(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 109716320 B
(45)授权公告日 2023.07.25

(21)申请号 201780056813.4

(22)申请日 2017.09.12

(65)同一申请的已公布的文献号
申请公布号 CN 109716320 A

(43)申请公布日 2019.05.03

(30)优先权数据
62/395,216 2016.09.15 US
15/700,862 2017.09.11 US

(85)PCT国际申请进入国家阶段日
2019.03.15

(86)PCT国际申请的申请数据
PCT/US2017/051196 2017.09.12

(87)PCT国际申请的公布数据
WO2018/052908 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 G·盖斯特

(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 周衡威

(51)Int.Cl.
G06F 16/22(2019.01)
G06F 16/2455(2019.01)
G06F 16/27(2019.01)

(56)对比文件
US 2014095471 A1,2014.04.03
US 2016006779 A1,2016.01.07
US 2015169786 A1,2015.06.18
US 2015222696 A1,2015.08.06
US 2008133209 A1,2008.06.05
CN 105723335 A,2016.06.29
CN 104838377 A,2015.08.12
CN 103502930 A,2014.01.08
Oracle Corporation.oracle fusion middleware developer guide for oracle event processing 11g(11.1.1.9).《Oracle》.2015,
inf-berlin.Distributed Event Based Systems - Complex Event Processing.《www.INF.FU-BERLIN.DE》.2015,

审查员 曹彦

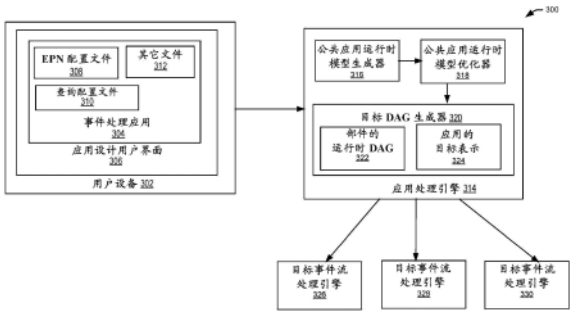
权利要求书3页 说明书47页 附图31页

(54)发明名称

用于事件处理的图生成的方法、系统、介质和应用处理引擎

(57)摘要

公开了一种用于处理事件流中的事件的事件处理系统。该系统接收识别应用的信息，并基于识别应用的信息而生成应用的公共应用运行时模型。系统将应用的公共应用运行时模型转换成应用的第一通用表示。应用的第一通用表示在多个目标事件处理系统中的第一目标事件处理系统中执行。应用的第一通用表示包括应用的部件的运行时有向无环图(DAG)。然后，系统将应用的第一通用表示发送到第一目标事件处理系统，以供第一目标事件处理系统执行。



CN 109716320 B

1.一种处理应用的方法,该方法包括:

在计算设备处接收识别应用的信息;

至少部分地基于识别所述应用的所述信息,由所述计算设备生成所述应用的公共应用运行时模型,其中生成所述应用的所述公共应用运行时模型包括将所述应用表示为一组一个或多个配置块,其中每个配置块表示具有相关联的元数据的事件bean;

由所述计算设备将所述应用的所述公共应用运行时模型转换成所述应用的第一通用表示,所述应用的所述第一通用表示被配置为在多个目标事件处理系统中的第一目标事件处理系统中执行,所述应用的所述第一通用表示包括所述应用的部件的运行时有向无环图DAG,其中将所述应用的所述公共应用运行时模型转换成所述应用的所述第一通用表示包括:将在所述应用的所述公共应用运行时模型中表示的一个或多个配置块转换成所述应用的部件的运行时DAG;以及

由所述计算设备将所述应用的所述第一通用表示发送到所述第一目标事件处理系统,以供所述第一目标事件处理系统执行。

2.如权利要求1所述的方法,其中所述应用被表达为部件的事件处理网络EPN,并且其中识别所述应用的所述信息包括与所述应用相关联的规则、EPN配置信息、和查询信息中的至少一个。

3.如权利要求1所述的方法,其中所述一组一个或多个配置块包括以下各项中的至少一个:入站套接字事件bean、出站套接字事件bean、持续查询语言CQL处理器事件bean、或一个或多个信道事件bean。

4.如权利要求1所述的方法,还包括由所述计算设备发送所述应用的部件的所述运行时DAG以供所述第一目标事件处理系统执行,其中所述第一目标事件处理系统是分布式事件处理系统。

5.如权利要求1至4中任一项所述的方法,还包括由所述计算设备将所述应用的所述公共应用运行时模型转换成所述应用的第二通用表示以在所述多个目标事件处理系统中的第二目标事件处理系统中执行,所述第二目标事件处理系统与所述第一目标事件处理系统不同。

6.如权利要求5所述的方法,其中将所述应用的所述公共应用运行时模型转换成所述应用的所述第二通用表示包括将在所述应用的所述公共应用运行时模型中表示的一个或多个配置块转换成所述应用的目标表示。

7.如权利要求6所述的方法,还包括由所述计算设备发送所述应用的所述目标表示以供所述第二目标事件处理系统执行,其中所述第二目标事件处理系统包括事件处理器系统。

8.一种存储计算机可执行指令的计算机可读介质,所述计算机可执行指令在由一个或多个处理器执行时将一个或多个计算机系统配置为执行至少以下指令:

使得所述一个或多个处理器接收识别应用的信息的指令;

使得所述一个或多个处理器至少部分地基于识别所述应用的所述信息来生成所述应用的公共应用运行时模型的指令,其中生成所述应用的所述公共应用运行时模型包括将所述应用表示为一组一个或多个配置块,其中每个配置块表示具有相关联的元数据的事件bean;

使得所述一个或多个处理器将所述应用的所述公共应用运行时模型转换成所述应用的第一通用表示的指令，所述应用的所述第一通用表示被配置为在多个目标事件处理系统中的第一目标事件处理系统中执行，所述应用的所述第一通用表示包括所述应用的部件的运行时有向无环图DAG，其中将所述应用的所述公共应用运行时模型转换成所述应用的所述第一通用表示包括：将在所述应用的所述公共应用运行时模型中表示的一个或多个配置块转换成所述应用的部件的运行时DAG；以及

使得所述一个或多个处理器将所述应用的所述第一通用表示发送到所述第一目标事件处理系统以供所述第一目标事件处理系统执行的指令。

9.如权利要求8所述的计算机可读介质，其中所述应用被表达为部件的事件处理网络EPN，并且其中识别所述应用的所述信息包括与所述应用相关联的规则、EPN配置信息、和查询信息中的至少一个。

