



(12) 发明专利

(10) 授权公告号 CN 109863485 B

(45) 授权公告日 2023. 04. 14

(21) 申请号 201780064336.6

(22) 申请日 2017.09.07

(65) 同一申请的已公布的文献号
申请公布号 CN 109863485 A

(43) 申请公布日 2019.06.07

(30) 优先权数据
62/395,044 2016.09.15 US
15/696,501 2017.09.06 US

(85) PCT国际申请进入国家阶段日
2019.04.18

(86) PCT国际申请的申请数据
PCT/US2017/050514 2017.09.07

(87) PCT国际申请的公布数据
W02018/052792 EN 2018.03.22

(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 H·帕克 S·比施诺伊
P·图卡拉姆

(74) 专利代理机构 中国贸促会专利商标事务所
有限公司 11038
专利代理师 边海梅

(51) Int.Cl.
G06F 16/2455 (2019.01)

(56) 对比文件
US 2016085810 A1,2016.03.24
US 2015169683 A1,2015.06.18
US 2016070739 A1,2016.03.10
US 2014201355 A1,2014.07.17
US 2015363464 A1,2015.12.17
US 2016085809 A1,2016.03.24
CN 104838377 A,2015.08.12

审查员 陈晓

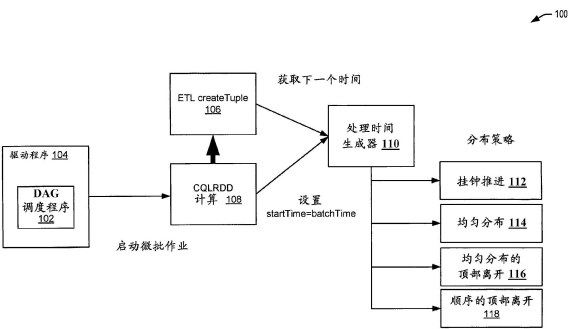
权利要求书2页 说明书20页 附图9页

(54) 发明名称

用于处理事件流中的事件的方法、系统、存储介质及装置

(57) 摘要

公开了一种用于处理事件流中的事件的方法、系统、存储介质及装置。该系统被配置用于对流处理器进行配置以对来自流源的传入事件进行微批处理。该系统还被配置用于为传入事件的微批生成单个时间戳和/或从流源接收传入事件的微批。该系统还可以被配置用于向微批的每个事件分配单个时间戳和/或为微批的每个相应事件生成单独的时间戳值。在一些示例中,该系统还可以被配置用于为微批的每个相应事件分配单独的时间戳值中的个体时间戳值。



1. 一种用于处理事件流中的事件的方法,包括:
对流处理器进行配置以对来自流源的传入事件进行微批处理;
为所述传入事件的微批生成单个时间戳;
从所述流源接收所述传入事件的所述微批;
向所述微批的每个事件分配所述单个时间戳;
至少基于包括挂钟推进算法、均匀分布算法、均匀分布的顶部离开算法或简单序列算法的算法,为所述微批的每个相应事件生成唯一的时间戳值;以及
为所述微批的每个相应事件分配所生成的唯一的时间戳值。
2. 如权利要求1所述的方法,其中分配所生成的唯一的时间戳值包括将所生成的唯一的时间戳值附加到所述微批的每个事件的所述单个时间戳。
3. 如权利要求1所述的方法,其中所述算法确保所述微批的每个事件能够被识别为所述微批的一部分。
4. 如权利要求1至3中任一项所述的方法,还包括将每个事件分配给一个或多个连续查询语言引擎分区。
5. 如权利要求4所述的方法,还包括:
识别所述一个或多个连续查询语言引擎分区中的第一分区;以及
确定所述第一分区具有比所述微批中被分配给所述第一分区的多个事件中的事件的数量更大的窗口尺寸。
6. 如权利要求5所述的方法,还包括:
为所述第一分区生成心跳事件;以及
将所述心跳事件插入到所述第一分区中。
7. 一种用于处理事件流中的事件的系统,包括:
存储器,被配置为存储计算机可执行指令;以及
处理器,被配置为访问所述存储器并执行所述计算机可执行指令,以至少进行以下操作:
对流处理器进行配置以对来自流源的传入事件进行微批处理;
为所述传入事件的微批生成单个时间戳;
从所述流源接收所述传入事件的所述微批;
向所述微批的每个事件分配所述单个时间戳;
至少基于包括挂钟推进算法、均匀分布算法、均匀分布的顶部离开算法或简单序列算法的算法,为所述微批的每个相应事件生成唯一的时间戳值;以及
为所述微批的每个相应事件分配所生成的唯一的时间戳值。
8. 如权利要求7所述的系统,其中分配所生成的唯一的时间戳值包括将所生成的唯一的时间戳值附加到所述微批的每个事件的所述单个时间戳。
9. 如权利要求7所述的系统,其中所述算法确保所述微批的每个事件能够被识别为所述微批的一部分。
10. 如权利要求7至9中任一项所述的系统,其中所述处理器还被配置为执行所述计算机可执行指令,以至少进行以下操作:
将每个事件分配给一个或多个连续查询语言引擎分区;

识别所述一个或多个连续查询语言引擎分区中的第一分区；

确定所述第一分区具有比所述微批中被分配给所述第一分区的多个事件中的事件的数量更大的窗口尺寸；

为所述第一分区生成心跳事件；以及

将所述心跳事件插入到所述第一分区中。

11. 一种存储计算机可执行代码的计算机可读介质，所述计算机可执行代码在由处理器执行时使所述处理器执行包括以下的操作：

对流处理器进行配置以对来自流源的传入事件进行微批处理；

为所述传入事件的微批生成单个时间戳；

从所述流源接收所述传入事件的所述微批；

为所述微批的每个事件分配所述单个时间戳；

至少基于包括挂钟推进算法、均匀分布算法、均匀分布的顶部离开算法或简单序列算法的算法，为所述微批的每个相应事件生成唯一的时间戳值；以及

为所述微批的每个相应事件分配所生成的唯一的时间戳值。

12. 如权利要求11所述的计算机可读介质，其中分配所生成的唯一的时间戳值包括将所生成的唯一的时间戳值附加到所述微批的每个事件的所述单个时间戳。

13. 如权利要求11所述的计算机可读介质，其中所述算法确保所述微批的每个事件能够被识别为所述微批的一部分。

14. 如权利要求11至13中任一项所述的计算机可读介质，其中所述操作包括：

将每个事件分配给一个或多个连续查询语言引擎分区；

识别所述一个或多个连续查询语言引擎分区中的第一分区；

确定所述第一分区具有比所述微批中被分配给所述第一分区的多个事件中的事件的数量更大的窗口尺寸；

为所述第一分区生成心跳事件；以及

将所述心跳事件插入到所述第一分区中。

15. 一种用于处理事件流中的事件的装置，所述装置包括用于执行如权利要求1至6中任一项所述的方法的部件。

用于处理事件流中的事件的方法、系统、存储介质及装置

[0001] 对相关申请的交叉引用

[0002] 本申请依据35 U.S.C§119 (e) 要求于2016年9月15日提交的标题为“PROCESSING TIMESTAMPS AND HEARTBEAT EVENTS FOR AUTOMATIC TIME PROGRESSION”的美国临时申请 62/395,044的权益和优先权,该申请的全部内容为了所有目的通过引用并入本文。

背景技术

[0003] 在传统的数据库系统中,数据通常以表的形式存储在一个或多个数据库中。然后使用数据管理语言(诸如结构化查询语言(SQL))来查询和操纵所存储的数据。例如,可以定义和执行SQL查询以从存储在数据库中的数据中识别相关数据。因此SQL查询针对存储在数据库中的有限数据集来执行。此外,当执行SQL查询时,SQL查询针对有限数据集来执行一次并产生有限静态结果。因此数据库最佳地被配备为对有限的存储的数据集运行查询。

[0004] 但是,许多现代应用和系统以连续数据或事件流而不是有限数据集的形式生成数据。这样的应用的示例包括但不限于传感器数据应用、金融报价机(ticker)、网络性能测量工具(例如,网络监视和业务管理应用)、点击流分析工具、汽车交通监视等。这种应用已经引起了对可以处理数据流的新型应用的需求。例如,温度传感器可以被配置为发出温度读数。

[0005] 管理和处理这些类型的基于事件流的应用的数据涉及构建具有强大时间焦点的数据管理和查询能力。需要包括针对连续的无界(unbounded)数据集的长期运行的查询的不同种类的查询机制。虽然一些供应商现在供应面向事件流处理的产品套件,但是这些产品供应物仍然缺乏处置当今的事件处理需求所需的处理灵活性。

发明内容

[0006] 提供了用于对流处理器进行配置以对来自流源的传入事件进行微批处理(micro-batch)的技术(例如,方法、系统、存储能够由一个或多个处理器执行的代码或指令的非瞬态计算机可读介质)。在实施例中,该系统被配置用于为传入事件的微批(micro-batch)生成单个时间戳和/或从流源接收传入事件的微批。该系统还被配置用于向微批的每个事件分配单个时间戳和/或至少部分地基于算法为微批的每个相应事件生成单独的时间戳值。在一些示例中,该系统还可以被配置用于为微批的每个相应事件分配单独的时间戳值中的个体时间戳值。

[0007] 在某些实施例中,该算法包括挂钟推进算法、均匀分布算法、均匀分布的顶部离开(top off)算法或简单序列算法。在一些示例中,分配单独的时间戳值中的个体时间戳值包括将单独的时间戳值中的个体时间戳值附加到微批的每个事件的单个时间戳。此外,该算法为微批的每个事件提供唯一的时间戳和/或该算法确保微批的每个事件能够被识别为微批的一部分。在一些情况下,该系统还可以被配置用于将每个事件分配给一个或多个连续查询语言引擎分区、识别该一个或多个连续查询语言引擎分区中的第一分区,和/或确定第一分区具有比批中分配给第一分区的多个事件中的事件的数量更大的窗口尺寸。此外,在

一些情况下,该系统可以被配置用于为第一分区生成心跳事件和/或将该心跳事件插入到第一分区中。

[0008] 以上和以下描述的技术可以以许多方式和在许多上下文中实现。参考如下面更详细描述以下各图提供了若干示例实现和上下文。但是,以下实现和上下文只是许多实现和上下文中的一小部分。

附图说明

[0009] 图1描绘了根据本公开的实施例的示例事件处理系统的方面。

[0010] 图2是可以结合本公开的实施例的事件处理系统的简化高级图。

[0011] 图3是图示根据本公开的实施例的分布式事件处理系统的示例用例的框图。

[0012] 图4是根据本公开的一个实施例的处理的示例流程图。

[0013] 图5是根据本公开的另一个实施例的处理的示例流程图。

[0014] 图6是根据本公开的另一个实施例的处理的示例流程图。

[0015] 图7描绘了用于实现本公开的实施例的分布式系统的简化图。

[0016] 图8是根据本公开的实施例的系统环境的一个或多个部件的简化框图,通过该系统环境,由实施例系统的一个或多个部件提供的服务可以被供应为云服务。

[0017] 图9图示了可以用于实现本公开的实施例的示例计算机系统。

具体实施方式

[0018] 在以下描述中,将描述各种实施例。出于解释的目的,阐述具体配置和细节以便提供对实施例的透彻理解。但是,对于本领域技术人员来说将显而易见的是,可以在没有具体细节的情况下实践这些实施例。此外,可以省略或简化众所周知的特征,以免模糊所描述的实施例。

[0019] 复杂事件处理 (CEP) 的概述

[0020] 复杂事件处理 (CEP) 提供用于基于事件驱动的体系架构来构建应用的模块化平台。CEP平台的核心处是连续查询语言 (CQL), 这允许应用使用声明性的类似SQL的语言对数据流进行过滤、查询和执行模式匹配操作。开发人员可以将CQL与轻量级Java编程模型结合使用来编写应用。其它平台模块包括特征丰富的IDE、管理控制台、聚类、分布式高速缓存、事件储存库和监视等。

[0021] 随着事件驱动的体系架构和复杂事件处理已经变成企业计算领域的突出特征,越来越多的企业已开始使用CEP技术来构建任务关键 (mission-critical) 应用。如今,任务关键CEP应用可以在许多不同的行业中找到。例如,CEP技术在电力行业中使用,以通过允许它们立即对电力需求的变化作出反应而使公用事业更高效。CEP技术在信用卡行业中使用,以在潜在欺诈性交易发生时实时地检测它们。任务关键CEP应用的列表继续增长。使用CEP技术构建任务关键应用导致需要使CEP应用具有高可用性和容错性。

[0022] 当今的信息技术 (IT) 环境为从监视金融市场和网络性能到业务处理执行和跟踪RFID标签的资产的每样事物生成连续的数据流。CEP为开发事件处理应用提供丰富的声明性环境,以提高业务操作的有效性。CEP可以处理多个事件流,以实时检测模式和趋势并为企业提供必要的可见性,以利用新出现的机会或减轻发展风险。

