术。此外，就术语“包括”、“具有”、“包含”和其他类似的词语的使用而言，为避免疑惑，这样的术语旨在以类似于术语“包括”作为开放的过渡词的方式解释而不排除任何附加或其他元素。

[0134] 如上所述，此处所述的各种技术可结合硬件或软件，或在适当时以两者的组合来实现。如在此所使用的，术语“组件”、“系统”等同样指的是计算机相关实体，或者是硬件、硬件和软件的组合、软件或执行中的软件。例如，组件可以是，但不限于是，在处理器上运行的进程、处理器、对象、可执行码、执行的线程、程序和 / 或计算机。作为说明，运行在计算机上的应用程序和计算机本身都可以是计算机组件。一个或多个组件可以驻留在进程和 / 或执行的线程中，并且组件可以位于一个计算机内和 / 或分布在两个或更多的计算机之间。

[0135] 如前所述的系统是利用多个组件之间的交互来描述的。可以理解的是，这样的系统和组件可以包括这些组件或指定的子组件，某些指定的组件或子组件，和 / 或附加的组件，并根据前述的内容的各种置换和组合。子组件也可以作为可通信地耦合到其他组件的组件来实现，而不是包括在父组件内（层次性）。另外，应注意到一个或多个组件可被组合成提供聚集功能的单个组件，或被分成若干单独的子组件，且诸如管理层等任何一个或多个中间层可被设置成通信耦合到这样的子组件以便提供集成功能。此处所描述的任何组件也可以与一个或多个此处没有专门描述的但本领域技术人员广泛地知道的其他组件进行交互。

[0136] 考虑到以上描述的示例性系统，参考各附图的流程图将也可以理解依照所描述的主题实现的方法。尽管为了说明简洁起见，作为一系列框示出和描述了方法，但是，应该理解，各实施例不仅限于所描述框的顺序，一些框可以按与此处所描绘和描述的不同的顺序

进行和 / 或与其他框并发地进行。尽管经由流程图示出了非顺序或分支的流程，但可以理解，可实现达成相同或类似结果的各种其他分支、流程路径和框次序。此外，并非全部所示出的框都是实现下面所描述的方法所必需的。

[0137] 除了此处所描述的各实施例之外，可以理解，可以使用其他相似的实施例或者可对所述实施例作出修改和添加以便执行对应的实施例的相同或等效的功能而不背离这些实施例。此外，多个处理芯片或多个设备可共享此处所描述的一个或多个功能的执行，并且类似地，存储可以跨多个设备实现。因此，本发明不应限于任何单个实施例，而是应该根据所附权利要求书的广度和范围来解释。