10.如权利要求8所述的计算机可读介质，其中所述一组一个或多个配置块包括以下各项中的至少一个：入站套接字事件bean、出站套接字事件bean、持续查询语言CQL处理器事件bean、或一个或多个信道事件bean。

11.如权利要求8-10中任一项所述的计算机可读介质，还包括使得所述一个或多个处理器发送所述应用的部件的运行时DAG以供所述第一目标事件处理系统执行的指令，其中所述第一目标事件处理系统是分布式事件处理系统。

12.一种事件处理系统，包括：

存储多个指令的存储器；以及

处理器，被配置为访问所述存储器，所述处理器还被配置为执行所述多个指令以至少执行以下操作：

接收识别应用的信息；

至少部分地基于识别所述应用的所述信息，生成所述应用的公共应用运行时模型，其中生成所述应用的所述公共应用运行时模型包括将所述应用表示为一组一个或多个配置块，其中每个配置块表示具有相关联的元数据的事件bean；

将所述应用的所述公共应用运行时模型转换成所述应用的第一通用表示，所述应用的所述第一通用表示被配置为在多个目标事件处理系统中的第一目标事件处理系统中执行，所述应用的所述第一通用表示包括所述应用的部件的运行时有向无环图DAG，其中将所述应用的所述公共应用运行时模型转换成所述应用的所述第一通用表示包括：将在所述应用的所述公共应用运行时模型中表示的一个或多个配置块转换成所述应用的部件的运行时DAG；以及

将所述应用的第一通用表示发送到所述第一目标事件处理系统，以供所述第一目标事件处理系统执行。

13.如权利要求12所述的系统，其中所述应用被表达为部件的事件处理网络EPN，并且其中识别所述应用的所述信息包括与所述应用相关联的规则、EPN配置信息、和查询信息中的至少一个。

14.如权利要求13所述的系统，其中所述一组一个或多个配置块包括以下各项中的至少一个：入站套接字事件bean、出站套接字事件bean、持续查询语言CQL处理器事件bean、或一个或多个信道事件bean。

15.如权利要求13-14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多个指令以将所述应用的所述公共应用运行时模型转换成所述应用的第二通用表示从而在所述多个目标事件处理系统中的第二目标事件处理系统中执行，所述第二目标事件处理系统与所述第一目标事件处理系统不同。

16.一种应用处理引擎，包括用于执行如权利要求1-7中任一项所述的方法的部件。

用于事件处理的图生成的方法、系统、介质和应用处理引擎

[0001] 相关申请的交叉引用

[0002] 本申请依据35U.S.C.119(e)要求于2016年9月15日提交的标题为“FAST SERIALIZATION OF TUPLE BATCHES”的美国临时申请No.62/395216以及于2017年9月11日提交的标题为“Graph Generation for a Distributed Event Processing System”的美国非临时申请No.15/700,862的权益和优先权,每个申请的全部内容均通过引用并入本文,用于所有目的。

[0003] 本申请还涉及与本申请同一天提交的标题为“DATA SERIALIZATION IN A DISTRIBUTED EVENT PROCESSING SYSTEM”的代理人案卷No.088325-1043334(177600US)的申请序列No.15/700,784,与本申请同一天提交的标题为“CLUSTERING EVENT PROCESSING ENGINES”的代理人案卷No.088325-1043336(177620US)的申请序列No.15/700,914,以及与本申请同一天提交的标题为“DATA PARTITIONING AND PARALLELISM IN A DISTRIBUTED EVENT PROCESSING SYSTEM”的代理人案卷No.088325-1043337(177630US)的申请序列No.15/701,019。每个申请的全部内容依据35U.S.C.§120通过引用并入本文,如同在本文完全阐述了一样。

背景技术

[0004] 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。然后使用诸如结构化查询语言(SQL)之类的数据管理语言来查询和操纵所存储的数据。例如,可以定义并执行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查询是对存储在数据库中的有限数据集执行的。另外,当执行SQL查询时,它在有限数据集上执行一次并产生有限静态结果。因此,数据库最适合在有限存储数据集上运行查询。

[0005] 然而,许多现代应用和系统以持续数据或事件流而不是有限数据集的形式生成数据。此类应用的示例包括但不限于传感器数据应用、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。这些应用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。

[0006] 为这些类型的基于事件流的应用管理和处理数据涉及以强时间焦点(temporal focus)构建数据管理和查询能力。需要不同类型的查询机制,该查询机制包括对持续无界数据集的长时间运行的查询。虽然一些供应商现在提供面向事件流处理的产品套件,但这些产品仍然缺乏用于处置当今事件处理需求所需的处理灵活性。

发明内容

[0007] 提供了用于处理事件流的事件的技术(例如,方法、系统、存储可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,公开了一种事件处理系统。该系统被配置为接收识别应用的信息,并基于识别应用的信息生成应用的公共应用运行时模型。该系统被配置为将应用的公共应用运行时模型转换成应用的第一通用表示。在某些示例中,应用的第一通用表示被配置为在多个目标事件处理系统的第一目标事件处理

系统中执行。在某些示例中，应用的第一通用表示包括应用的部件的运行时有向无环图(DAG)。

[0008] 在某些实施例中，事件处理系统被配置为将应用的第一通用表示发送到第一目标事件处理系统以供第一目标事件处理系统执行。

[0009] 在某些实施例中，应用被表达为部件的事件处理网络(EPN)，并且识别应用的信息包括与应用相关联的规则、EPN配置信息、和查询信息。

[0010] 在某些实施例中，生成应用的公共应用运行时模型包括将应用表示为一组一个或多个配置块。在一些示例中，每个配置块表示具有相关联的元数据的事件bean。

[0011] 在某些实施例中，该一组一个或多个配置块包括以下各项中的至少一个：入站套接字事件bean、出站套接字事件bean、持续查询语言(CQL)处理器事件bean、或者一个或多个信道事件bean。

[0012] 在某些实施例中，将应用的公共应用运行时模型转换成应用的第一通用表示包括将在应用的公共应用运行时模型中表示的一个或多个配置块转换成应用的部件的运行时DAG。

[0013] 在某些实施例中，事件处理系统被配置为将应用的公共应用运行时模型转换成应用的第二通用表示，以在多个目标事件处理系统中的第二目标事件处理系统中执行。在某些示例中，第二目标事件处理系统与第一目标事件处理系统不同。