[0023] 连续的数据流(也被称为事件流)可以包括没有明确结束的、本质上可以是连续的或无界的事件或数据的流。逻辑上,事件流或数据流可以是数据元素(也被称为事件)的序列,每个数据元素具有相关联的时间戳。连续的事件流可以在逻辑上被表示为元素(s,T)的包或集合,其中“s”表示数据部分,并且“T”在时域中。“s”部分一般被称为元组(tuple)或事件。因此事件流可以是带时间戳的元组或事件的序列。

[0024] 在一些方面,与流中的事件相关联的时间戳可以等同于时钟时间。但是,在其它示例中,与事件流中的事件相关联的时间可以由应用域定义,并且可以不与时钟时间对应,而是可以代替地例如由序列号表示。因而,与事件流中的事件相关联的时间信息可以由数字、时间戳或表示时间概念的任何其它信息表示。对于接收输入事件流的系统,事件按照递增时间戳的次序到达系统。可以存在具有相同时间戳的多于一个的事件。

[0025] 在一些示例中,事件流中的事件可以表示某个世间事件(worldly event)的发生(例如,温度传感器将值改变为新值,股票代码的价格改变)并且与事件相关联的时间信息可以指示由数据流事件表示的世间事件何时发生。

[0026] 对于经由事件流接收的事件,可以使用与事件相关联的时间信息来确保事件流中的事件按照递增的时间戳值的次序到达。这可以使事件流中接收到的事件能够基于它们相关联的时间信息被排序。为了实现这种排序,时间戳可以以非递减的方式与事件流中的事件相关联,使得较晚生成的事件具有比较早生成的事件晚的时间戳。作为另一个示例,如果序列号被用作时间信息,则与较晚生成的事件相关联的序列号可以大于与较早生成的事件相关联的序列号。在一些示例中,例如,当由数据流事件表示的世间事件同时发生时,多个事件可以与相同的时间戳或序列号相关联。属于相同事件流的事件一般可以按照由相关联的时间信息施加在事件上的次序来处理,其中较早的事件在较晚的事件之前被处理。

[0027] 与事件流中的事件相关联的时间信息(例如,时间戳)可以由流的源设置,或者替代地可以由接收流的系统设置。例如,在某些实施例中,可以在接收事件流的系统上维持心跳,并且与事件相关联的时间可以基于由心跳测得的事件到达系统处的时间。事件流中的两个事件有可能具有相同的时间信息。要注意的是,虽然时间戳排序要求特定于一个事件流,但是不同流的事件可以被任意交织。

[0028] 事件流具有相关联的模式“S”,该模式包括时间信息和一个或多个命名属性的集合。属于特定事件流的所有事件符合与该特定事件流相关联的模式。因而,对于事件流(s,T),事件流可以具有作为(<time_stamp>,<attribute(S)>)的模式“S”,其中<attributes>表示模式的数据部分并且可以包括一个或多个属性。例如,股票报价机事件流的模式可以包括属性<stock symbol(股票代码)>和<stock price(股票价格)>。经由这样的流接收的每个事件将具有时间戳和这两个属性。例如,股票报价机事件流可以接收以下事件和相关联的时间戳:

[0029] ...

[0030] (<timestamp_N>,<NVDA,4>)

[0031] (<timestamp_N+1>,<ORCL,62>)

[0032] (<timestamp_N+2>,<PCAR,38>)

[0033] (<timestamp_N+3>,<SPOT,53>)

[0034] (<timestamp_N+4>,<PDCO,44>)

[0035] (<timestamp_N+5>, <PTEN, 50>)

[0036] ...

[0037] 在上面的流中, 对于流元素(<timestamp_N+1>, <ORCL, 62>), 事件是属性为“stock_symbol”和“stock_value”的<ORCL, 62>。与流元素相关联的时间戳是“timestamp_N+1”。因此, 连续事件流是事件的流, 每个事件具有相同的一系列属性。

[0038] 如上面所指出的, 流可以是CQL查询可以对其起作用的数据的主要源。流S可以是元素(s, T)的包(也被称为“多集”), 其中“s”处于S的模式中, 并且“T”在时域中。此外, 流元素可以是元组-时间戳对, 该元组-时间戳对可以被表示为带时间戳的元组插入的序列。换句话说, 流可以是带时间戳的元组的序列。在一些情况下, 可以存在具有相同时间戳的多于一个的元组。并且, 可以请求输入流的元组按照递增的时间戳次序到达系统。可替代地, 关系(也被称为“时变关系”, 并且不与“关系数据”(“关系数据”可以包括来自关系数据库的数据)混淆)可以是时域到模式R的元组的无界包的映射。在一些示例中, 关系可以是无序的、时变的元组包(即, 瞬时关系)。在一些情况下, 在每个时间实例, 关系可以是有界集合。它还可以被表示为可以包括插入、删除和/或更新以捕获关系的变化状态的带时间戳的元组的序列。类似于流, 关系可以具有关系的每个元组可以符合的固定模式。此外, 如本文所使用的, 连续查询可能一般能够处理流和/或关系的数据(即, 针对流和/或关系查询数据)。此外, 关系可以引用流的数据。

[0039] 在一些方面, CQL引擎可以包括完整的查询语言。由此, 用户可以根据查询来指定计算。此外, CQL引擎可以被设计为用于优化存储器、利用查询语言特征、运算符共享、丰富的模式匹配、丰富的语言构造等。此外, 在一些示例中, CQL引擎可以处理历史数据和流传输数据两者。例如, 用户可以设置查询以在加利福尼亚州销售额超过某个目标时发送提醒(alert)。因此, 在一些示例中, 该提醒可以至少部分地基于历史销售数据以及传入的实况(即, 实时)销售数据。

[0040] 在一些示例中, CQL引擎或下面描述的概念的其它特征可以被配置为以实时方式将历史上下文(即, 仓库数据)与传入数据组合。因此, 在一些情况下, 本公开可以描述数据库存储的信息和空中(in-flight)信息的边界。数据库存储的信息和空中信息两者都可以包括BI数据。由此, 在一些示例中, 数据库可以是BI服务器, 或者它可以是任何类型的数据库。此外, 在一些示例中, 本公开的特征可以使得能够实现上述特征而无需用户知道如何编程或以其它方式编写代码。换句话说, 可以以特征丰富的用户界面(UI)提供特征, 或以允许非开发人员实现历史数据与实时数据的组合的其它方式来提供特征。

[0041] 在一些实施例中, 当连续的(例如, CQL)查询被创建或注册时, 它可以经历解析和语义分析, 在该解析和语义分析结束时逻辑查询计划被创建。例如, 当启动CQL查询时, 通过发出“alter query<queryname>start(提醒查询<查询名称>开始)”DDL, 可以将逻辑查询计划转换为物理查询计划。在一个示例中, 物理查询计划可以被表示为物理运算符的有向无环图(DAG)。然后, 可以将物理运算符转换为执行运算符, 以获得针对该CQL查询的最终查询计划。到CQL引擎的传入事件到达(一个或多个)源运算符, 并最终以对这些事件执行其处理并产生适当的输出事件的方式和运算符一起向下游移动。

[0042] 事件处理应用

[0043] 在IT环境中, 原始基础设施和业务事件两者的数量和速度呈指数增长。无论是用

于金融服务的流传输股票数据、用于军事的流传输卫星数据还是用于运输和物流业务的实时车辆位置数据,多个行业中的公司都必须实时处理大量复杂数据。此外,移动设备的爆炸式增长和无处不在的高速连接性增加了移动数据的爆炸性增长。与此同时,对业务处理敏捷性和执行的需求也在增长。这两个趋势已经给组织带来了增加其能力以支持事件驱动的体系架构实现模式的压力。实时事件处理要求基础设施和应用开发环境两者针对事件处理要求来执行。这些要求常常包括从日常使用案例扩展到极高速度的数据和事件吞吐量的需求,可能还有以响应时间的微秒而不是秒为单位测得的延时。此外,事件处理应用必须常常检测这些事件的流中的复杂模式。

[0044] Oracle流分析(Oracle Stream Analytics)平台面向众多行业和功能领域。以下是一些用例:

[0045] 电信:执行服务攻击检测的实时呼叫细节(CDR)记录监视和分布式拒绝的能力。

[0046] 金融服务:利用存在于毫秒或微秒窗口中的套利机会的能力。执行金融证券交易的实时风险分析、监视和报告并计算外汇价格的能力。

[0047] 运输:在由于当地或目的地城市天气、地勤人员操作、机场安全等引起飞行差异的情况下创建乘客警报并检测行李位置的能力。

[0048] 公共部门/军事:检测分散的地理敌人信息、抽象它并解读敌人攻击的高概率的能力。提醒最合适的资源以应对紧急情况的能力。

[0049] 保险:学习和检测潜在欺诈性索赔的能力。

[0050] IT系统:实时检测失败的应用或服务器并触发纠正措施的能力。

[0051] 供应链和物流:实时跟踪货物并检测和报告到达的潜在延迟的能力。

[0052] 处理时间戳

[0053] 近年来,已经开发了可以以连续的方式对潜在无界的实时数据流执行查询的数据流管理系统(DSM)。在新的DSM中,一些系统采用基于微批处理的流处理,以便从单个框架提供批处理和流处理的组合。在Spark上运行的Spark Streaming就是这样的一个系统。但是,微批处理流处理不提供逐事件的处理,并为微批的所有事件分配相同的时间戳。因此,当窗口滑动持续时间应为毫秒或纳秒时,微批处理流处理不能检测到同一微批内的关系,或者无法为某些情况创建正确的结果。

[0054] 利用Oracle流分析对上述问题的一个解决方案是为微批中的每个事件生成唯一的时间戳。

[0055] 以下算法用于生成时间戳:

[0056] 1) 使用'batchTime'作为为批创建的事件的开始时间

[0057] 2) 批处理的结束-1作为心跳事件被发送

[0058] 3) 从批创建的元组的所有时间戳应该在范围(`currentBatchTime+batchDuration-2`)中

[0059] 4) 在创建元组的摄取(ingestion)阶段分配时间戳

[0060] 5) 存在四种不同的策略用于在`currentBatchTime`到`currentBatchTime+batchDuration`的范围内分配时间戳(挂钟推进、均匀分布、均匀分布的顶部离开(evenly distributed top off)和/或顺序的顶部离开(sequenced top off))

[0061] 利用该解决方案,以下是新的:

[0062] -对基于微批处理的流处理添加处理时间戳

[0063] -添加处理时间戳的心跳用于自动时间推进

[0064] 在使用本文描述的特征之前,基于微批的流处理不能为每个事件分配不同的时间戳,并且逐事件的处理是不可能的。缺乏逐事件的处理使得流处理和时间系列分析即使可能的话也非常困难。本发明通过添加逐事件的处理并且允许解决基于微批处理的流处理过去无法解决的用例而解决了基于微批处理的流处理的若干缺点。

[0065] 图1图示了用于处理微批处理系统内的时间戳的示例流程100。在一些示例中,作为驱动程序104的一部分执行的DAG调度程序102启动微批作业。该作业可以包括ETL变换,该变换使用具有CQLRDD 108作为子变换的ConvertToTupleRDD变换使用createTuples 106读取输入事件并创建元组。CQLRDD 108的计算方法将当前批处理时间设置为开始时间并调用父(ConvertToTupleRDD)变换的计算方法。在一些示例中,ConvertToTupleRDD的计算方法使用处理时间生成器110将处理时间戳设置到RDD中的元组。处理时间生成器110使用不同的分布策略将时间戳分布在批之间。例如,处理时间生成器110可以使用挂钟分布策略112、均匀分布的分布策略114、均匀分布的顶部离开分布策略116或顺序的顶部离开分布策略118。

[0066] 挂钟推进112:

[0067] • 使用当前时间与基准的差

[0068] • 纳入摄取延迟

[0069] • 溢出(overflow)保护

[0070] -将差上限限制到某个时间(比如批处理持续时间的1/2或2/3)

[0071] -在达到上限之后使用序列

[0072] 均匀分布114:

- 获取微批中的事件的数量

- 间隔 = (批处理持续时间) / (事件的数量)

均匀分布的顶部离开 116:

- 由用户通过提供'maxRate'来控制分布

- for (seq <- 0 until events.size)

- if (seq < maxRate)

[0073]

- timestamp = (currentBatchTime + (seq * baseInterval))
 << 3

- else

- timestamp = (currentBatchTime + (maxRate - 1) *
 baseInterval) << 3 + (maxRate - seq)

[0074] 顺序的顶部离开(简单序列) 118:

[0075] • 微批中的每个事件根据批处理时间获取序列号

[0076] • 单位为毫秒,通过在到达批处理的结束之后变成纳秒来进行溢出保护

[0077] 事件处理服务

[0078] 图2描绘了可以结合本公开的实施例的事件处理系统200的简化高级图。在实施例中,事件处理系统200可以表示由**Oracle®**管理的Oracle事件处理器(OEP)系统。事件处理系统200可以包括一个或多个事件源(604、206、208)、被配置为提供用于处理事件流的环境的事件处理服务(EPS)202(也称为CQ服务202),以及一个或多个事件汇点(sink)(610、212)。事件源生成由EPS 202接收的事件流。EPS 202可以从一个或多个事件源接收一个或多个事件流。

[0079] 例如,如图2中所示,EPS 202从事件源204接收第一输入事件流214,从事件源206接收第二输入事件流216,以及从事件源208接收第三事件流218。一个或多个事件处理应用(614、216和218)可以部署在EPS 202上并由EPS 202执行。由EPS 202执行的事件处理应用可以被配置为监听一个或多个输入事件流、基于从输入事件流中选择一个或多个事件作为值得注意的(notable)事件的处理逻辑来处理经由一个或多个事件流接收的事件。然后将值得注意的事件以一个或多个输出事件流的形式发送到一个或多个事件汇点(610、212)。例如,在图2中,EPS 202将第一输出事件流220输出到事件汇点210,并将第二输出事件流222输出到事件汇点212。在某些实施例中,事件源、事件处理应用和事件汇点彼此解耦,使得可以添加或移除这些部件中的任何部件而不会造成对其它部件的改变。

[0080] 在一个实施例中,EPS 202可以被实现为Java服务器,该Java服务器包括具有共享服务的轻量级Java应用容器,诸如基于Equinox OSGi的容器。在一些实施例中,例如通过使用JRockit Real Time,EPS 202可以支持针对处理事件的超高吞吐量和微秒延时。EPS 202还可以提供开发平台(例如,完整的实时端到端Java事件驱动的体系架构(EDA)开发平台),该开发平台包括用于开发事件处理应用的工具(例如,Oracle CEP可视化装置(Visualizer)和Oracle CEP IDE)。

[0081] 事件处理应用被配置为监听一个或多个输入事件流,执行用于从一个或多个输入事件流中选择一个或多个值得注意的事件的逻辑(例如,查询),并将所选择的值得注意的事件经由一个或多个输出事件流输出到一个或多个事件源。图2提供了用于一个这样的事件处理应用214的深入分析。如图2中所示,事件处理应用214被配置为监听输入事件流218,执行包括用于从输入事件218中选择一个或多个值得注意的事件的逻辑的连续查询230,并将所选择的值得注意的事件经由输出事件流222输出到事件汇点212。事件源的示例包括但不限于适配器(例如,JMS、HTTP和文件)、信道、处理器、表、高速缓存等。事件汇点的示例包括但不限于适配器(例如,JMS、HTTP和文件)、信道、处理器、高速缓存等。

[0082] 虽然图2中的事件处理应用214被示为监听一个输入流并经由一个输出流来输出所选择的事件,但这不旨在是限制性的。在替代实施例中,事件处理应用可以被配置为监听从一个或多个事件源接收的多个输入流,从被监视的流中选择事件,并将所选择的事件经由一个或多个输出事件流输出到一个或多个事件汇点。相同的查询可以与多于一个的事件汇点相关联并与不同类型的事件汇点相关联。

[0083] 由于其无界性质,经由事件流接收的数据量一般非常大。因此,为了查询目的而存储或存档所有数据一般是不切实际且不期望的。事件流的处理要求在EPS 202接收到事件

时实时地处理事件,而无需存储所有接收到的事件数据。因而,EPS 202提供了特殊的查询机制,该机制使得能够在EPS 202接收事件时执行事件的处理,而无需存储所有接收到的事件。

[0084] 事件驱动的应用是规则驱动的,并且这些规则可以以用于处理输入流的连续查询的形式来表达。连续查询可以包括指令(例如,逻辑),该指令识别要对接收到的事件执行的处理,包括将要选择什么事件作为值得注意的事件并作为查询处理的结果进行输出。连续查询可以持久保存到数据存储库并且用于处理事件的输入流并生成事件的输出流。连续查询通常执行过滤和聚合功能,以从输入事件流中发现和提取值得注意的事件。因此,输出事件流中的出站事件的数量一般远低于从中选择事件的输入事件流中的事件的数量。

[0085] 与在有限数据集上运行一次的SQL查询不同,每次在特定事件流中接收到事件时,都可以执行已经由具有EPS 202的应用向该事件流注册的连续查询。作为连续查询执行的一部分,EPS 202基于由连续查询指定的指令来评估接收到的事件,以确定是否要选择一个或多个事件作为值得注意的事件并且作为连续查询执行的结果进行输出。

[0086] 可以使用不同语言对连续查询进行编程。在某些实施例中,可以使用由Oracle公司提供并且由Oracle的复杂事件处理(CEP)产品供应使用的CQL对连续查询进行配置。Oracle的CQL是可以用于对可以针对事件流执行的查询(称为CQL查询)进行编程的声明性语言。在某些实施例中,CQL基于SQL,其中添加了支持流传输事件数据处理的构造。

[0087] 应该认识到的是,图2中描绘的系统200可以具有除图2中描绘的那些部件之外的其它部件。另外,图2中示出的实施例仅是可以结合本公开实施例的系统的一个示例。在一些其它实施例中,系统200可以具有比图2中所示更多或更少的部件、可以组合两个或更多个部件,或者可以具有不同的部件布置或配置。系统200可以是各种类型的,包括服务提供商计算机、个人计算机、便携式设备(例如,移动电话或设备)、工作站、网络计算机、大型机、信息亭、服务器或任何其它数据处理系统。

[0088] 在一些示例中,微批处理流处理不提供逐事件的处理,并且为微批内的所有事件分配相同的时间戳。因此,微批处理流处理可能不能检测到同一微批内的事件的关系。

[0089] 对于微批内的所有事件具有相同的时间戳,更严重的问题是除非窗口滑动持续时间是毫秒或纳秒,否则可能无法创建正确的结果。图3示出了这样的情况:用例是对三秒间隔中具有三个或更多个交易的信用卡做标志。批间隔可以是每个一秒,并且假设事件到达率是2个事件/秒/批。处理时间戳是在从到系统的摄取创建元组时分配给元组的时间戳。每个时间戳应该单调增加,并且不应该与先前的时间戳重叠或重复。通过针对每批使用相同的时间戳,系统将看到两个窗口的以下事件:

[0090] 时间 t_0 时的窗口: { (E1,CC1), (E2,CC2), (E3,CC3), (E4,CC4), (E5,CC5), (E6,CC6) }

[0091] 时间 t_1 时的窗口: { (E3,CC3), (E4,CC4), (E5,CC5), (E6,CC2), (E7,CC2), (E8,CC6) }

[0092] 假设窗口长度为三秒,时间 t_0 时的窗口将从 t_0 变为 t_3 ,而时间 t_1 时的窗口将从 t_1 变为 t_4 。因此,仅查看CC2的事件, $t_{0.5}$: (E2,CC2), $t_{2.5}$: (E6,CC2), t_3 : (E7,CC2), 在3秒窗口内存在三个交易,其中具有0.5秒的滑动,但是上面对于每批使用相同时间戳的系统错过了它。换句话说,该系统将批处理时间 t_4 分配给事件E7和E8两者,并且默认情况下使用批间隔

进行滑动。但是,通过滑动1秒,该系统将错过在 $t=3.5$ 时CC2有三个交易的事实。

[0093] 用于自动时间推进的心跳事件

[0094] 为了提供逐事件的处理,已经添加了处理时间戳和事件时间戳。在逐事件的处理系统内,系统需要不断地推进时间。否则,我们可能遇到以下问题:

[0095] -延迟的输出

[0096] -窗口不会到期

[0097] -不会从丢失事件检测模式匹配而发出持续时间

[0098] 为了说明该问题,让我们举个例子,其中我们具有在两个执行器上运行的两个cq1引擎上运行的范围窗口。为了简单起见,将假设每个批包含一条记录。

[0099] 查询:SELECT*FROM S[RANGE 3]

[0100] 输入:

[0101] 1001:1

[0102] 1002:2

[0103] 1003:3

[0104] 1004:4

[0105] 1005:5

[0106] 假设1001和1002去往作为处理分区-1的CQL引擎1。因此,CQL引擎1的窗口将有两条记录:

[0107] $\{(1001,1), (1002,2)\}$

[0108] 并且,其余的三个事件去往作为处理分区-2的CQL引擎2。因此,CQL引擎2的窗口将有三条记录:

[0109] $\{(1003,3), (1004,4), (1005,5)\}$

[0110] 因此,在这种应用时间戳的情况下,当(1003,3)到达时,将评估节点1处的空CQL RDD,因为(1003,3)属于在节点2上运行的分区。现在关于其它三个输入,时间戳被移动到1005。因此,理想情况下,CQL引擎1中的窗口的事件应该在窗口尺寸为3时到期。要使这些事件到期,CQL引擎1可能需要来自CQL RDD计算的心跳。但是,可能不知道/不清楚谁将把1003的心跳传播到CQL引擎1。

[0111] 在一些示例中,一种解决方案是从重新分区插入心跳,然后跟踪分区的最新事件时间戳。另一个实施例是通过时间戳对输出事件进行排序,并将最新的心跳传播到所有分区。例如:

[0112] 来自自定义分区程序(partitioner)的心跳可以被插入:

[0113] 1001,1p1

[0114] 1002,2p1

[0115] 1003.3p2

[0116] 1004,4p2

[0117] 1005,5p2

[0118] $\langle 1005, HT_p1 \rangle$ (这是发送到分区1的心跳)

[0119] 然后:

[0120] 从CQL RDD,可以发送快照信息,并且可以添加来自分区的最大事件时间。在一些

情况下,批的最大值(最大事件时间)作为getCQLEngine请求的一部分被返回给CQL RDD。

[0121] 批1

[0122] 1001,1p1

[0123] 1002,2p1

[0124] 将1002作为snapshotInfo的一部分发送到CQLEngineTracker

[0125] 1003,3p2

[0126] 1004,4p2

[0127] 1005,5p2

[0128] 将1005作为snapshotInfo的一部分发送到CQLEngineTracker

[0129] CQLEngineTracker

[0130] p1的最大事件时间戳-1002

[0131] p2的最大事件时间戳-1005

[0132] 批的最大事件时间戳=1005

[0133] 批2

[0134] 因为 $1002 < 1005$,因此getCQLEngine(p1) 返回1005

[0135] 在输入元组之前,(1005)的心跳被发送到CQLEngine

[0136] p1的最大事件时间戳-1005

[0137] 因为 $1005 = 1005$,因此getCQLEngine(p2) 不包括心跳

[0138] 图4图示了用于生成心跳的流程400。在从ConvertToTupleRDD变换创建处理时间戳时生成初始心跳。对于每批,我们在该批的处理时间戳持续时间内具有最高时间戳的批的结束处附加一个心跳事件。在批为空并且该批没有输入事件的情况下,我们仍然插入一个心跳事件以便确保时间推进。

[0139] 使用以下算法根据分区逻辑来传播心跳:

[0140] 1. 在402处使用sortBykey变换在RDD中对元组进行排序,其中时间戳作为键(key)。

[0141] 2. 在404处使用HeartbeatPropagateRDD将心跳复制到分区。

[0142] 3. 在406处使用partitionBy变换进行实际分区并且完成心跳到分区的实际传播。

[0143] 实际的HeartbeatPropagation算法按如下进行工作:

[0144] 1. 除了最后一个心跳之外,跳过所有心跳

[0145] 2. 移除重复的心跳时间戳

[0146] 3. 通过具有携带分区号的HBPartitionValue将心跳复制到每个分区。

[0147] 由于分区是通过按字段的某种分组完成的,因此心跳事件不能使用这样的准则来分区,因为心跳事件仅携带时间戳。HBPartitionValue是其中携带分区信息的特殊标记对象。当分区程序从心跳事件中看到HBPartitionValue时,可以正确地提取分区值,并将其用于把心跳事件设置到分区。

[0148] 示例方法

[0149] 图5是根据本公开的实施例的描述用于处理时间戳的一组操作的处理500的示例流程图。在某些实施例中,处理500可以由驱动程序(诸如图1的驱动程序104)、发生器(诸如图1的处理时间发生器110)和/或服务(诸如图2的事件处理服务202)来执行,它们中的每一

个可以是分布式事件处理系统的一部分。图5的处理描述了一种技术,通过该技术可以对一批事件中的数据加时间戳。图5中描绘的特定系列的处理步骤不旨在是限制性的。替代实施例可以在各种布置和组合中具有比图5中所示的步骤更多或更少的步骤。