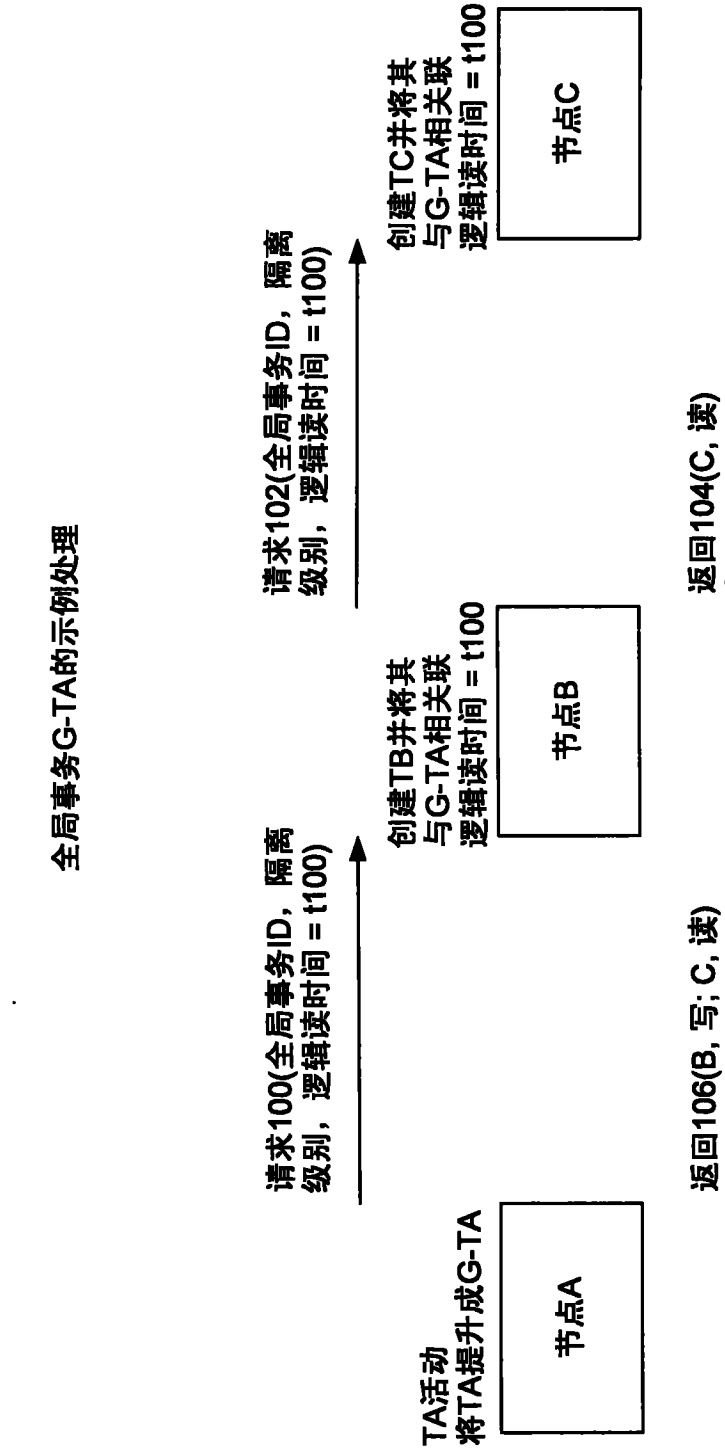

全局事务G-TA的示例处理
节点A
节点B
节点C
TA活动
将TA提升成G-TA
请求100(全局事务ID，隔离
级别，逻辑读时间 = t100)
创建TB并将其
与G-TA相关联
逻辑读时间 = t100
请求102(全局事务ID，隔离
级别，逻辑读时间 = t100)
创建TC并将其
与G-TA相关联
逻辑读时间 = t100
返回104(C, 读)
返回106(B, 写; C, 读)