[0014] 在某些实施例中，将应用的公共应用运行时模型转换成应用的第二通用表示包括将在应用的公共运行时应用模型中表示的一个或多个配置块转换成应用的目标表示。

[0015] 在某些实施例中，事件处理系统被配置为发送应用的目标表示以供第二目标事件处理系统执行。在某些示例中，第二目标事件处理系统包括事件处理器系统。

[0016] 以上和以下描述的技术可以以多种方式并在许多上下文中实现。参考以下附图提供若干示例实现方案和上下文，如下面更详细地描述的。但是，以下实现方案和上下文只是其中的一小部分。

附图说明

[0017] 图1描绘了根据本公开实施例的示例事件处理系统体系架构的各方面，该示例事件处理系统体系架构提供可以针对不同的执行环境来处理事件处理应用的环境。

[0018] 图2是根据本公开实施例的、用于事件处理应用的事件处理网络(EPN)的图形表示。

[0019] 图3是图示了根据本公开实施例的应用处理引擎的部件的简化框图。

[0020] 图4是根据本公开实施例的、由公共应用模型生成器生成的“公共应用运行时模型”的表示的示例。

[0021] 图5是根据本公开实施例的、由有向无环图(DAG)生成器生成的部件的运行时DAG的示例。

[0022] 图6是可以结合本公开实施例的事件处理系统的简化高级图。

[0023] 图7是图示根据本公开实施例的分布式事件处理系统的部件的框图。

[0024] 图8是根据本公开一个实施例的、描述用于处理事件处理应用以生成应用的公共应用运行时模型的一组操作的过程的示例流程图。

[0025] 图9是根据本公开另一个实施例的、描述用于处理事件处理应用以生成应用的公共应用运行时模型的一组操作的过程的示例流程图。

[0026] 图10是图示根据本公开实施例的分布式事件处理系统的部件的简化框图。

[0027] 图11是根据本公开实施例的、用于执行弹性分布式数据集(RDD)对象中的数据的序列化和解序列化的过程的高级数据流。

[0028] 图12是根据本公开实施例的、描述可以通过其对一批事件中包括的数据进行序列化的一组操作的过程的示例流程图。

[0029] 图13A是根据本公开实施例的、描述用于针对事件的数值属性生成经序列化的数据值的集合的一组操作的过程的示例流程图。

[0030] 图13B是根据本公开实施例的、描述用于使用精度约简压缩技术针对事件的数值属性生成经序列化的数据值的集合的一组操作的过程的示例流程图。

[0031] 图13C是根据本公开实施例的、描述用于使用常规压缩技术针对事件的数值属性生成经序列化的数据值的集合的一组操作的过程的示例流程图。

[0032] 图13D是根据本公开实施例的、描述用于使用精度约简值索引压缩技术针对事件的数值属性生成经序列化的数据值的集合的一组操作的过程的示例流程图。

[0033] 图14是根据本公开实施例的、描述用于针对事件的非数值属性生成经序列化的数据值的集合的一组操作的过程的示例流程图。

[0034] 图15是根据本公开实施例的、可以基于确定事件流中的事件的属性的数据类型来对事件流数据进行序列化的方式的示例。

[0035] 图16是根据本公开实施例的、描述可以通过其对一批事件中包括的数据进行解序列化的一组操作的过程的示例流程图。

[0036] 图17是根据本公开实施例的、描述针对一批事件中的事件的一个或多个属性生成解序列化的数据值的集合的一组操作的过程的示例流程图。

[0037] 图18是根据本公开的实施例的、描述使用值索引压缩生成与一批事件中的事件的数值属性或非数值属性对应的解序列化的数据值的集合的一组操作的过程的示例流程图。

[0038] 图19是根据本公开实施例的、描述使用精度约简压缩技术生成与一批事件中的事件的数值属性对应的解序列化的数据值的集合的一组操作的过程的示例流程图。

[0039] 图20是根据本公开实施例的、描述用于生成与一批事件中的事件的数值属性对应的解序列化的数据值的集合的一组操作的过程的示例流程图。

[0040] 图21是根据本公开实施例的、描述用于生成与一批事件中的事件的数值属性对应的解序列化的数据值的集合的一组操作的过程的示例流程图。

[0041] 图22是图示根据本公开实施例的、被配置为用于调度和管理多个CEP引擎的分布式事件处理系统的部件的简化框图。

[0042] 图23是根据本公开实施例的、描述用于调度和管理多个CEP引擎的一组操作的过程的示例流程图。

[0043] 图24是图示根据本公开实施例的、被配置用于数据分区和并行性的分布式事件处理系统的部件的简化框图。

[0044] 图25是根据本公开实施例的、描述用于使用查询子句和对象ID自动对数据进行分区和并行化的一组操作的过程的示例流程图。

[0045] 图26描绘了用于实现本公开实施例的分布式系统的简化图。

[0046] 图27是根据本公开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作为云服务提供。

[0047] 图28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

[0048] 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。然而，对于本领域技术人员来说将清楚的是，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。

[0049] 复杂事件处理(CEP)的概述

[0050] 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平台的核心是持续查询语言(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、查询和执行型式匹配操作。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、集群(clustering)、分布式高速缓存、事件储存库和监视等。

[0051] 随着事件驱动的体系架构和复杂事件处理已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，以通过允许设施立即对电力需求的变化作出反应而使这些设施更高效。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。

[0052] 当今的信息技术(IT)环境为每样事物生成持续的数据流，从监视金融市场和网络性能，到业务处理执行和跟踪RFID标记的资产。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可以实时处理多个事件流以检测型式和趋势，并为企业提供利用新出现的机会或减轻发展中的风险的必要的可见性。

[0053] 持续数据流(也称为事件流)可以包括数据流或事件流，其本质上可以是持续的或无界的，没有明确的结束。在逻辑上，事件流或数据流可以是数据元素(也称为事件)的序列，每个数据元素具有相关联的时间戳。持续事件流可以在逻辑上表示为元素(s,T)的包(bag)或集合，其中“s”表示数据部分，并且“T”在时间域中。“s”部分一般被称为元组或事件。因此，事件流可以是带时间戳的元组或事件的序列。

[0054] 在一些方面，与流中的事件相关联的时间戳可以等于时钟时间。然而，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来表示。对于接收输入事件流的系统，事件以时间戳增加的次序到达系统。可以存在具有相同时间戳的多个事件。

[0055] 在一些示例中，事件流中的事件可以表示某个世俗事件的发生(例如，当温度传感器将值改变为了新值时，当股票代号的价格改变时)，并且与该事件相关联的时间信息可以指示由数据流事件表示的世俗事件何时发生。