[0150] 在某些实施例中,处理500在502处通过将流处理器配置为对来自流源的传入事件进行微批处理而开始。在504处,该处理包括为传入事件的微批生成单个时间戳。在某些示例中,处理500可以包括在506处从流源接收传入事件的微批,并且在508处将单个时间戳分配给微批的每个事件。在510处,处理500包括至少部分地基于算法为微批的每个相应事件生成单独的时间戳值。此外,在512处,处理500包括为微批的每个相应事件分配单独的时间戳值中的个体时间戳值。

[0151] 图6是描述用于在微批处理的数据中生成和插入心跳事件的一组操作的处理600的示例流程图。在实施例中,处理600描述了图5的处理500的操作的附加细节。在某些示例中,处理600在602处通过将每个事件分配给一个或多个连续查询语言引擎分区而开始。在604处,该处理包括识别一个或多个连续查询语言引擎分区中的第一分区。在606处,该处理包括确定第一分区具有比批中被分配给第一分区的多个事件中的事件数量更大的窗口尺寸。在608处,该处理包括为第一分区生成心跳事件。在610处,该处理包括将心跳事件插入到第一分区中。

[0152] 说明性系统

[0153] 图7-图9图示了用于实现根据各种实施例的本公开的方面的示例环境的方面。图7描绘了用于实现本公开实施例的分布式系统700的简化图。在所示实施例中,分布式系统700包括一个或多个客户端计算设备702、704、706和708,这些客户端计算设备被配置为通过一个或多个网络710执行和操作客户端应用,诸如web浏览器、专有客户端(例如,Oracle Forms)等。服务器712可以经由网络710与远程客户端计算设备702、704、706和708通信地耦合。

[0154] 在各种实施例中,服务器712可以适于运行一个或多个服务或软件应用,诸如提供事件处理服务的服务和应用。在某些实施例中,服务器712还可以提供其它服务,或者软件应用可以包括非虚拟环境和虚拟环境。在一些实施例中,这些服务可以被供应为基于web的服务或云服务,或者在软件即服务(SaaS)模型下被供应给客户端计算设备702、704、706和/或708的用户。操作客户端计算设备702、704、706和/或708的用户进而可以利用一个或多个客户端应用与服务器712进行交互以利用由这些部件提供的服务。

[0155] 在图7描绘的配置中,系统700的软件部件718、720和722被示出为在服务器712上实现。在其它实施例中,系统700的一个或多个部件和/或由这些部件提供的服务可以也可以由客户端计算设备702、704、706和/或708中的一个或多个来实现。然后,操作客户端计算设备的用户可以利用一个或多个客户端应用来使用由这些部件提供的服务。这些部件可以用硬件、固件、软件或其组合来实现。应该理解的是,各种不同的系统配置是可能的,其可以与分布式系统700不同。因此,图7中所示的实施例是用于实现实施例系统的分布式系统的一个示例,而不旨在是限制性的。

[0156] 客户端计算设备702、704、706和/或708可以包括各种类型的计算系统。例如,客户端设备可以包括便携式手持设备(例如,iPhone®、蜂窝电话、iPad®、计算平板、个人数字助理(PDA))或可穿戴设备(例如,Google Glass®头戴式显示器),运行诸如

Microsoft Windows **Mobile®**的软件和/或诸如iOS、Windows Phone、Android、BlackBerry 7、Palm OS等的各种移动操作系统。设备可以支持各种应用(诸如各种互联网相关的应用、电子邮件、短消息服务(SMS)应用),并且可以使用各种其它通信协议。客户端计算设备还可以包括通用个人计算机,包括例如运行各种版本的Microsoft **Windows®**、Apple **Macintosh®**和/或Linux操作系统的膝上型计算机和/或个人计算机。客户端计算设备可以是运行各种商业可用的**UNIX®**或类UNIX操作系统(包括但不限于诸如例如Google Chrome OS的各种GNU/Linux操作系统)中的任何操作系统的工作站计算机。客户端计算设备还可以包括能够通过(一个或多个)网络710进行通信的电子设备,诸如瘦客户端计算机、启用互联网的游戏系统(例如,具有或不具有**Kinect®**手势输入设备的Microsoft **Xbox®**游戏控制台)和/或个人消息传送设备。

[0157] 虽然图7中的分布式系统700被示出具有四个客户端计算设备,但是可以支持任何数量的客户端计算设备。其它设备(诸如具有传感器的设备等)可以与服务器712进行交互。

[0158] 分布式系统700中的(一个或多个)网络710可以是对本领域技术人员熟悉的可以利用各种可用协议中的任何协议来支持数据通信的任何类型的网络,其中各种可用协议包括但不限于TCP/IP(传输控制协议/互联网协议)、SNA(系统网络体系架构)、IPX(互联网分组交换)、AppleTalk等。仅仅作为示例,(一个或多个)网络710可以是局域网(LAN)、基于以太网的网络、令牌环、广域网、互联网、虚拟网络、虚拟专用网络(VPN)、内联网、外联网、公共交换电话网络(PSTN)、红外网络、无线网络(例如,在任何电气和电子协会(IEEE) 702.11协议套件、**Bluetooth®**、和/或任何其它无线协议下操作的网络)和/或这些网络和/或其它网络的任何组合。

[0159] 服务器712可以由一个或多个通用计算机、专用服务器计算机(作为示例,包括PC(个人计算机)服务器、**UNIX®**服务器、中档服务器、大型计算机、机架安装的服务器等)、服务器场、服务器集群或任何其它适当的布置和/或组合组成。服务器712可以包括运行虚拟操作系统的一个或多个虚拟机,或涉及虚拟化的其它计算体系架构。一个或多个灵活的逻辑存储设备池可以被虚拟化,以维护服务器的虚拟存储设备。虚拟网络可以由服务器712使用软件定义的联网来控制。在各种实施例中,服务器712可以适于运行在前述公开内容中描述的一个或多个服务或软件应用。例如,服务器712可以与根据本公开的实施例的用于如上所述执行处理的服务器相对应。

[0160] 服务器712可以运行包括以上讨论的任何操作系统的操作系统,以及任何商业可用的服务器操作系统。服务器712还可以运行任何各种附加的服务器应用和/或中间层应用,包括HTTP(超文本传输协议)服务器、FTP(文件传输协议)服务器、CGI(公共网关接口)服务器、**JAVA®**服务器、数据库服务器等。示例性数据库服务器包括但不限于来自Oracle、Microsoft、Sybase、IBM(国际商业机器)等的商业可用的数据库服务器。

[0161] 在一些实现中,服务器712可以包括一个或多个应用,以分析和整合从客户端计算设备702、704、706和708的用户接收到的数据馈送和/或事件更新。作为示例,数据馈送和/或事件更新可以包括但不限于从一个或多个第三方信息源和连续数据流接收到的**Twitter®**馈送、**Facebook®**更新或实时更新,其可以包括与传感器数据应用、金融报价机、网络性能测量工具(例如,网络监视和流量管理应用)、点击流分析工具、汽车流量监

视等相关的实时事件。服务器712还可以包括经由客户端计算设备702、704、706和708的一个或多个显示设备来显示数据馈送和/或实时事件的一个或多个应用。

[0162] 分布式系统700还可以包括一个或多个数据库714和716。这些数据库可以提供用于存储信息(诸如事件信息和由本公开实施例使用的其它信息)的机制。数据库714和716可以驻留在各种位置中。例如,数据库714和716中的一个或多个可以驻留在服务器712本地(和/或驻留在服务器712中)的非瞬态存储介质上。可替代地,数据库714和716可以远离服务器712并经由基于网络的连接或专用连接与服务器712进行通信。在一组实施例中,数据库714和716可以驻留在存储区域网络(SAN)中。类似地,用于执行归属于服务器712的功能的任何必要文件可以适当地本地存储在服务器712上和/或远程存储。在一组实施例中,数据库714和716可以包括适于响应于SQL格式的命令来存储、更新和检索数据的关系数据库,诸如由Oracle提供的数据库。

[0163] 一些附图中描绘的系统可以以各种配置来提供。在一些实施例中,系统可以被配置为分布式系统,其中系统的一个或多个部件跨一个或多个网络分布在一个或多个云基础设施系统中。

[0164] 云基础设施系统是一个或多个服务器计算设备,网络设备和/或存储设备的集合。这些资源可以由云服务提供商划分,并以某种方式拨给其客户。例如,云服务提供商(诸如加利福尼亚州Redwood Shores的Oracle公司)可以供应各种类型的云服务,包括但不限于在软件即服务(SaaS)类别下提供的一个或多个服务、在平台即服务(PaaS)类别下提供的服务、在基础设施即服务(IaaS)类别下提供的服务,或包括混合服务的其它类别的服务。SaaS服务的示例包括但不限于构建和递送按需应用套件(诸如Oracle融合(Fusion)应用)的能力。SaaS服务使客户能够利用在云基础设施系统上执行的应用,而无需客户购买应用的软件。PaaS服务的示例包括但不限于使组织(诸如Oracle)能够在共享的公共体系架构上整合现有应用的服务,以及构建充分利用由平台提供的共享服务(诸如Oracle Java云服务(JCS)、Oracle数据库云服务(DBCS)等)的新应用的能力。IaaS服务通常有助于管理和控制底层计算资源,诸如存储装置、网络和其它基本计算资源,用于让客户利用由SaaS平台和PaaS平台提供的服务。

[0165] 图8是根据本公开实施例的系统环境800的一个或多个部件的简化框图,通过该系统环境800,可以供应由实施例系统的一个或多个部件提供的服务作为云服务。在所示实施例中,系统环境800包括一个或多个客户端计算设备804、806和808,其可以由用户用于与提供云服务的云基础设施系统802进行交互。客户端计算设备可以被配置为操作可以由客户端计算设备的用户用来与云基础设施系统802进行交互以使用由云基础设施系统802提供的服务的客户端应用,诸如web浏览器、专有客户端应用(例如,Oracle Forms)或某个其它应用。

[0166] 应该认识到的是,图中描绘的云基础设施系统802可以具有除了所描绘的那些部件之外的其它部件。另外,图中所示的实施例仅是可以结合本公开实施例的云基础设施系统的一个示例。在一些其它实施例中,云基础设施系统802可以具有比图中所示更多或更少的部件、可以组合两个或更多个部件、或者可以具有不同的部件布置或配置。

[0167] 客户端计算设备804、806和808可以是与上面针对502、504、506和508描述的设备类似的设备。

[0168] 虽然示出了具有三个客户端计算设备的示例系统环境800,但是可以支持任何数量的客户端计算设备。诸如具有传感器的设备等的其它设备可以与云基础设施系统802进行交互。

[0169] (一个或多个)网络810可以促进客户端804、806和808与云基础设施系统802之间的数据的交换和通信。每个网络可以是本领域技术人员熟悉的、可以使用各种商业可用的协议(包括上面针对(一个或多个)网络810描述的那些协议)中的任何协议来支持数据通信的任何类型的网络。

[0170] 云基础设施系统802可以包括一个或多个计算机和/或服务器,其可以包括上面针对服务器812描述的那些计算机和/或服务器。

[0171] 在某些实施例中,由云基础设施系统提供的服务可以包括按需对云基础设施系统的用户可用的大量服务,诸如在线数据存储和备份解决方案、基于Web的电子邮件服务、托管的办公套件和文档协作服务、数据库处理、受管理的技术支持服务等。由云基础设施系统提供的服务可以动态扩展以满足其用户的需求。由云基础设施系统提供的服务的具体实例化在本文中被称为“服务实例”。一般而言,经由通信网络(诸如互联网)从云服务提供商的系统使得对用户可用的任何服务被称为“云服务”。通常,在公共云环境中,构成云服务提供商的系统的服务器和系统与客户自己的本地服务器和系统不同。例如,云服务提供商的系统可以托管应用,并且用户可以经由诸如互联网的通信网络按需订购和使用应用。

[0172] 在一些示例中,计算机网络云基础设施中的服务可以包括对存储装置、托管的数据库、托管的web服务器、软件应用或者由云供应商向用户提供的其它服务的受保护的计算机网络访问,或者如本领域中另外已知的。例如,服务可以包括通过互联网对云上的远程存储的受密码保护的访问。作为另一个示例,服务可以包括基于web服务的托管的关系数据库和脚本语言中间件引擎,用于由联网的开发人员私人使用。作为另一个示例,服务可以包括对在云供应商的网站上托管的电子邮件软件应用的访问。

[0173] 在某些实施例中,云基础设施系统802可以包括以自助服务、基于订阅、弹性可扩展、可靠、高度可用和安全的方式交付给客户的应用套件、中间件和数据库服务产品。这种云基础设施系统的示例是由本受让人提供的Oracle公共云(Oracle Public Cloud)。