图 1

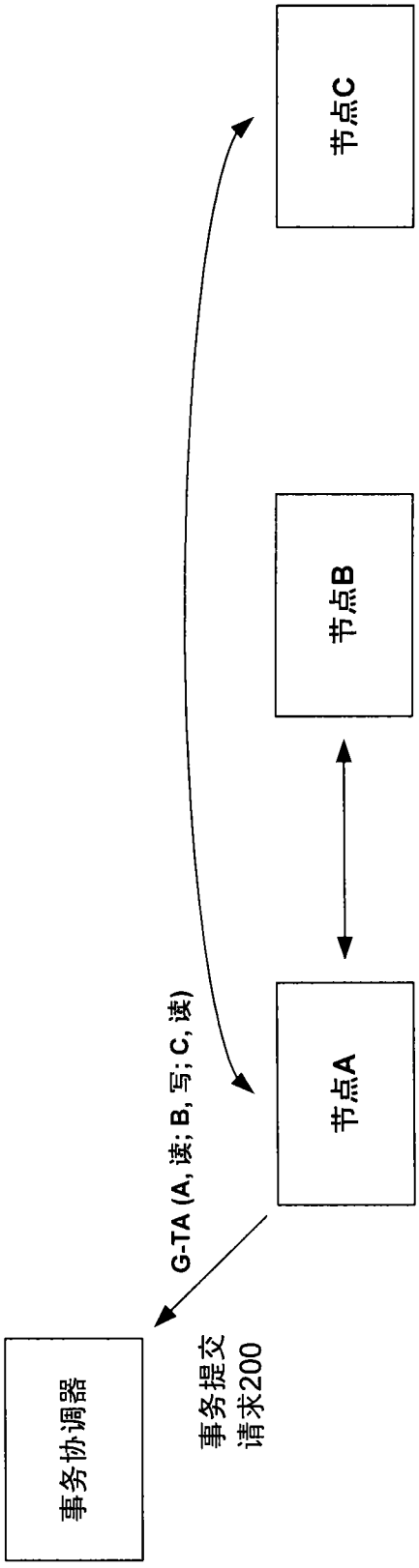


图 2

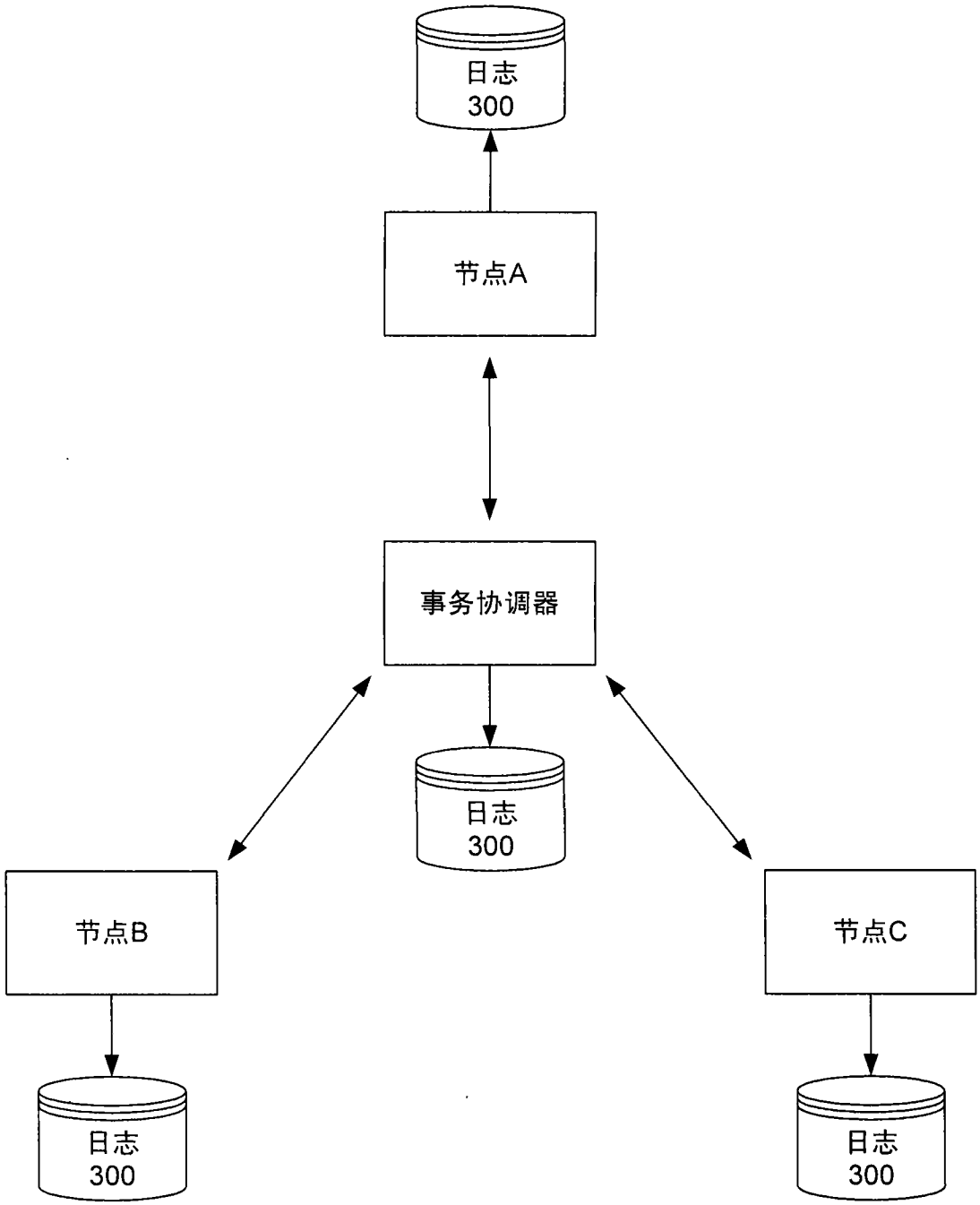


图 3

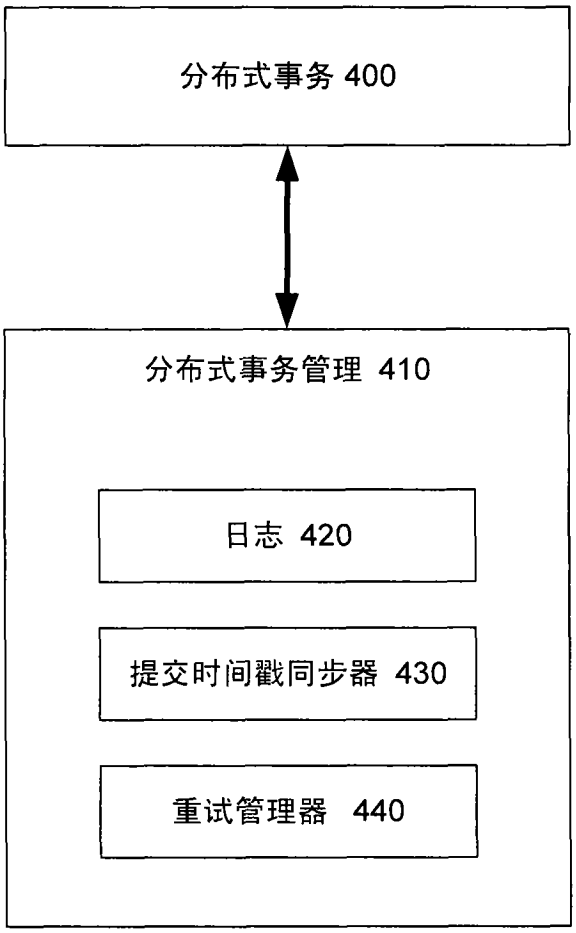


图 4

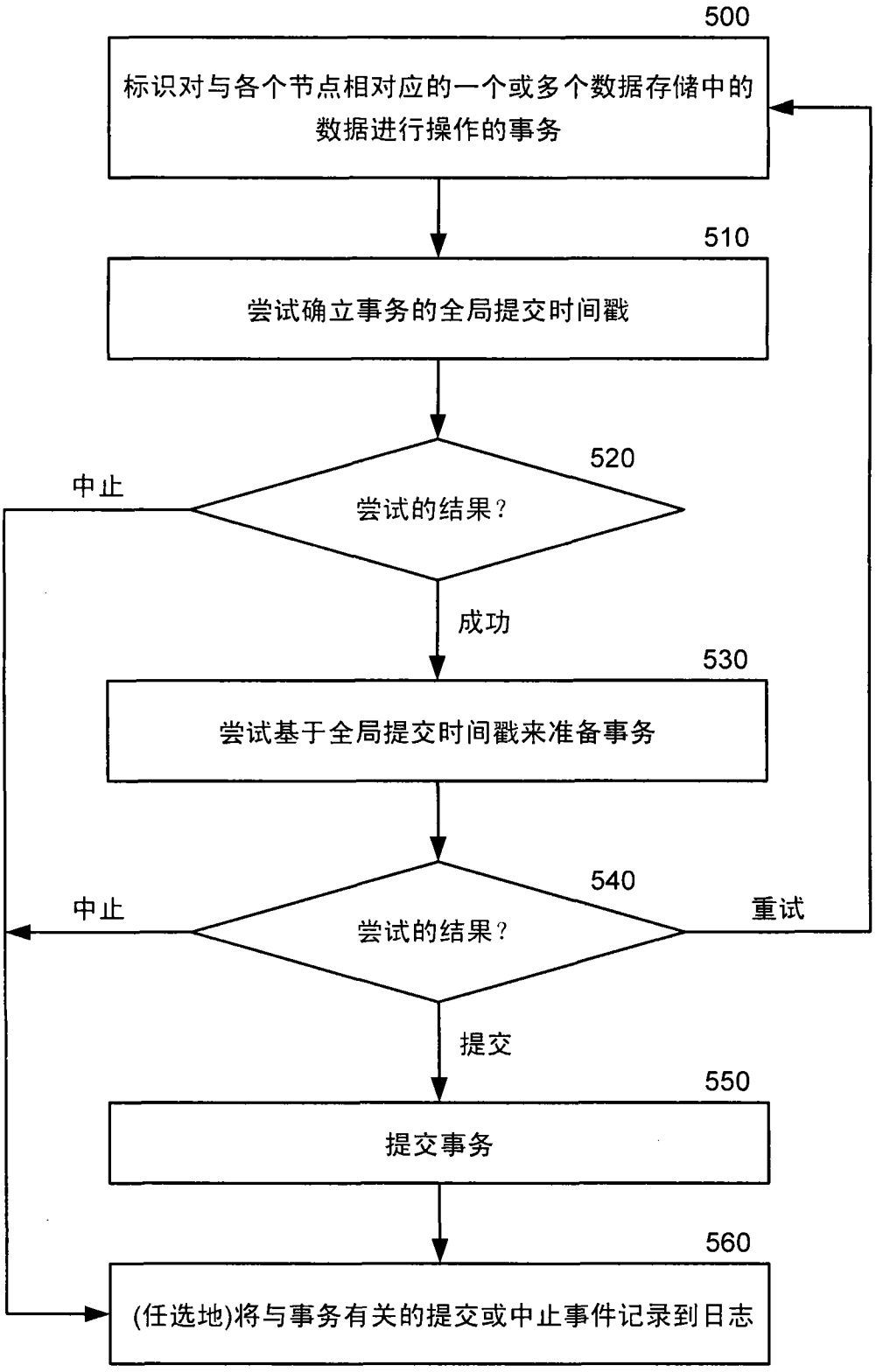

500
标识对与各个节点相对应的一个或多个数据存储中的数据进行操作的事务
510
尝试确立事务的全局提交时间戳
520
尝试的结果？
中止
成功
530
尝试基于全局提交时间戳来准备事务
540
尝试的结果？
中止
重试
提交
550
提交事务
560
(任选地)将与事务有关的提交或中止事件记录到日志