[0056] 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以时间戳值增加的次序到达。这可以使得在事件流中接收的事件能够基于这些事件所关联的时间信息来排序。为了实现这种排序，时间戳可以以非递减方式与事件流中的事件相关联，使得较晚生成的事件与较早生成的事件相比具有较晚的时间戳。作为另一个示例，如果序列号被用作时间信息，那么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，多个事件可以与相同的时间戳或序列号相关联，例如，当由数据流事件表示的世俗事件同时发生时。属于同一事件流的事件一般可以按照由相关联时间信息强加于事件上的次序来处理，使得较早的事件在较晚的事件之前被处理。

[0057] 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，或者可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任意交织。

[0058] 事件流具有相关联的模式(schema)“S”，该模式包括时间信息以及一个或多个命名属性的集合。属于特定事件流的所有事件都符合与该特定事件流相关联的模式。因而，对于事件流(s,T)，事件流可以具有模式“S”如(<time_stamp>,<attribute(s)>)(即，(<时间戳>,<属性>))，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，用于股票报价机事件流的模式可以包括属性<stock symbol>(<股票代号>)和<stock price>(<股票价格>)。经由这种流接收的每个事件将具有时间戳和两个属性。例如，股票报价机事件流可能会接收以下事件和相关的时间戳：

[0059] …

[0060] (<timestamp_N>,<NVDA,4>)

[0061] (<timestamp_N+1>,<ORCL,62>)

[0062] (<timestamp_N+2>,<PCAR,38>)

[0063] (<timestamp_N+3>,<SPOT,53>)

[0064] (<timestamp_N+4>,<PDCO,44>)

[0065] (<timestamp_N+5>,<PTEN,50>)

[0066] …

[0067] 在上述流中，对于流元素(<timestamp_N+1>,<ORCL,62>)，事件是<ORCL,62>，具有属性“stock_symbol”(“股票代号”)和“stock_value”(“股票值”)。与流元素相关联的时间戳是“timestamp_N+1”。因此，持续事件流是事件的流，每个事件具有相同的属性系列。

[0068] 如上所述，流可以是CQL查询可以对其作用的数据的主要源。流S可以是元素(s,T)的包(也称为“多集”)，其中“s”在S的模式中，并且“T”在时间域中。此外，流元素可以是元组-时间戳对，其可以被表示为带时间戳的元组插入(insertion)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元组。此外，可以请求输入流的元组，以便以时间戳增加的次序到达系统。可替代地，关系(也称为“时变关系”，并且不与“关系型数据”混淆，“关系型数据”可以包括来自关系型数据库的数据)可以是从时间域到模式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

变元组包(即,瞬时关系)。在一些情况下,在每个时间实例处,关系可以是有界集。它还可以被表示为带时间戳的元组的序列,这些元组可以包括插入、删除和/或更新以捕获关系的改变的状态。与流类似,关系可以具有固定模式,关系的每个元组都可以符合该固定模式。另外,如本文所使用的,持续查询一般可以能够处理流和/或关系的数据(即,针对其进行查询)。此外,关系可以引用流的数据。

[0069] 事件处理应用

[0070] 在IT环境中,原始基础设施和业务事件的数量和速度都呈指数增长。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,移动设备的爆发和无处不在的高速连接性增加了移动数据的爆发。与此同时,对业务过程敏捷性和执行的需求也在增长。这两个趋势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方案型式的压力。实时事件处理要求基础架构和应用开发环境二者都在事件处理要求上执行。这些要求常常包括需要从日常用例扩展到极高速度的数据和事件吞吐量(可能还有以微秒而不是秒来度量的响应时间的时延)。此外,事件处理应用必须常常检测这些事件的流中的复杂型式。

[0071] Oracle Stream Analytics(Oracle流分析)平台面向众多行业和功能领域。以下是一些用例:

[0072] 电信:执行服务攻击检测的分布式拒绝以及实时呼叫细节(CDR)记录监视的能力。

[0073] 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。

[0074] 运输:在由于当地或目的地城市天气、地勤人员操作、机场安全等引起航班差异的情况下创建乘客警报并检测行李位置的能力。

[0075] 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。

[0076] 保险:学习和检测潜在欺诈性索赔的能力。

[0077] IT系统:实时检测故障的应用或服务器并触发纠正措施的能力。

[0078] 供应链和物流:实时跟踪货物并检测和报告潜在的到达延迟的能力。

[0079] 实时流传输和事件处理分析

[0080] 随着来自增加数量的连接设备的爆发性的数据,大量动态变化的数据增加;数据不仅是在组织内部移动,而且还在防火墙外移动。高速数据带来高价值,尤其是对于不确定性的业务过程。然而,这种数据中的一些在短时间内失去其操作价值。大数据使得用于获得可动作的洞察力的处理时间是奢侈的。另一方面,快数据要求从高度动态和战略性数据中提取最大价值。它要求更快地处理,并且促进尽可能迫近生成的数据采取及时的行动。Oracle Stream Analytics平台提供快数据和响应性。Oracle Edge Analytics(Oracle边缘分析)实时地将处理推送到网络边缘(network edge)、关连、过滤并分析数据,以获得可动作的洞察力。

[0081] Oracle Stream Analytics平台提供将传入的流传输事件与持久数据联接的能力,从而提供上下文感知的过滤、关连、聚合和型式匹配。它为常见的事件源提供轻量级的开箱即用适配器。它还为自定义的适配器开发提供易于使用的适配器框架。利用这个平台,

组织可以识别和预期看似无关的事件所表示的机会以及威胁。其增量处理范例可以使用最少量资源来处理事件，从而提供极低时延的处理。它还允许它创建非常及时的警报，并立即检测丢失或延迟的事件，诸如以下各项：

[0082] 相关连的事件：如果事件A发生，那么事件B几乎总是跟随在事件A的2秒内。

[0083] 丢失或失序的事件：事件A、B、C应当按次序发生。在A之后立即看到C，没有B。

[0084] 因果事件：所制造项的权重缓慢地趋于变低或读数落在可接受的范围之外。这标志着潜在的问题或未来的维护需求。

[0085] 除了实时事件源之外，Oracle Stream Analytics平台设计环境和运行时执行还支持跨事件流和持久数据存储区(如数据库和高性能数据网格)二者的基于标准的持续查询执行。这使得平台能够充当需要在几微秒或几分钟内进行回答的系统的智能的核心，以辨别否则将被忽视的型式和趋势。事件处理用例要求存储器内处理的速度以及标准数据库SQL的数学准确性和可靠性。这个平台查询监听传入的事件流，并对每个事件在存储器中持续地执行已注册的查询，同时利用先进的自动化算法进行查询优化。然而，在基于存储器中(in-memory)执行模型的同时，这个平台充分利用标准ANSI SQL语法进行查询开发，从而确保查询构造的准确性和可扩展性。这个平台完全符合ANSI SQL'99标准，并且是业界首批可用的、支持ANSI SQL审查的对标准SQL的扩展以实现实时持续查询型式匹配的产品之一。CQL引擎优化处理器内查询的执行，从而使开发人员更多地关注业务逻辑而不是优化。