[0174] 在各种实施例中,云基础设施系统802可以适于自动地供应、管理和跟踪客户对由云基础设施系统802供应的服务的订阅。云基础设施系统802可以经由不同的部署模型来提供云服务。例如,服务可以在公共云模型下提供,其中云基础设施系统802由销售云服务的组织所拥有(例如,由Oracle所拥有)并且使服务对一般公众或不同的工业企业可用。作为另一个示例,服务可以在私有云模型下提供,其中云基础设施系统802仅针对单个组织操作,并且可以为组织内的一个或多个实体提供服务。云服务还可以在社区云模型下提供,其中云基础设施系统802和由云基础设施系统802提供的服务由相关社区中的若干个组织共享。云服务还可以在混合云模型下提供,混合云模型是两个或更多个不同模型的组合。

[0175] 在一些实施例中,由云基础设施系统802提供的服务可以包括在软件即服务(SaaS)类别、平台即服务(PaaS)类别、基础设施即服务(IaaS)类别、或包括混合服务的服务的其它类别下提供的一个或多个服务。客户经由订阅订单可以订购由云基础设施系统802提供的一个或多个服务。云基础设施系统802然后执行处理,以提供客户的订阅订单中的服务。

[0176] 在一些实施例中,由云基础设施系统802提供的服务可以包括但不限于应用服务、平台服务和基础设施服务。在一些示例中,应用服务可以由云基础设施系统经由SaaS平台提供。SaaS平台可以被配置为提供属于SaaS类别下的云服务。例如,SaaS平台可以提供在集成的开发和部署平台上构建和交付点播应用套件的能力。SaaS平台可以管理和控制用于提供SaaS服务的底层软件和基础设施。通过利用由SaaS平台提供的服务,客户可以利用在云基础设施系统上执行的应用。客户可以获取应用服务,而无需客户购买单独的许可证和支持。可以提供各种不同的SaaS服务。示例包括但不限于为大型组织提供用于销售绩效管理、企业集成和业务灵活性的解决方案的服务。

[0177] 在一些实施例中,平台服务可以由云基础设施系统经由PaaS平台提供。PaaS平台可以被配置为提供属于PaaS类别的云服务。平台服务的示例可以包括但不限于使组织(诸如Oracle)能够在共享的共同体系统架构上整合现有应用的服务,以及构建充分利用由平台提供的共享服务的新应用的能力。PaaS平台可以管理和控制用于提供PaaS服务的底层软件和基础设施。客户可以获取由云基础设施系统提供的PaaS服务,而无需客户购买单独的许可证和支持。平台服务的示例包括但不限于Oracle Java云服务(JCS)、Oracle数据库云服务(DBCS)等。

[0178] 通过利用由PaaS平台提供的服务,客户可以采用由云基础设施系统支持的编程语言和工具,并且还控制所部署的服务。在一些实施例中,由云基础设施系统提供的平台服务可以包括数据库云服务、中间件云服务(例如,Oracle融合中间件服务)和Java云服务。在一个实施例中,数据库云服务可以支持共享服务部署模型,该模型使得组织能够汇集数据库资源并且以数据库云的形式向客户供应数据库即服务。中间件云服务可以为客户提供开发和部署各种业务应用的平台,以及Java云服务可以在云基础设施系统中为客户提供部署Java应用的平台。

[0179] 可以由云基础设施系统中的IaaS平台提供各种不同的基础设施服务。基础设施服务促进底层计算资源(诸如存储装置、网络和其它基本计算资源)的管理和控制,以便客户利用由SaaS平台和PaaS平台提供的服务。

[0180] 在某些实施例中,云基础设施系统802还可以包括基础设施资源830,用于提供用来向云基础设施系统的客户提供各种服务的资源。在一个实施例中,基础设施资源830可以包括执行由PaaS平台和SaaS平台提供的服务的硬件(诸如服务器、存储装置和联网资源)的预先集成和优化的组合。

[0181] 在一些实施例中,云基础设施系统802中的资源可以由多个用户共享并且按需动态地重新分派。此外,资源可以被分派给在不同时区中的用户。例如,云基础设施系统830可以使得在指定的小时数内第一时区中的第一用户集合能够利用云基础设施系统的资源,然后使得能够将相同资源重新分派给位于不同时区中的另一个用户集合,从而最大化资源的利用率。

[0182] 在某些实施例中,可以提供由云基础设施系统802的不同部件或模块以及由云基础设施系统802提供的服务共享的多个内部共享服务832。这些内部共享服务可以包括但不限于:安全和身份服务、集成服务、企业储存库服务、企业管理器服务、病毒扫描和白名单服务、高可用性、备份和恢复服务、用于启用云支持的服务、电子邮件服务、通知服务、文件传输服务等。

[0183] 在某些实施例中,云基础设施系统802可以在云基础设施系统中提供云服务(例如,SaaS、PaaS和IaaS服务)的综合管理。在一个实施例中,云管理功能可以包括用于供应、管理和跟踪由云基础设施系统802等接收到的客户的订阅的能力。

[0184] 在一个实施例中,如图中所描绘的,云管理功能可以由诸如订单管理模块820、订单编排(orchestration)模块822、订单供应模块824、订单管理和监视模块826以及身份管理模块828的一个或多个模块提供。这些模块可以包括一个或多个计算机和/或服务器,或可以使用一个或多个计算机和/或服务器来提供,该一个或多个计算机和/或服务器可以是通用计算机、专用服务器计算机、服务器场,服务器集群或任何其它适当的布置和/或组合。

[0185] 在示例操作834中,使用客户端设备(诸如客户端设备804、806或808)的客户可以通过请求由云基础设施系统802提供的一个或多个服务并且对由云基础设施系统802供应的一个或多个服务的订阅下订单来与云基础设施系统802进行交互。在某些实施例中,客户可以访问云用户界面(UI)(云UI 812、云UI 814和/或云UI 816)并经由这些UI下订阅订单。响应于客户下订单而由云基础设施系统802接收到的订单信息可以包括识别客户和客户打算订阅的由云基础设施系统802供应的一个或多个服务的信息。

[0186] 在客户已经下订单之后,经由云UI 812、814和/或816来接收订单信息。

[0187] 在操作836处,订单存储在订单数据库818中。订单数据库818可以是由云基础设施系统818操作并且与其它系统元件一起操作的若干数据库中的一个数据库。

[0188] 在操作838处,订单信息被转发到订单管理模块820。在一些实例中,订单管理模块820可以被配置为执行与订单相关的计费和记帐功能,诸如验证订单,并且在验证后预订订单。

[0189] 在操作840处,将关于订单的信息传送到订单编排模块822。订单编排模块822可以利用订单信息来编排针对客户下的订单的服务和资源的供应。在一些实例中,订单编排模块822可以使用订单供应模块824的服务来编排资源的供应以支持订阅的服务。

[0190] 在某些实施例中,订单编排模块822使得能够管理与每个订单相关联的业务处理并且应用业务逻辑以确定订单是否应该前进到供应。在操作842处,在接收到对新订阅的订单后,订单编排模块822向订单供应模块824发送对分派资源并配置完成订阅的订单所需的那些资源的请求。订单供应模块824使得能够为客户订阅的服务分派资源。订单供应模块824提供由云基础设施系统800提供的云服务与用于供应用于提供所请求服务的资源的物理实现层之间的抽象级别。因此,订单编排模块822可以与实现细节隔离,诸如服务和资源实际上是实时供应的还是预先供应并且仅在请求后进行分派/分配。

[0191] 在操作844处,一旦供应了服务和资源,就可以通过云基础设施系统802的订单提供模块824向客户端设备804、806和/或808上的客户发送所提供服务的通知。在操作846处,客户的订阅订单可以由订单管理和监视模块826管理和跟踪。在一些情况下,订单管理和监视模块826可以被配置为收集订阅订单中的服务的使用统计数据,诸如所使用的存储量、传送的数据量、用户数量、系统运行时间和系统停机时间。

[0192] 在某些实施例中,云基础设施系统800可以包括身份管理模块828。身份管理模块828可以被配置为在云基础设施系统800中提供身份服务,诸如访问管理和授权服务。在一些实施例中,身份管理模块828可以控制关于希望利用由云基础设施系统802提供的服务的客户的信息。这种信息可以包括认证这些客户的身份的信息和描述这些客户被授权相对于

各种系统资源(例如,文件、目录、应用、通信端口、存储器段等)执行的动作的信息。身份管理模块828还可以包括关于每个客户的描述性信息以及关于可以如何以及由谁来访问和修改该描述性信息的管理。

[0193] 图9图示了可以用于实现本公开实施例的示例计算机系统900。在一些实施例中,计算机系统900可以用于实现上述各种服务器和计算机系统上的任何服务器和计算机系统。如图9所示,计算机系统900包括各种子系统,包括经由总线子系统902与多个外围子系统进行通信的处理单元904。这些外围子系统可以包括处理加速单元906、I/O子系统908、存储子系统918和通信子系统924。存储子系统918可以包括有形计算机可读存储介质922和系统存储器910。

[0194] 总线子系统902提供用于使计算机系统900的各种部件和子系统按照期望彼此进行通信的机制。虽然总线子系统902被示意性地示为单条总线,但是总线子系统的替代实施例可以利用多条总线。总线子系统902可以是若干种类型的总线结构中的任何类型的总线结构,包括存储器总线或存储器控制器、外围总线和使用各种总线体系架构中的任何总线体系架构的局部总线。例如,此类体系架构可以包括工业标准体系架构 (ISA) 总线、微通道体系架构 (MCA) 总线、增强型ISA (EISA) 总线、视频电子标准协会 (VESA) 局部总线和外围部件互连 (PCI) 总线(其可以实现为根据IEEE P1386.1标准制造的夹层 (Mezzanine) 总线) 等。

[0195] 处理子系统904控制计算机系统900的操作并且可以包括一个或多个处理单元932、934等。处理单元可以包括一个或多个处理器,包括单核或多核处理器、处理器的一个或多个核、或其组合。在一些实施例中,处理子系统904可以包括一个或多个专用协处理器,诸如图形处理器、数字信号处理器 (DSP) 等。在一些实施例中,处理子系统904的处理单元中的一些或全部可以使用定制电路来实现,诸如专用集成电路 (ASIC) 或现场可编程门阵列 (FPGA)。

[0196] 在一些实施例中,处理子系统904中的处理单元可以执行存储在系统存储器910中或计算机可读存储介质922上的指令。在各种实施例中,处理单元可以执行各种程序或代码指令,并且可以维护多个并发执行的程序或进程。在任何给定的时间,要执行的程序代码中的一些或全部可以驻留在系统存储器910中和/或计算机可读存储介质910上,潜在地包括在一个或多个存储设备上。通过适当的编程,处理子系统904可以提供上述各种功能,用于响应于使用模式而动态地修改文档(例如,网页)。

[0197] 在某些实施例中,可以提供处理加速单元906,用于执行定制的处理或用于卸载由处理子系统904执行的一些处理,以便加速由计算机系统900执行的总体处理。

[0198] I/O子系统908可以包括用于向计算机系统900输入信息和/或用于从或经由计算机系统900输出信息的设备和机制。一般而言,术语“输入设备”的使用旨在包括用于向计算机系统900输入信息的所有可能类型的设备和机制。用户接口输入设备可以包括例如键盘、诸如鼠标或轨迹球的指向设备、结合到显示器中的触摸板或触摸屏、滚轮、点击轮、拨盘、按钮、开关、键板、具有语音命令识别系统的音频输入设备、麦克风以及其它类型的输入设备。用户接口输入设备也可以包括使得用户能够控制输入设备并与输入设备进行交互的诸如Microsoft **Kinect**®运动传感器的运动感测和/或姿势识别设备、Microsoft **Xbox**® 360游戏控制器、提供用于接收使用姿势和口语命令的输入的接口的设备。用户接口输入设备也可以包括眼睛姿势识别设备,诸如从用户检测眼睛活动(例如,当拍摄照片和/或进行

菜单选择时的“眨眼”)并将眼睛姿势转换为进入输入设备(例如,Google **Glass**®)中的输入的Google **Glass**®眨眼检测器。此外,用户接口输入设备可以包括使得用户能够通过语音命令与语音识别系统(例如,**Siri**®导航器)进行交互的语音识别感测设备。

[0199] 用户接口输入设备的其它示例包括但不限于三维(3D)鼠标、操纵杆或指向杆、游戏板和图形平板、以及音频/视频设备,诸如扬声器、数字相机、数字摄录机、便携式媒体播放器、网络摄像机、图像扫描仪、指纹扫描仪、条形码读取器3D扫描仪、3D打印机、激光测距仪、以及眼睛注视跟踪设备。此外,用户接口输入设备可以包括例如医疗成像输入设备,诸如计算机断层摄影、磁共振成像、位置发射断层摄影、医疗超声检查设备。用户接口输入设备也可以包括例如音频输入设备,诸如MIDI键盘、数字乐器等。