图 5

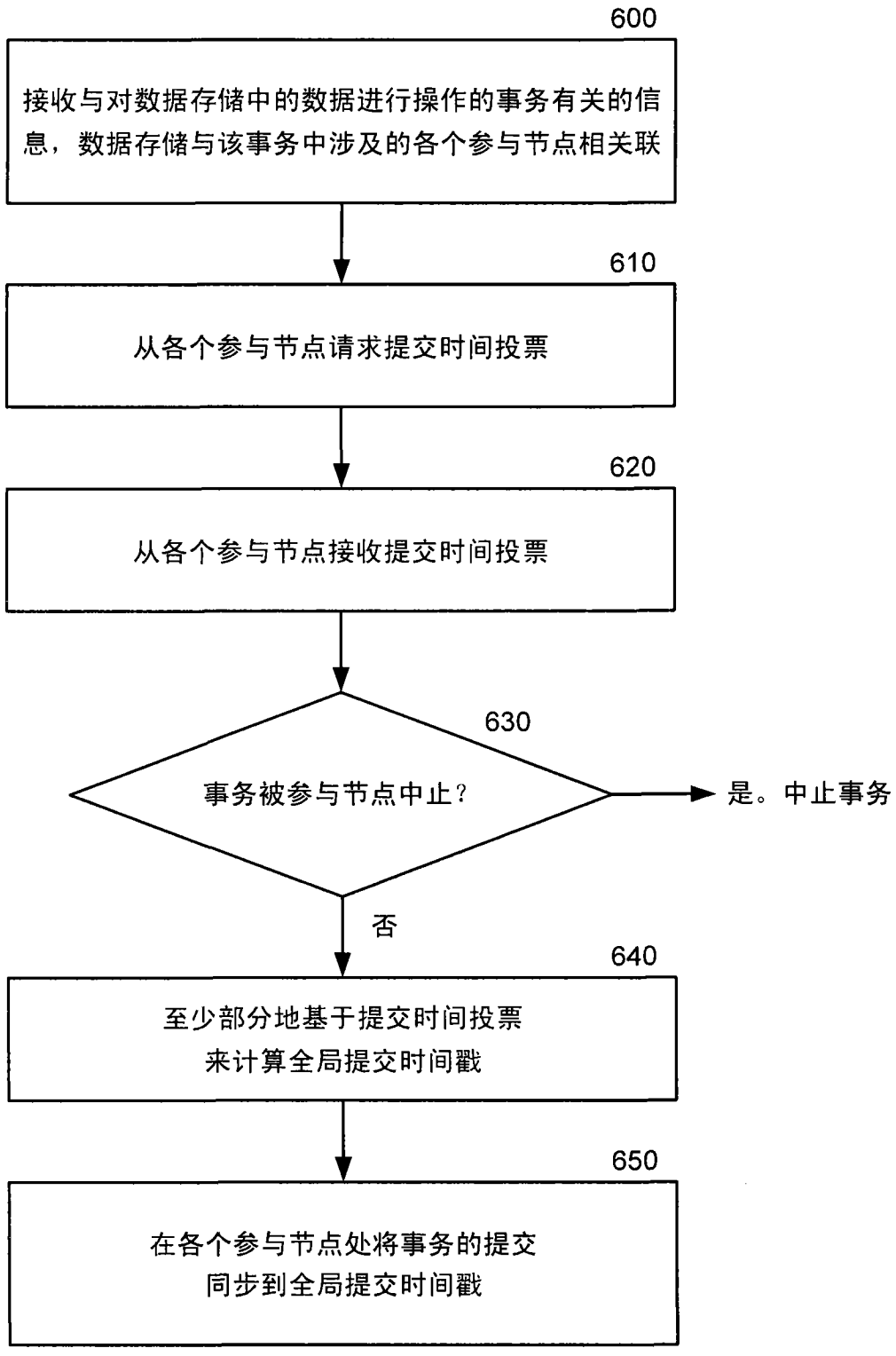

600
接收与对数据存储中的数据进行操作的事务有关的信息，数据存储与该事务中涉及的各个参与节点相关联
610
从各个参与节点请求提交时间投票
620
从各个参与节点接收提交时间投票
630
事务被参与节点中止？
是。中止事务
否
640
至少部分地基于提交时间投票来计算全局提交时间戳
650
在各个参与节点处将事务的提交同步到全局提交时间戳