[0086] Oracle Stream Analytics平台允许组合SQL和Java代码二者以提供健壮的事件处理应用。充分利用标准行业术语来描述事件源、处理器和事件输出或汇点(sink)，这个平台提供在应用内定义和操纵事件的、元数据驱动的方法。它的开发人员使用可视化的有向图画布和调色板进行应用设计，以快速概述事件的流以及跨事件和数据源二者的处理。通过拖放建模和配置向导来开发流程，开发人员然后可以输入适当的元数据定义以将设计与实现方案连接起来。在必要或优选时，使用一次点击，开发人员就可以进入自定义Java代码开发或直接使用Spring®框架将高级概念编码到其应用中。

[0087] 事件驱动的应用的特征常常在于在处置极高速率的流传输输入数据时提供低的和确定性的时延的需要。Oracle Stream Analytics平台的基础是基于OSGi®背板的轻量级Java容器。它包含来自WebLogic JEE应用服务器的成熟部件，诸如安全性、日志记录和工作管理算法，但在实时事件处理环境中充分利用这些服务。集成的实时内核提供独特的服务来优化由JMX框架支持的线程和存储器管理，从而为了性能和配置而实现与容器的交互。Web 2.0丰富互联网应用可以使用HTTP发布和订阅服务与平台进行通信，这使他们能够订阅应用频道并将事件推送到客户端。这个平台占地面积小，是轻量级的基于Java的容器，其提供更快的生产时间(time-to-production)并降低总体拥有成本。

[0088] Oracle Stream Analytics平台具有在标准的商用硬件上(或者最优地使用Oracle Exalogic及其与其它工程系统组合)以几微秒的处理时延来处置每秒数百万个事件的能力。这是通过完整的"自上而下"分层解决方案实现的，该分层解决方案不仅具有在高性能事件处理用例上的设计焦点，而且还具有与企业级实时处理基础设施部件的紧密集成。面向性能的服务器集群的平台体系架构通过与Oracle Coherence技术的紧密集成而专注于可靠性、容错和极高的灵活性，并使得企业能够可预测地跨数据网格对任务关键应用进行缩放，从而确保持续的数据可用性和事务完整性。

[0089] 此外,这个平台允许确定性的处理,这意味着可以以不同的速率将相同的事件馈送到多个服务器或同一个服务器,从而每次都实现相同的结果。与仅依赖正在运行的服务器的系统时钟的系统相比,这具有难以置信的优势。

[0090] 以上和以下描述的技术可以以多种方式并在许多上下文中实现。参考以下附图提供了若干示例实现方案和上下文,如下面更详细地描述的那样。然而,以下实现方案和上下文只是其中的一小部分。

[0091] 分布式事件处理

[0092] 在某些情况下,企业的用户可能期望快速识别并响应企业内发生的重要事件,以便他们可以在识别出这些事件后立即采取行动。例如,用户可能希望识别与已经超过企业内的阈值的销售订单相关的重要事件。在这种情况下,用户可以向数据存储库/数据仓库提交一个或多个查询,并希望在几秒钟内而不是几分钟或几小时内查看查询结果,以便用户在异常被检测到的情况下可以立即采取行动。当在几秒或几分钟内做出动作具有重大意义时,企业可以使用实时数据处理和数据分析来实时处理事件流以进行更具反应性的决策并且立即采取行动。

[0093] 根据本公开的实施例,公开了一种分布式事件处理系统,该系统可以使用CEP和分布式事件流处理的组合来相对快速且实时地处理或查询非常大量的数据。分布式事件处理系统可以通过针对持续接收的数据流(例如,实况馈送)执行查询(例如,CQL查询)来执行对数据流的实时处理。分布式事件处理系统可以接收一个或多个持续数据流、针对数据流注册持续查询,并在新数据在流中出现时持续地执行查询。由于这种类型的持续查询是长时间运行的,因此分布式事件处理系统可以向用户提供持续的结果流。

[0094] 在某些实施例中,所公开的分布式事件处理系统可以被配置为通过将应用的执行分布在系统内的机器集群上来部署和执行应用(例如,事件处理应用)。本文描述的事件处理应用可以包括规则集,该规则集可以以用于处理输入流的持续查询的形式来表达。持续查询可以包括识别要对接收到的事件执行的处理的指令(例如,逻辑),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输出。持续查询通常可以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。应用可以被配置为监听一个或多个输入事件流,执行逻辑(例如,查询)以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,并经由一个或多个输出事件流输出所选择的值得注意的事件。

[0095] 例如,事件处理应用可以包括词计数应用,该应用计数输入文本集合内对特定词的引用的数量。这种应用可以包括例如读取文本集合并计数每个词在每个文本中出现的次数的持续查询。输入文本可以包含例如来自在线应用(诸如**Facebook®**或**Twitter®**)的流中接收的短消息。如上所述,可以使用CQL语言来配置持续查询。例如,为了指定要在词计数流应用中执行的词计数任务/操作,用户可以编写CQL查询,该查询可以采用诸如如下形式:FROM location GROUP BY word SELECT count。这种查询可以搜集来自指定位置的所有句子、将来自这些句子的独特词分组成不同的组,然后计数每个组中的词的数量。

[0096] 通过将应用的执行分布在机器集群上,所公开的分布式事件处理系统可以被配置为向用户快速且实时地提供与应用的执行有关的结果。分布式事件处理系统可以被配置为将与应用有关的数据分区到分离的计算节点,并且每个计算节点可以作为分离的计算机器上的分离的文件来维护。每个这样的机器可以被配置为相对于在该机器上维护的数据与其

它机器并行地在应用中执行查询。

[0097] 用于分布式事件处理系统的高效DAG生成

[0098] 在本公开的某些实施例中，公开了一种用于处理与应用(例如，事件处理应用)相关的信息的应用处理引擎。该应用处理引擎被配置为接收识别事件处理应用的信息。在某些示例中，事件处理应用被表达为部件的事件处理网络(EPN)，并且识别事件处理应用信息的信息包括与事件处理应用的各种部件(例如，适配器、处理器、流或事件bean)相关的信息。例如，识别事件处理应用的信息可以包括配置信息、查询信息以及与应用相关的其它类型的信息。