[0200] 用户接口输出设备可以包括显示子系统、指示器灯或诸如音频输出设备的非可视显示器等。显示子系统可以是阴极射线管(CRT)、诸如使用液晶显示器(LCD)或等离子体显示器的平板设备、投影设备、触摸屏等。一般而言,术语“输出设备”的使用旨在包括用于从计算机系统900向用户或其它计算机输出信息的所有可能类型的设备和机制。例如,用户接口输出设备可以包括但不限于,可视地传达文本、图形和音频/视频信息的各种显示设备,诸如监视器、打印机、扬声器、耳机、汽车导航系统、绘图仪、语音输出设备和调制解调器。

[0201] 存储子系统918提供用于存储由计算机系统900使用的信息的储存库或数据存储库。存储子系统918提供有形非瞬态计算机可读存储介质,用于存储提供一些实施例的功能的基本编程和数据构造。当由处理子系统904执行时提供上述功能的软件(程序、代码模块、指令)可以存储在存储子系统918中。软件可以由处理子系统904的一个或多个处理单元执行。存储子系统918也可以提供用于存储根据本公开使用的数据的储存库。

[0202] 存储子系统918可以包括一个或多个非瞬态存储器设备,包括易失性和非易失性存储器设备。如图9所示,存储子系统918包括系统存储器910和计算机可读存储介质922。系统存储器910可以包括多个存储器,包括用于在程序执行期间存储指令和数据的易失性主随机存取存储器(RAM)和其中存储固定指令的非易失性只读存储器(ROM)或闪存存储器。在一些实现中,包含帮助在诸如启动期间在计算机系统900内的元件之间传送信息的基本例程的基本输入/输出系统(BIOS)通常可以存储在ROM中。RAM通常包含当前由处理子系统904操作和执行的数据和/或程序模块。在一些实现中,系统存储器910可以包括多个不同类型的存储器,诸如静态随机存取存储器(SRAM)或动态随机存取存储器(DRAM)。

[0203] 作为示例而非限制,如在图9中所绘出的,系统存储器910可以存储应用程序912,其可以包括客户端应用、Web浏览器、中间层应用、关系数据库管理系统(RDBMS)等、程序数据914和操作系统916。作为示例,操作系统916可以包括各种版本的Microsoft **Windows**®、Apple **Macintosh**®和/或Linux操作系统、各种商业可用的**UNIX**®或类UNIX操作系统(包括但不限于各种GNU/Linux操作系统、Google **Chrome**® OS等)和/或诸如iOS、**Windows**® Phone、**Android**® OS、**BlackBerry**® 70S和**Palm**® OS操作系统的移动操作系统。

[0204] 计算机可读存储介质922可以存储提供一些实施例的功能的编程和数据构造。当由处理子系统904执行时使处理器提供上述功能的软件(程序、代码模块、指令)可以存储在存储子系统918中。作为示例,计算机可读存储介质922可以包括非易失性存储器,诸如硬盘

驱动器、磁盘驱动器、诸如CD ROM、DVD、**Blu-Ray®** (蓝光) 盘的光盘驱动器或其它光学介质。计算机可读存储介质922可以包括但不限于：**Zip®**驱动器、闪存存储器卡、通用串行总线(USB) 闪存驱动器、安全数字(SD) 卡、DVD盘、数字视频带等。计算机可读存储介质922也可以包括基于非易失性存储器的固态驱动器(SSD) (诸如基于闪存存储器的SSD、企业闪存驱动器、固态ROM等)、基于易失性存储器的SSD (诸如基于固态RAM、动态RAM、静态RAM、DRAM的SSD、磁阻RAM(MRAM) SSD)，以及使用基于闪存存储器和DRAM的SSD的组的混合SSD。计算机可读介质922可以为计算机系统900提供计算机可读指令、数据结构、程序模块和其它数据的存储。

[0205] 在某些实施例中，存储子系统900也可以包括计算机可读存储介质读取器920，其可以进一步连接到计算机可读存储介质922。与系统存储器910一起并且可选地组合，计算机可读存储介质922可以全面地表示远程、本地、固定和/或可移动存储设备加上用于存储计算机可读信息的存储介质。

[0206] 在某些实施例中，计算机系统900可以提供对执行一个或多个虚拟机的支持。计算机系统900可以执行诸如管理程序之类的程序，以便促进虚拟机的配置和管理。每个虚拟机可以被分派存储器、计算(例如，处理器、内核)、I/O和联网资源。每个虚拟机通常运行其自己的操作系统，该操作系统可以与由计算机系统900执行的其它虚拟机所执行的操作系统相同或不同。对应地，多个操作系统可以潜在地由计算机系统900并发地运行。每个虚拟机一般独立于其它虚拟机而运行。

[0207] 通信子系统924提供到其它计算机系统和网络的接口。通信子系统924用作用于从计算机系统900接收数据和从计算机系统900向其它系统发送数据的接口。例如，通信子系统924可以使计算机系统900能够经由互联网建立到一个或多个客户端设备的通信信道，用于从客户端设备接收信息并且向客户端设备发送信息。此外，通信子系统924可以用于从特权账户管理器向发出请求的用户传送成功登录的通知或重新输入密码的通知。

[0208] 通信子系统924可以支持有线和/或无线通信协议两者。例如，在某些实施例中，通信子系统924可以包括用于(例如，使用蜂窝电话技术、高级数据网络技术(诸如3G、4G或EDGE(全球演进的增强数据速率)、WiFi (IEEE 802.11族标准)、或其它移动通信技术、或其任何组合)接入无线语音和/或数据网络的射频(RF)收发器部件、全球定位系统(GPS)接收器部件和/或其它部件。在一些实施例中，作为无线接口的附加或替代，通信子系统924可以提供有线网络连接(例如，以太网)。

[0209] 通信子系统924可以以各种形式接收和发送数据。例如，在一些实施例中，通信子系统924可以以结构化和/或非结构化的数据馈送926、事件流9217、事件更新930等形式接收输入通信。例如，通信子系统924可以被配置为实时地从社交媒体网络的用户和/或其它通信服务接收(或发送)数据馈送926(诸如**Twitter®** 馈送、**Facebook®**更新、诸如丰富站点摘要(RSS) 馈送的web馈送)，和/或来自一个或多个第三方信息源的实时更新。

[0210] 在某些实施例中，通信子系统924可以被配置为以连续数据流的形式接收本质上可能是连续的或无界的没有明确结束的数据，其中连续数据流可以包括实时事件的事件流928和/或事件更新930。生成连续数据的应用的示例可以包括例如传感器数据应用、金融报价机、网络性能测量工具(例如网络监视和流量管理应用)、点击流分析工具、汽车流量监视

等。

[0211] 通信子系统924也可以被配置为向一个或多个数据库输出结构化和/或非结构化的数据馈送926、事件流928、事件更新930等,其中该一个或多个数据库可以与耦合到计算机系统900的一个或多个流传输数据源计算机进行通信。

[0212] 计算机系统900可以是各种类型中的一种类型,包括手持便携式设备(例如,**iPhone®**蜂窝电话、**iPad®**计算平板、PDA)、可穿戴设备(例如,Google **Glass®**头戴式显示器)、个人计算机、工作站、大型机、信息站、服务器机架或任何其它数据处理系统。

[0213] 由于计算机和网络不断变化的性质,对图9中绘出的计算机系统900的描述旨在仅作为具体示例。具有比图9中所绘出的系统更多或更少部件的许多其它配置是可能的。基于本文所提供的公开内容和教导,本领域普通技术人员将认识到实现各种实施例的其它方式和/或方法。

[0214] 虽然已经描述了本公开的具体实施例,但是各种修改、变更、替代构造和等同物也包含在本公开的范围之内。本公开的实施例不限于在某些特定数据处理环境内的操作,而是可以在多个数据处理环境内自由操作。此外,虽然已经使用特定系列的事务和步骤描述了本公开的实施例,但是,对本领域技术人员应该显而易见的是,本公开的范围不限于所描述系列的事务和步骤。上述实施例的各种特征和方面可以被单独使用或结合使用。

[0215] 另外,虽然已经使用硬件和软件的特定组合描述了本公开的实施例,但是应该认识到的是,硬件和软件的其它组合也在本公开的范围之内。本公开的实施例可以仅用硬件、或仅用软件、或使用其组合来实现。本文描述的各种进程可以在同一处理器上实现,或以任何组合的不同处理器上实现。对应地,在部件或模块被描述为被配置为执行某些操作的情况下,这种配置可以例如通过设计电子电路来执行操作、通过对可编程电子电路(诸如微处理器)进行编程来执行操作、或其任何组合来完成。进程可以使用各种技术进行通信,其中各种技术包括但不限于用于进程间通信的常规技术,并且不同的进程对可以使用不同的技术,或者同一对进程可以在不同时间使用不同的技术。

[0216] 因而,说明书和附图应该在说明性而非限制性的意义上来看待。但是,将显而易见的是,在不脱离权利要求中所阐述的更广泛精神和范围的情况下,可以对其进行添加、减少、删除和其它修改和改变。因此,虽然已描述了具体的公开实施例,但是这些实施例不旨在是限制性的。各种修改和等同物在以下权利要求的范围之内。