图 6

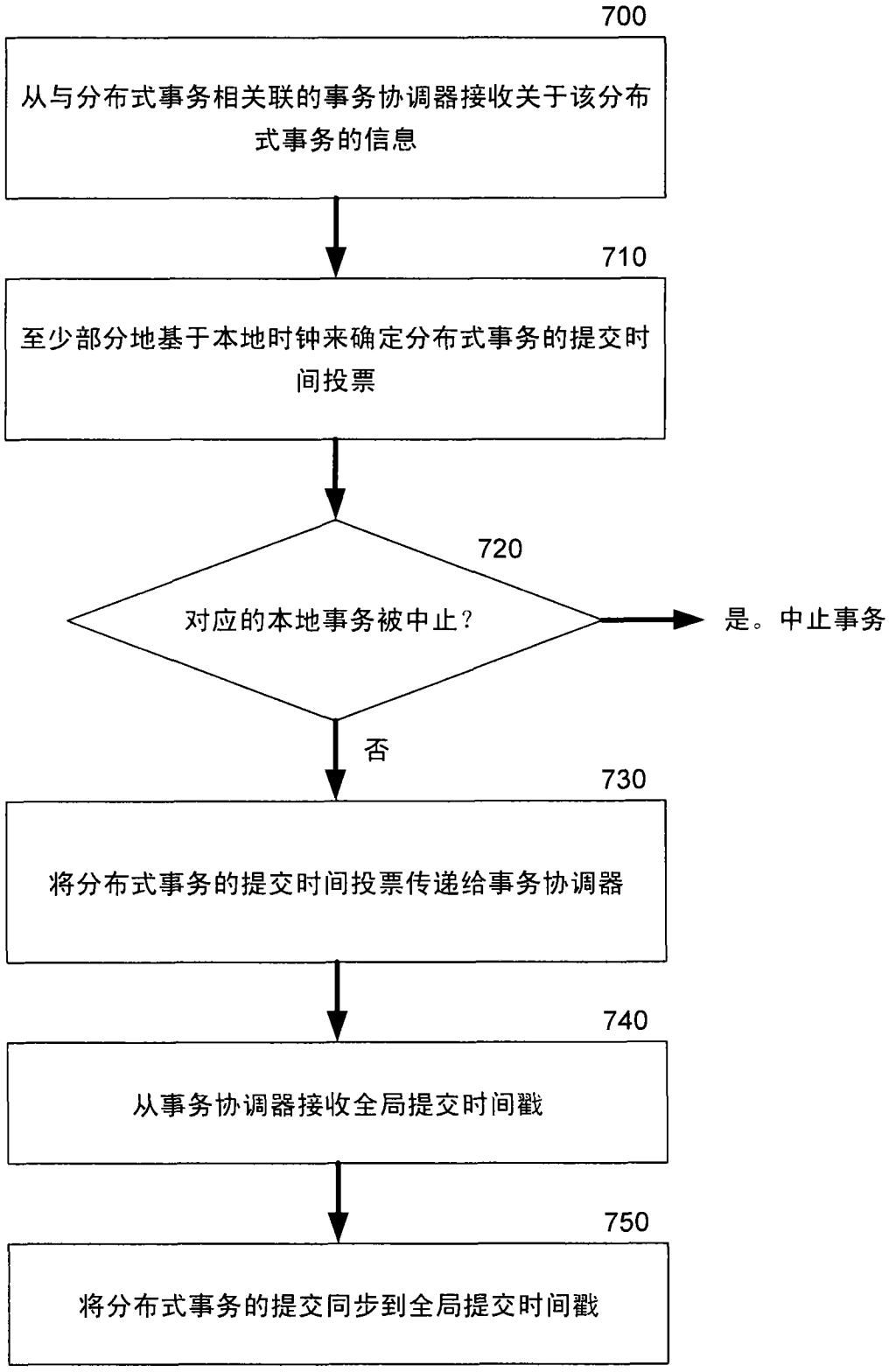

700
从与分布式事务相关联的事务协调器接收关于该分布式事务的信息
710
至少部分地基于本地时钟来确定分布式事务的提交时间投票
720
对应的本地事务被中止？
是。中止事务
否
730
将分布式事务的提交时间投票传递给事务协调器
740
从事务协调器接收全局提交时间戳