[0099] 在某些实施例中，应用处理引擎可以被配置为处理识别应用的信息并生成应用的“公共应用运行时模型”。如本文所述，应用的“公共应用运行时模型”是应用作为一组一个或多个配置块的表示，其中每个配置块表示具有描述该应用的相关联的元数据的处理阶段。应用处理引擎可以被配置为将应用的“公共应用运行时模型”转换成应用的一个或多个通用表示。然后，应用处理引擎可以被配置为使得应用的这一个或多个通用表示在由不同目标事件处理系统支持的不同执行(运行时)环境中执行。

[0100] 根据本公开实施例的“公共应用运行时模型”的生成使得能够在不同的物理执行(运行时)环境中执行应用的通用表示，而无需应用的开发人员(例如，用户)重新编写应用代码以在目标引擎中执行应用之前适配目标引擎的特定物理执行(运行时)环境。

[0101] 可以以多种方式以及在多个上下文中实现上述技术。参考下面的图1-9提供了若干示例实现方案和上下文，其描述了所公开的分布式事件处理系统可以执行与事件处理应用的部署、处理和执行相关的操作的方式的附加细节。

[0102] 图1描绘了根据本公开的实施例的示例事件处理系统体系架构100的各方面，该示例事件处理系统体系架构100提供可以通过其针对不同的执行环境来处理事件处理应用的环境。在实施例中，体系架构(事件处理系统)100包括经由网络108可通信地连接到一个或多个用户设备102的应用处理引擎110。

[0103] 网络108可以促进用户设备102与应用处理引擎之间的数据通信和交换。网络108可以是本领域技术人员熟悉的任何类型的网络，其可以支持使用各种商业可用协议(包括但不限于TCP/IP、SNA、IPX、AppleTalk等)中的任何协议的数据通信。仅作为示例，网络108可以是局域网(LAN)(诸如以太网、令牌环网络等)、广域网、虚拟网络(包括但不限于虚拟专用网络(VPN))、互联网、内联网、外联网、公共交换电话网(PSTN)、红外网络、无线网络(例如，在任何IEEE 802.1X协议套件、本领域已知的蓝牙协议，和/或任何其它无线协议下操作的网络)，和/或这些和/或其它网络的任意组合。

[0104] 用户设备102可以是通用个人计算机(包括例如运行各种版本的Microsoft Windows和/或Apple Macintosh操作系统的个人计算机和/或膝上型计算机)、蜂窝电话或PDA(运行诸如Microsoft Windows Mobile并且启用互联网、电子邮件、SMS、Blackberry或其它通信协议的软件)、运行各种商用UNIX或类UNIX操作系统(包括但不限于各种GNU/Linux操作系统)的工作站计算机，或任何其它计算设备。例如，用户设备102可以是能够通过网络(例如，网络108)进行通信的任何其它电子设备(诸如瘦客户端计算机、启用互联网的游戏系统，和/或个人消息传递设备)。虽然示例系统环境100被示出为具有一个用户设备，但是在其它实施例中可以支持任何数量的用户和/或客户端计算设备。

[0105] 在某些实施例中，应用处理引擎110可以被配置为处理事件处理应用以供在不同的运行时环境中执行。在某些示例中，事件处理应用可以由用户设备102的用户生成。例如，用户102可以使用由客户端应用104提供的应用设计用户界面106使用用户设备中的客户端应用104(例如，浏览器)来构建应用(例如，事件处理应用)。如上所述，事件处理应用可以包括用于处理来自事件源的数据的输入流的(例如，以持续查询的形式表达的)规则集。事件源可以包括各种数据源，诸如监视设备、金融服务公司或机动车辆。使用这些数据，事件处理应用可能识别并响应型式、查找特殊事件并提醒其它应用，或进行基于快速变化的数据需要立即行动的某种其它工作。

[0106] 应用处理引擎110可以包括一个或多个计算机和/或服务器，其可以是通用计算机、专用服务器计算机(包括例如PC服务器、UNIX服务器、中程服务器、大型计算机、机架式服务器等)、服务器场、服务器集群或者任何其它适当的布置和/或组合。组成应用处理引擎110的计算设备可以运行任何操作系统或各种附加服务器应用和/或中间层应用，包括HTTP服务器、FTP服务器、CGI服务器、Java服务器、数据库服务器等。示例数据库服务器包括但不限于可从Oracle、Microsoft、Sybase、IBM等商业获得的那些服务器。

[0107] 在某些实施例中，应用处理引擎110可以被配置为从用户设备102接收如上所述的应用(例如，事件处理应用)并且处理应用中的信息以生成应用的“公共应用运行时模型”。如上所述，应用的“公共应用运行时模型”是应用的作为一组一个或多个配置块的表示，其中每个配置块表示具有描述该应用的相关联的元数据的事件bean。应用处理引擎110可以被配置为将应用的“公共应用运行时模型”转换成应用的一个或多个通用表示。在某些实施例中，应用处理引擎110然后可以被配置为使得应用的这些更通用的表示在由不同目标事件处理系统支持的不同执行(运行时)环境中执行。

[0108] 在某些实施例中，应用处理引擎110可以包括公共应用运行时模型生成器112、公共应用运行时模型优化器114和目标DAG生成器116。这些部件可以用硬件、固件、软件或其组合实现。公共应用运行时模型生成器112可以被配置为基于与应用相关联的信息为应用生成“公共应用运行时模型”。公共应用运行时模型优化器114可以被配置为优化“公共应用运行时模型”以便为应用生成经优化的公共应用运行时模型。目标DAG生成器116可以被配置为将经优化的公共应用运行时模型转换成可以由目标事件流处理引擎(系统)之一执行的应用的一个或多个通用表示。由应用处理引擎110的部件112、114和116执行的操作将在下面关于图2详细讨论。