100

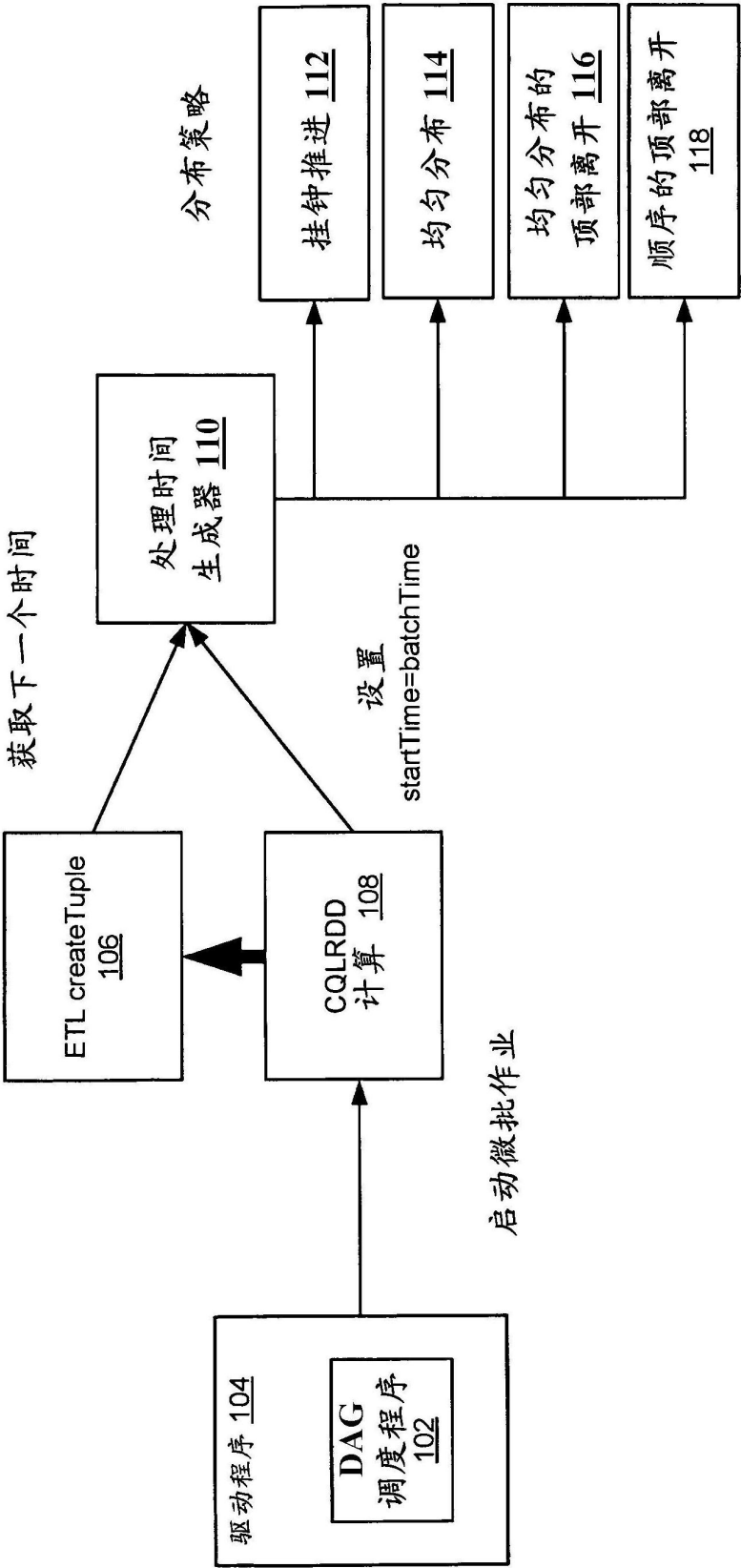


图1

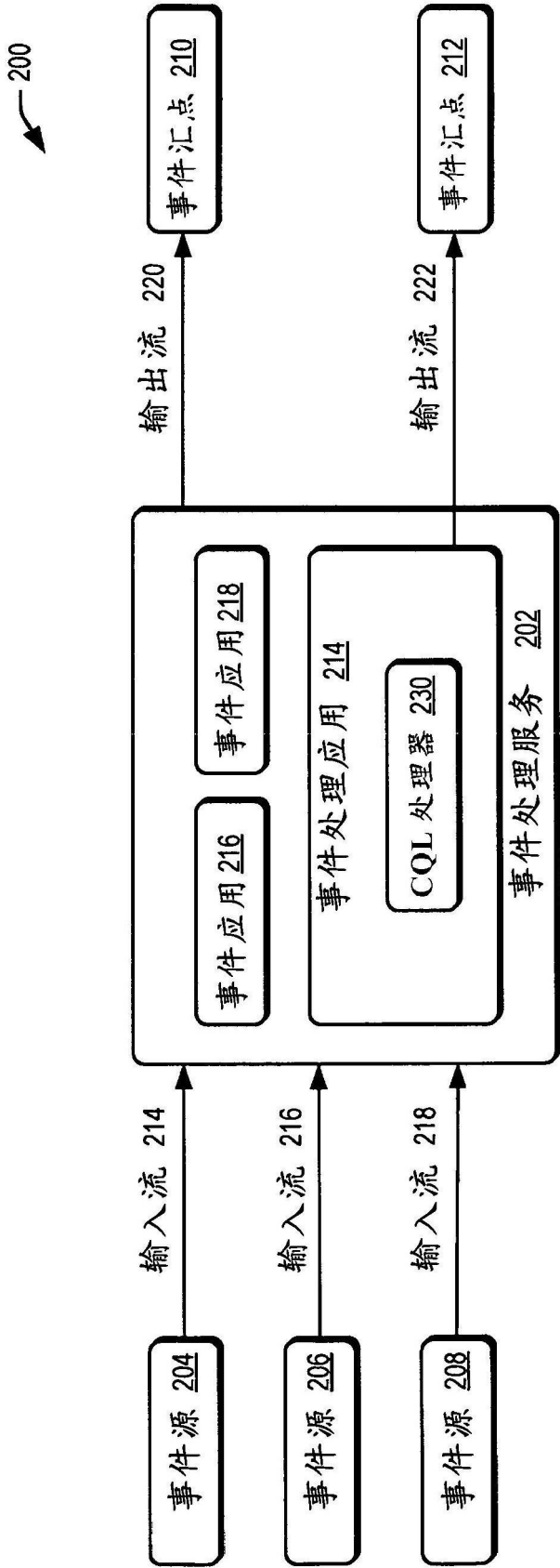


图2

300

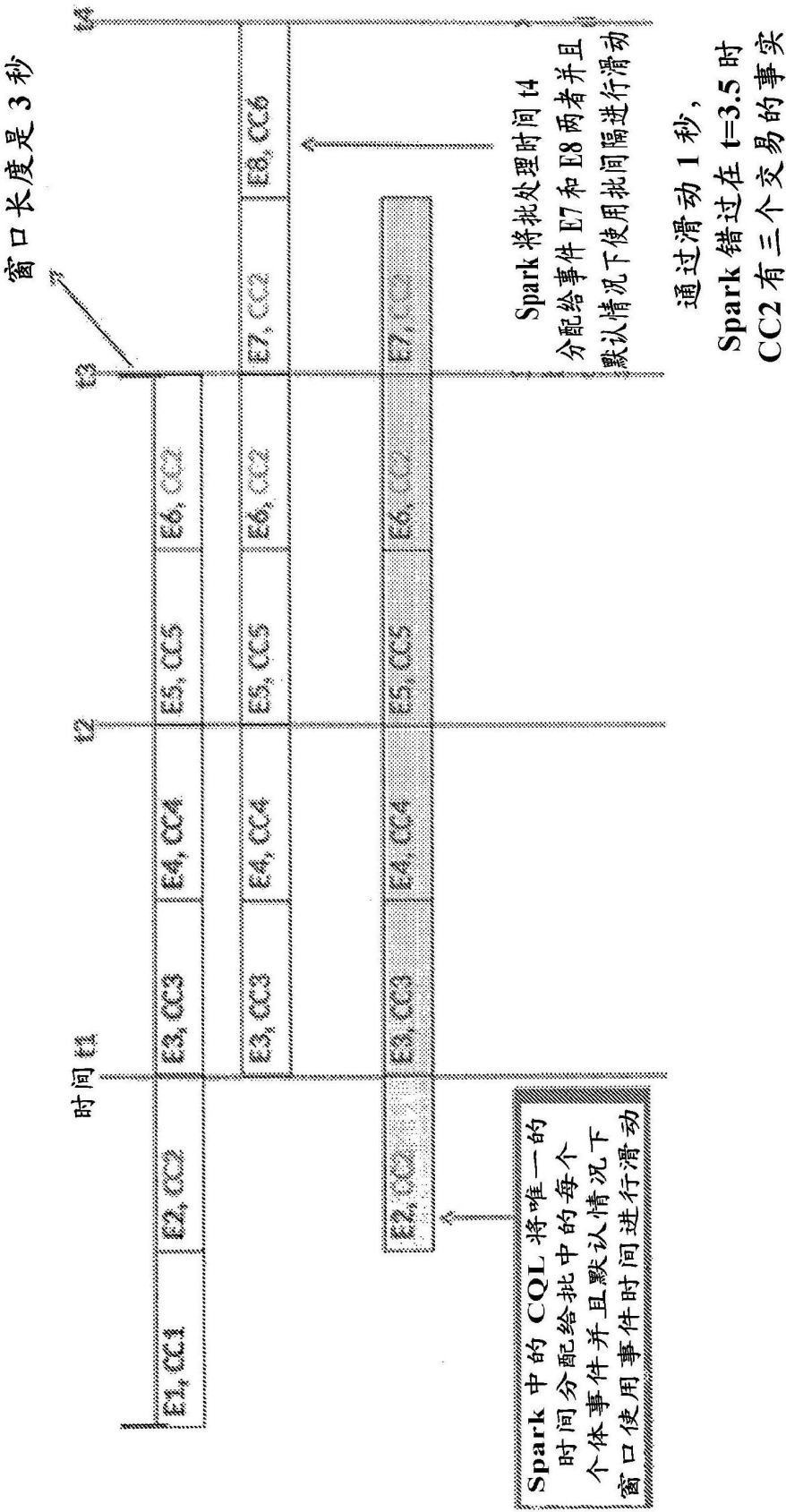


图3

400

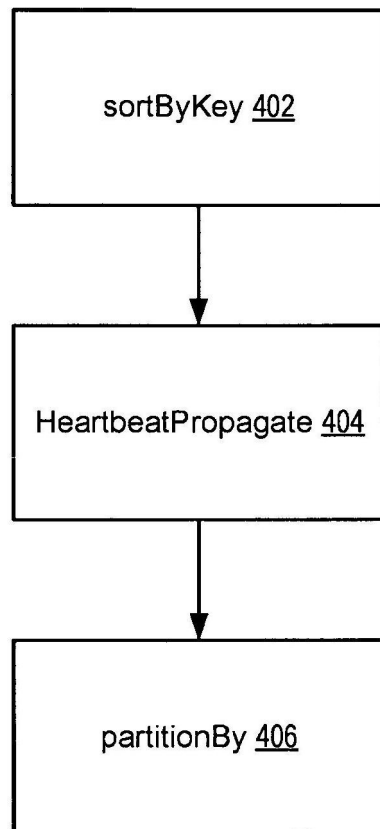


图4

500