750
将分布式事务的提交同步到全局提交时间戳

图 7

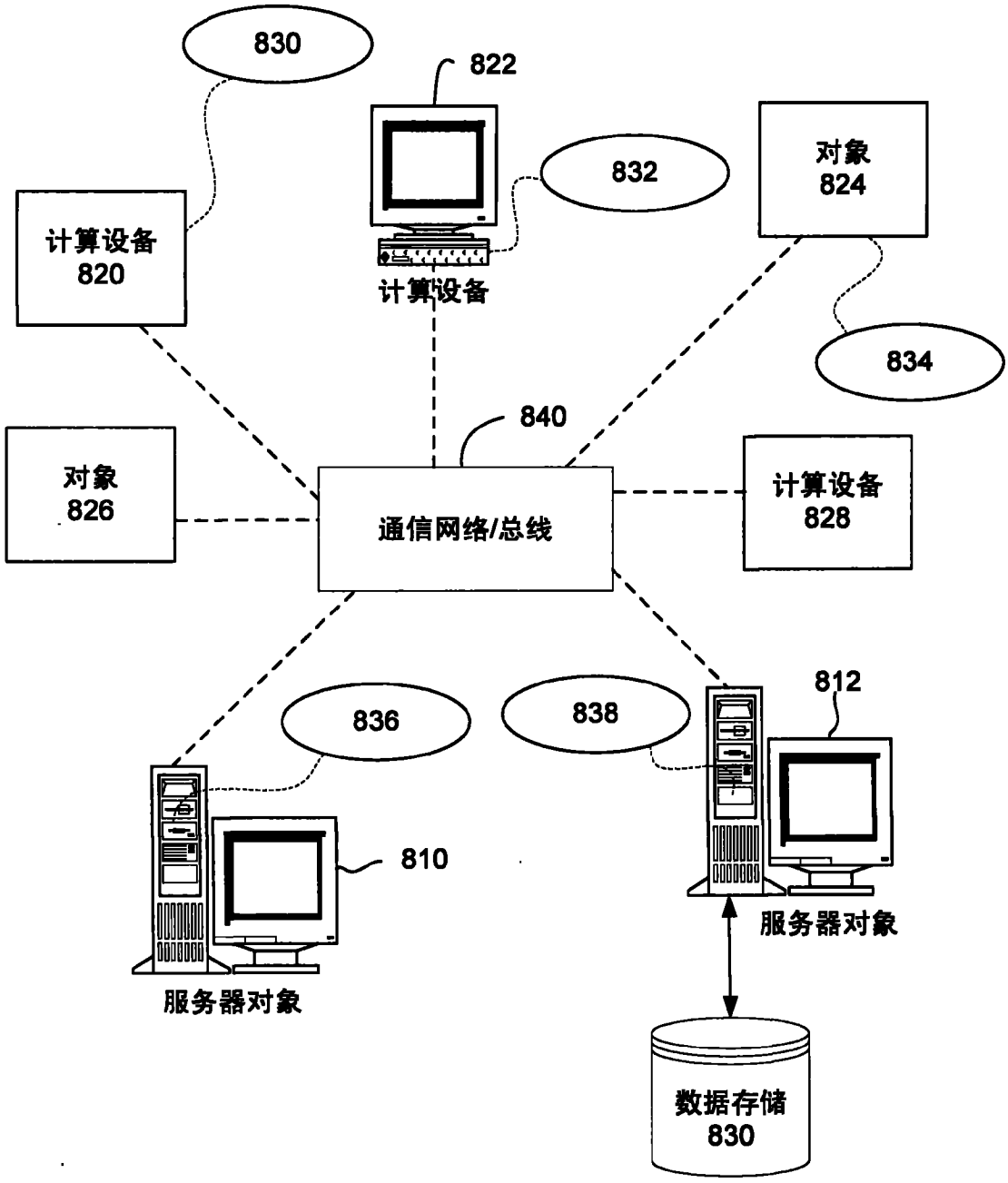

830
822
832
对象
824
计算设备
820
计算设备
834
840
对象
826
通信网络/总线
计算设备
828
812
836
838
810
服务器对象
服务器对象
数据存储
830