[0109] 在某些实施例中，目标事件处理引擎118可以被配置为从应用处理引擎110接收公共应用运行时模型，并将公共应用运行时模型中的信息转换成应用的特定于平台的实现方案(即，目标事件处理应用120)，该实现方案可以在由目标事件流处理引擎118提供的运行时(执行)环境中执行。然后，目标事件流处理引擎118可以被配置为当新数据出现在流中时持续地执行目标事件处理应用120，并为用户提供持续的结果流。目标事件流处理引擎118可以通过针对持续接收到的数据流(例如，实况馈送)执行在目标事件处理应用120中定义的一个或多个操作(例如，CQL查询)来执行对数据流的实时处理。例如，目标事件流处理引擎118可以接收一个或多个持续数据流、针对数据流注册目标事件处理应用120，并且当新数据在流中出现时持续地执行在目标事件处理应用120中定义的一个或多个查询。由于这种类型的持续查询是长时间运行的，因此目标事件流处理引擎可以向用户提供持续的结果

流。由目标事件流处理引擎118执行的附加操作关于图3详细讨论。

[0110] 在某些实施例中，每个目标事件流处理引擎118可以表示用于执行目标事件处理应用的特定物理执行环境。例如，第一目标事件流处理引擎可以包括被配置为在第一物理执行(运行时)环境中执行目标事件处理应用的第一事件流传输平台，第二目标事件流处理引擎可以包括被配置为在第二物理执行(运行时)环境中执行应用的第二事件流传输平台，第三目标事件流处理引擎可以包括被配置为在第三物理执行(运行时)环境中执行应用的第三事件流传输平台，等等。第一、第二和第三事件流传输平台可以彼此不同。例如，第一事件流传输平台可以表示由Oracle®管理的Oracle事件处理器(OEP)系统。第二事件流传输平台可以表示第一类型的分布式事件处理平台(诸如由Spark®系统管理的Spark®框架)，并且第三事件流传输平台可以表示第三类型的分布式事件处理平台(诸如由Flink®系统管理的Flink®框架)。

[0111] 图2是根据本公开实施例的、用于事件处理应用的事件处理网络(EPN)200的图形表示。在某些示例中，事件处理应用可以被表达为部件的网络。这种部件的网络通常被称为事件处理网络(EPN)200。EPN 200是用于在事件处理应用的部件当中表达基于事件的交互和事件处理规范的概念模型。事件处理应用的部件可以包括适配器、流、处理器、业务逻辑普通旧Java对象(POJO)和bean。EPN200中的每个部件在处理经由事件流接收的数据中都具有角色。如上所述，事件处理网络(EPN)200可以包括描述这些各种部件、部件如何连接在一起、由应用处理的事件类型、用于对应用使用的事件的进行选择的持续查询或逻辑、在应用中定义的业务规则等等的信息。

[0112] 在某些实施例中，用户设备102的用户可以使用由用户设备中的客户端应用(例如，104)提供的应用设计用户界面(例如，106)来为事件处理应用生成EPN 200。在其它实施例中，用户可以经由应用设计用户界面提供识别应用的信息。这种信息可以包括例如在应用中定义的一个或多个持续查询、指定应用的部署类型的应用参数、应用的运行时配置(例如，要使用的执行器的数量、并行性参数、存储器的尺寸、高可用性参数)等，并且客户端应用可以基于这种信息为事件处理应用构建和/或生成EPN 200。

[0113] 在某些实施例中，并且如图2中所示，用于事件处理应用的EPN 200可以由以下部件类型组成：

[0114] (1)一个或多个适配器(202、204)，它们直接接口到输入和输出流以及关系源和汇点。适配器被配置为理解输入和输出流协议，并负责将事件数据转换成可由应用处理器查询的规格化形式。适配器可以将经规格化的事件数据转发到信道或输出流和关系汇点中。可以为各种数据源和汇点定义事件适配器。在图2所示的实施例中，适配器包括StreamOrRelationSource1(流或关系源1)适配器202和StreamOrRelationSource2(流或关系源2)适配器204。

[0115] (2)充当事件处理端点的一个或多个信道(206、208、210)。除其它之外，信道尤其负责对事件数据进行排队，直到事件处理代理可以对事件数据进行动作。

[0116] (3)一个或多个应用处理器(或事件处理代理)212被配置为消费来自信道的经规格化的事件数据、使用查询来处理它以选择值得注意的事件，并将所选择的值得注意的事件转发(或复制)到输出信道210。

[0117] (4)一个或多个bean 214、216和218被配置为监听输出信道220，并且通过将新事件插入输出信道220而被触发。在一些实施例中，这个用户代码是简单的普通旧Java对象(POJO)。用户应用可以使用一组外部服务(诸如JMS、Web服务和文件编写器)将所生成的事件转发到外部事件汇点。

[0118] (5)事件bean 214、216和218可以被注册以监听输出信道220，并且通过将新事件插入输出信道而被触发。在一些实施例中，这个用户代码可以使用Oracle CEP事件bean API，以便可以由Oracle CEP来管理该bean。

[0119] 在一个实施例中，事件适配器(202、204)将事件数据提供给输入信道(206、208)。输入信道(206、208)连接到与一个或多个CQL查询相关联的CQL处理器(212)，这些CQL查询对由输入信道(206、208)提供的事件进行操作。CQL处理器(212)连接到输出信道(220)，查询结果被写入该输出信道。

[0120] 在一些实施例中，可以为事件处理应用提供组装文件，该文件描述事件处理应用的各种部件、部件如何连接在一起、以及由应用处理的事件类型。可以提供分离的配置文件，用于指定用于对事件进行选择的持续查询或逻辑。在某些实施例中，可以使用Spring® XML框架来组装事件处理应用中的信息。如下面将更详细描述的，这个方法使得应用能够容易地与现有的Spring® bean以及基于依赖性注入机制的其它轻量级编程框架集成。例如，组装文件可以是Spring®框架上下文XML配置文件的自定义扩展，以便事件服务器可以整体上充分利用Spring®的控制反转(IoC)容器，从而允许用户在EPN的组装中无缝地使用Spring® bean。

[0121] 图3是图示根据本公开实施例的应用处理引擎的部件的简化框图300。在某些实施例中，应用处理引擎314可以被配置为从用户设备302的用户接收识别事件处理应用(例如，304)的信息并基于该信息生成公共运行时应用模型。如上所述，事件处理应用304可以由用户设备302的用户使用由用户设备中的客户端应用(例如，104)提供的应用设计用户界面306来生成。

[0122] 在一些示例中，识别应用的信息可以包括描述应用的各种部件(例如，适配器、处理器、流或事件bean)的信息。这种信息可以包括例如配置信息、查询信息和其它类型的信息。配置信息可以包括例如描述事件处理应用的各种部件、部件如何连接在一起、以及由应用处理的事件类型的信息。例如，配置信息可以包括将事件处理应用描述为部件的网络(即，EPN 200)的信息。查询信息可以包括指定用于由应用对事件进行选择的持续查询或逻辑的信息。其它类型的信息可以包括应用中定义的普通旧Java对象(POJO)和业务规则。