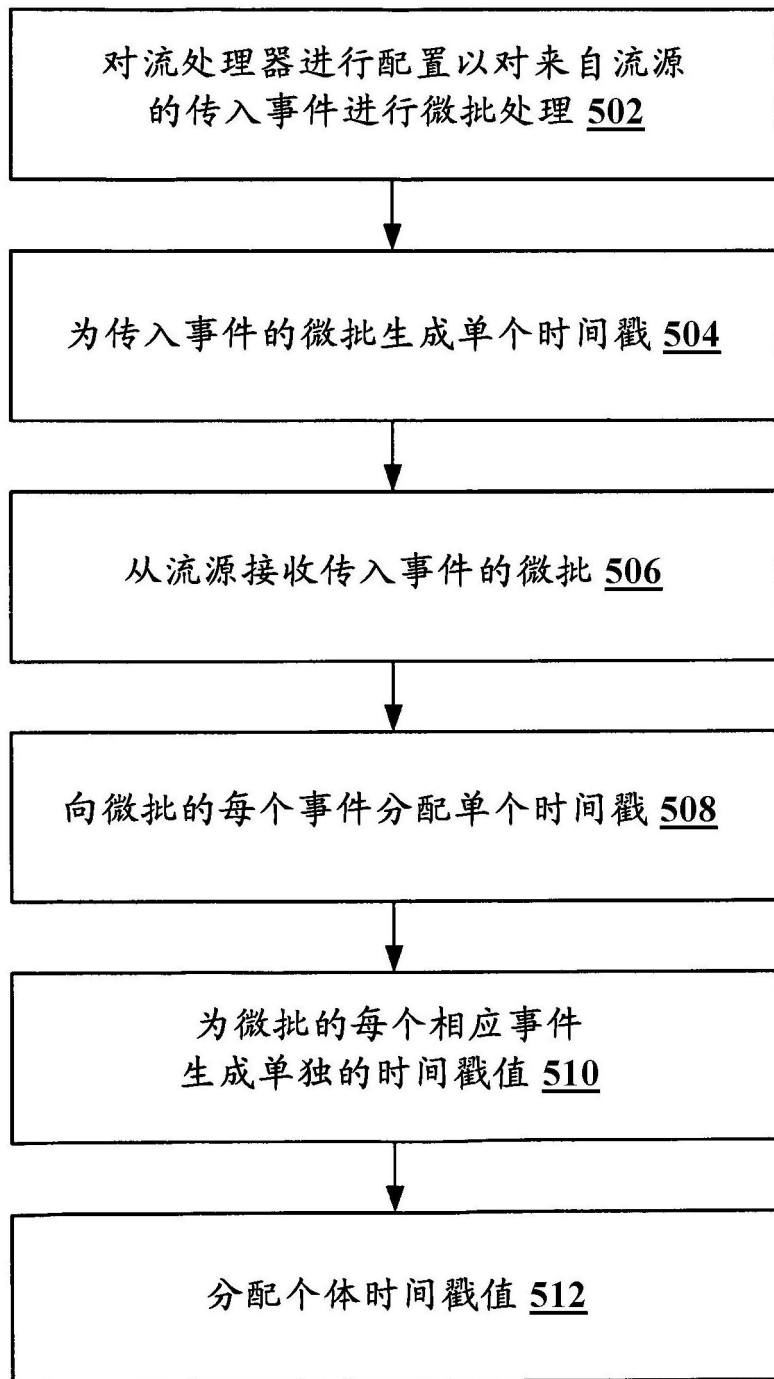


图5

600

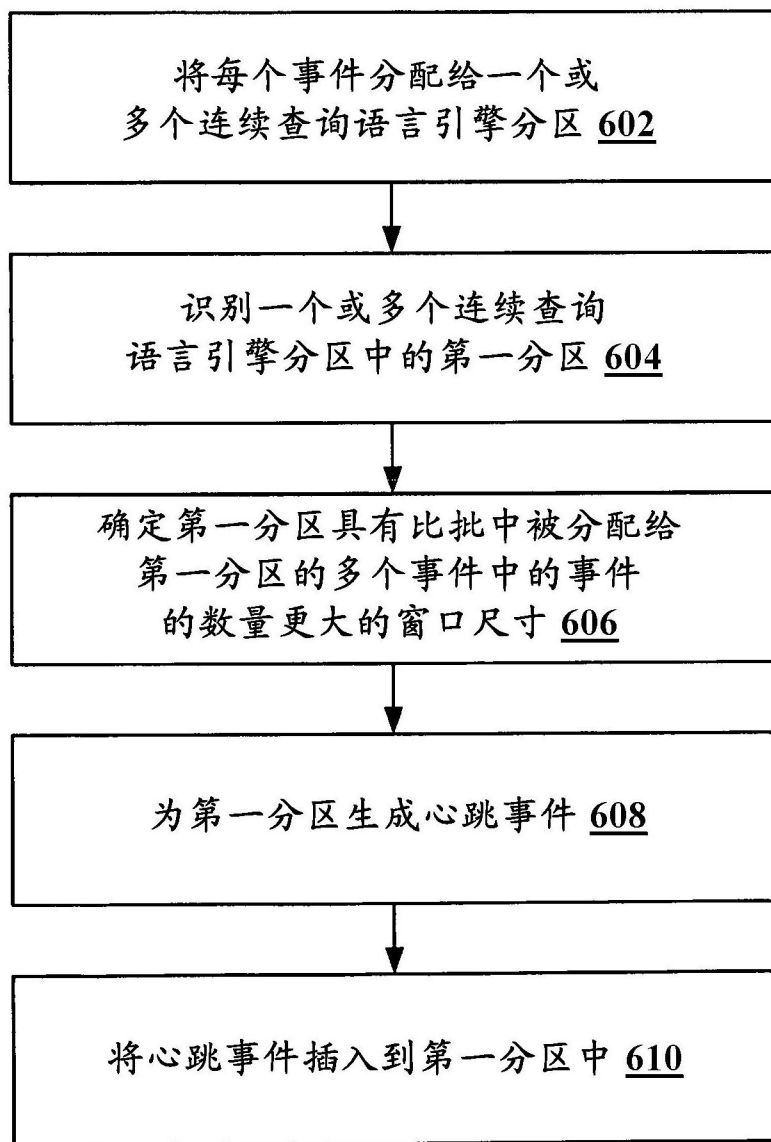


图6

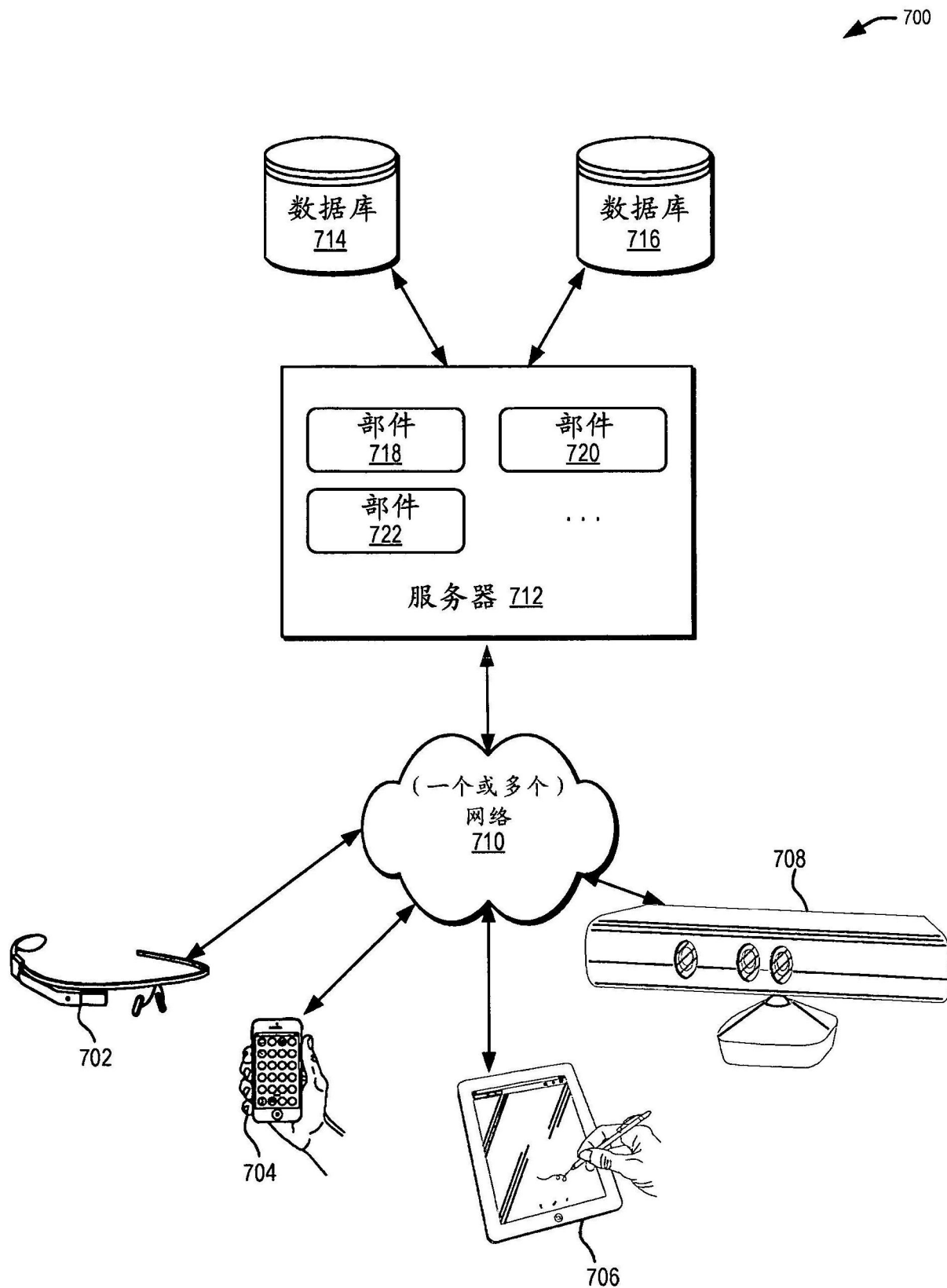


图7

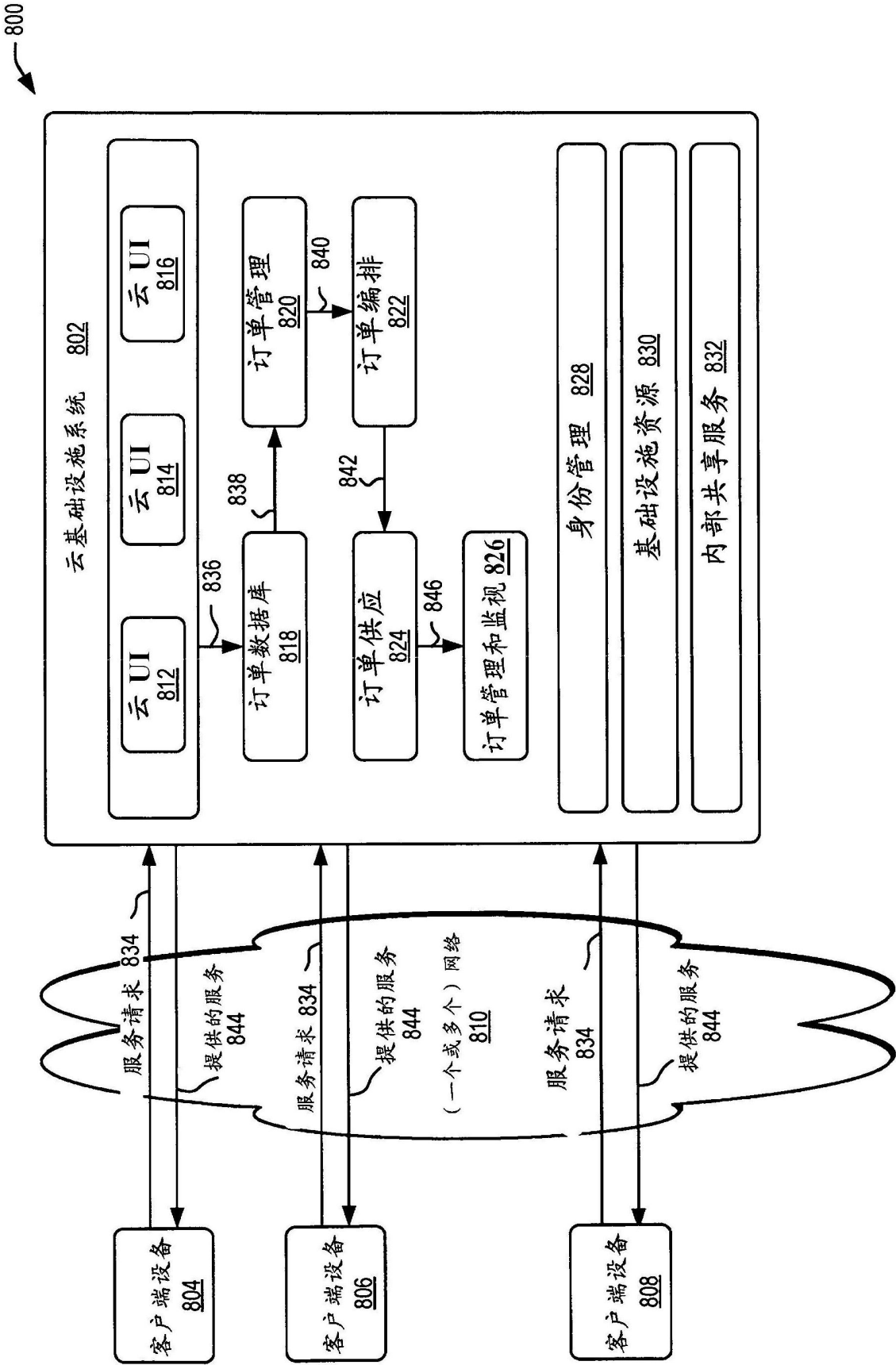


图8

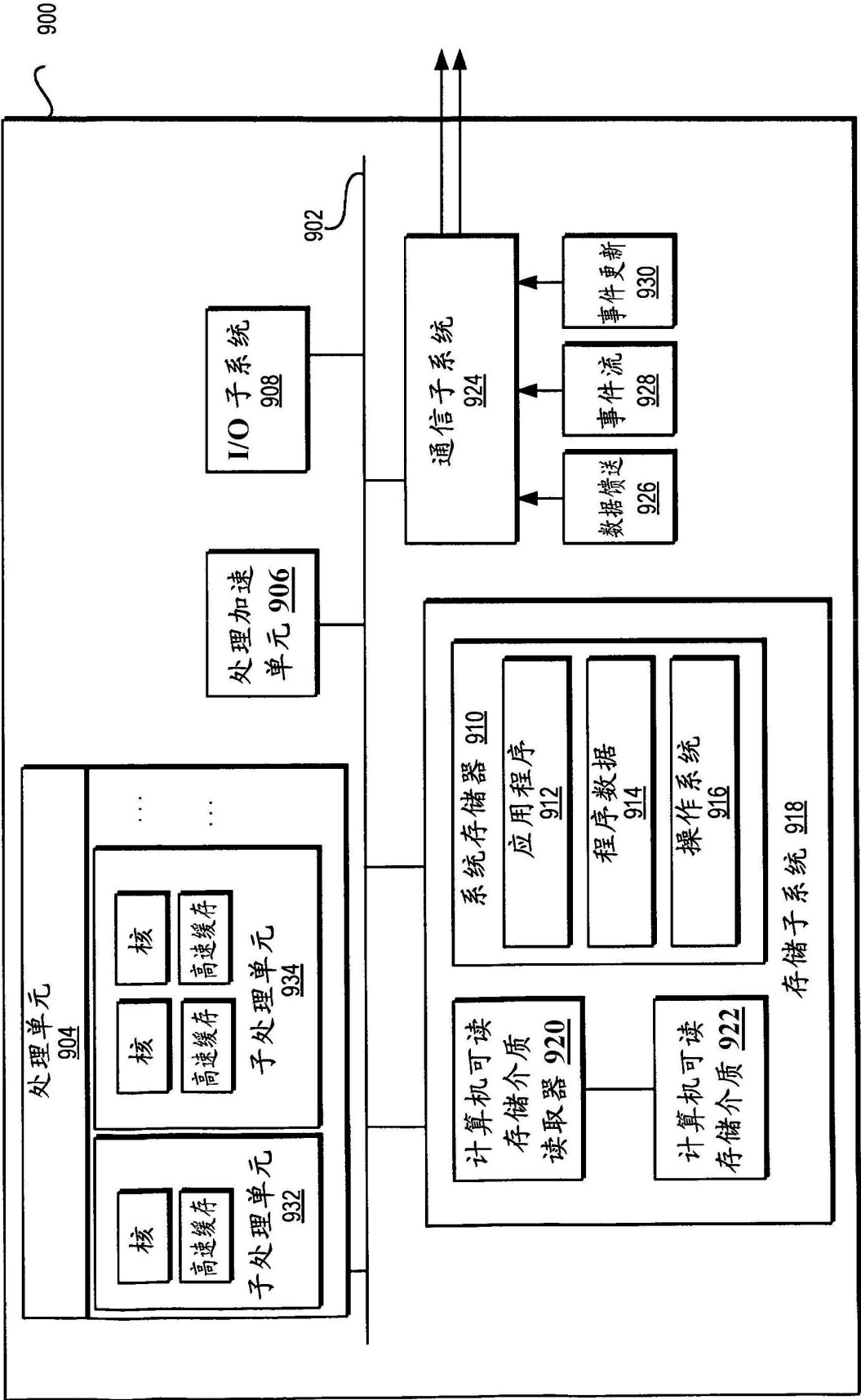


图9