图 8

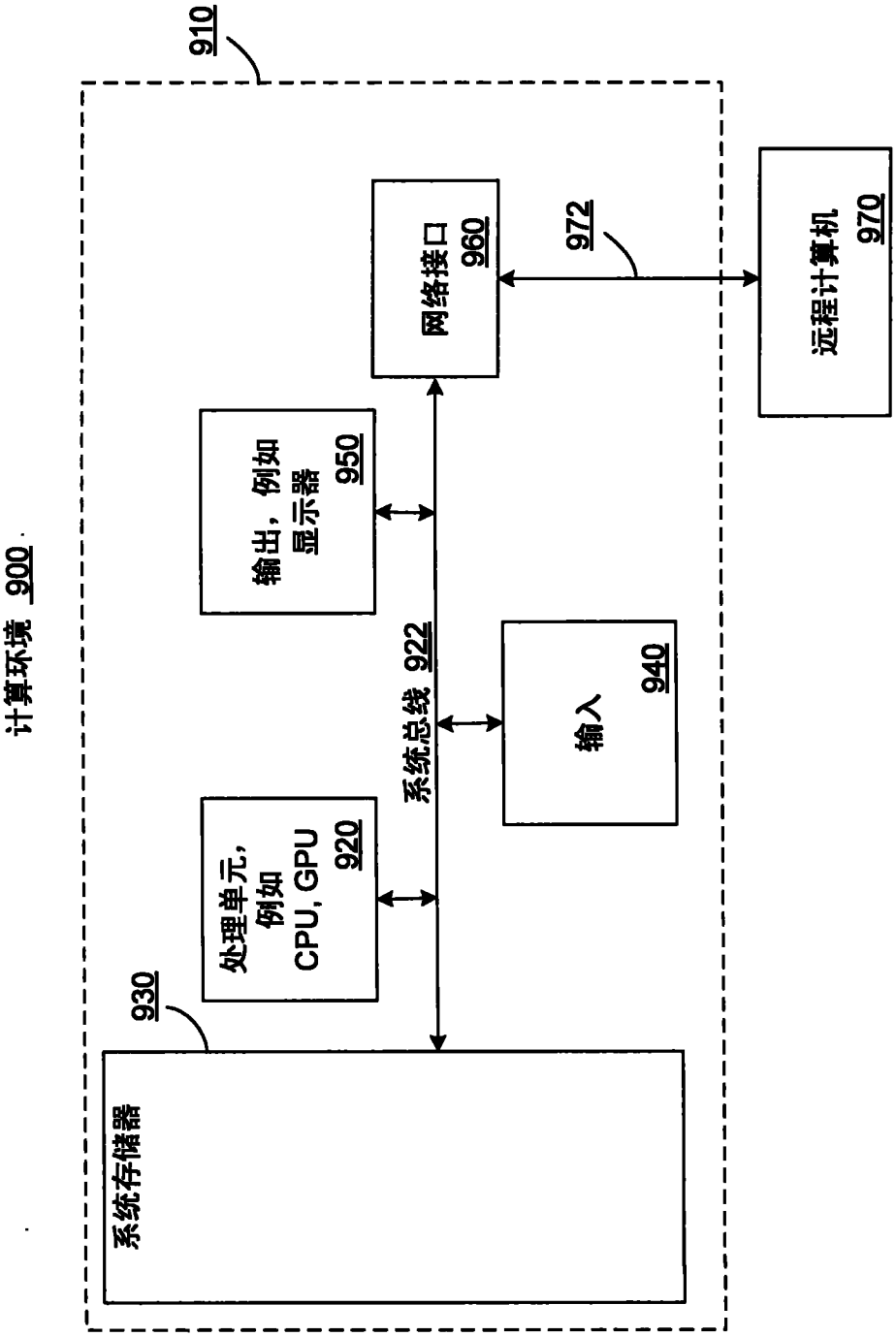

计算环境 900
910
网络接口 960
972
远程计算机 970
输出，例如显示器 950
系统总线 922
输入 940
处理单元，例如 CPU, GPU 920
930
系统存储器

图 9