[0123] 在某些示例中，可以在配置文件中指定识别应用的信息。例如，事件处理应用304的EPN 200中的每个部件可以具有相关联的配置文件。在其它示例中，应用304中的信息可以在单个配置文件中表示，该文件包括用于应用中的所有部件的信息。在一个实现方案中，配置文件可以表达为常规XML文档，该常规XML文档的结构使用标准XML模式基于由公共运行时模型定义的配置模式来定义。

[0124] 在某些示例中，可以使用诸如EPN配置文件、查询配置文件和其它文件之类的各种配置文件来指定识别应用的信息。下面示出了用于事件处理应用的EPN配置文件和查询配

置文件的示例。在所示示例中，事件处理应用是订单事件处理应用304，该应用被配置为接收和处理持续的事件流，其中每个事件表示针对公司销售的物品的订单。订单事件流中的每个订单可以包括与物品相关的诸如订单标识符、订单状态和订单量之类的属性。以下示出用于事件处理应用304的EPN配置文件308的示例。EPN配置文件308包括一系列子元素，其中每个子元素包括用于事件处理应用中的事件处理部件的配置信息。

[0125] EPN配置文件

[0126] <wlevs:event-type-repository>

[0127] <wlevs:event-type type-name＝“OrderEvent”>

[0128] <wlevs:properties>

[0129] <wlevs:property name＝“orderId”type＝“int”/>

[0130] <wlevs:property name＝“status”type＝“char”/>

[0131] <wlevs:property name＝“amount”type＝“int”/>

[0132] </wlevs:properties>

[0133] </wlevs:event-type>

[0134] </wlevs:event-type-repository>

[0135] <wlevs:adapter id＝“socketAdapter“provider＝“socket”/><wlevs:channel id＝“orders”event-type＝“OrderEvent”>

[0136] <wlevs:listener ref＝“orderProcessor”/>

[0137] <wlevs:source ref＝”socketAdapter”/>

[0138] </wlevs:channel>

[0139] <wlevs:processor id＝”orderProcessor”/>

[0140] <wlevs:channel id＝“omtputChannel”event-type＝”OrderEvent“>

[0141] <wlevs:listener ref＝“outputAdapter”/>

[0142] <wlevs:source ref＝”orderProcessor”/>

[0143] </wlevs:channel>

[0144] <wlevs:adapter id＝”outputAdapter”provider＝”stdout”/>

[0145] 以下示出用于事件处理应用304的查询配置文件310的示例。查询配置文件310指定用于在事件处理应用中选择事件的一个或多个持续查询或逻辑。

[0146] 查询配置文件

[0147] <processor>

[0148] <name>orderProcessor</name>

[0149] <rules>

[0150] <query id＝”helloworldRule”><![CDATA[

[0151] select status,count(*)from orders group by status

[0152]]]>

[0153] </query>

[0154] </rules>

[0155] </processor>

[0156] 在某些实施例中，应用处理引擎314可以被配置为基于在EPN配置文件308、查询配

置文件310和其它文件312中指定的配置信息、查询信息以及其它信息为应用(例如,304)生成“公共应用运行时模型”。然后,该“公共应用运行时模型”可以由目标DAG生成器320转换成应用的一个或多个通用表示,以便在由不同目标事件流处理引擎226、228和230支持的不同的物理执行(运行时)环境中执行。“公共应用运行时模型”的生成使得能够在不同的物理执行(运行时)环境中执行应用的通用表示,而应用的开发人员(例如,用户)不必在应用在目标引擎中执行之前重写应用代码以适应目标引擎的特定物理执行(运行时)环境。公共应用运行时模型独立于物理执行环境。独立于平台的抽象允许系统容易地针对物理执行环境生成DAG和代码。

[0157] 在某些实施例中,由应用处理引擎314生成“公共应用运行时模型”可以包括将应用表示为具有相关联的配置信息的一组一个或多个事件bean。下面详细描述应用处理引擎314可以将“公共应用运行时模型”表示为一组事件bean的方式的附加细节。

[0158] 在某些实施例中,应用处理引擎314包括公共应用运行时模型生成器316、公共应用运行时模型优化器318和目标DAG生成器320。公共应用运行时模型生成器316被配置为基于在EPN配置文件308、查询配置文件310和其它文件312中指定的识别应用的信息为应用生成“公共应用运行时模型”。在某些实施例中,由公共应用运行时模型生成器316生成“公共应用运行时模型”涉及使用利用Spring®应用框架实现的EPN加载器将EPN配置文件308、查询配置文件310和其它文件312加载到存储器中。其结果是通过Spring®控制反转注入(inversion of control injection)连接的一组Spring®bean。然后,公共应用运行时模型生成器316被配置为使用诸如JAXB(用于XML绑定的Java体系架构)之类的XML解析器来解析EPN配置文件308,并将解析后的配置文件设置到EPN网络中每个相关联的bean。在某些示例中,每个解析后的配置块或文件将具有标识符,以便块或文件可以找到事件bean并将配置块或文件设置到该事件bean。因此,在实施例中,为应用生成“公共应用运行时模型”包括将应用表示为一组一个或多个配置块,其中每个配置块表示具有相关联的元数据的Spring®事件bean。图4中示出了“公共应用运行时模型”的表示的示例。在某些示例中,“公共应用运行时模型”维护关于事件处理应用的部件的元数据并且在其中没有任何处理逻辑。

[0159] 图4描绘了根据本公开实施例的、由公共应用模型生成器316生成的“公共应用运行时模型”400的表示的示例。在实施例中,可以基于(例如,来自EPN配置文件308、查询配置文件310和其它文件312的)识别应用的信息生成“公共应用运行时模型”400,并且可以将“公共应用运行时模型”400表示为一组一个或多个配置块,其中每个配置块表示具有相关联的元数据的Spring®事件bean。在实施例中,公共运行时应用模型400中的配置块402、404、406、408和410可以包括以下信息:

```
SocketInbound 402
        host = "localhost"
        port = 9999

Channel-1 404
        tableName = "orders"
        eventType = "OrderEvent"
        relation = false
```

[0160]

```
CQL Processor 406
        rule= "select status, count(*) from orders group by status"

Channel-2 408
        eventType = "OrderEvent"

StdoutOutbound 410
        <no additional info>
```

[0161] 例如，在配置块402中，socket inbound(入站套接字)表示EPN中的"socketAdapter"(套接字适配器)适配器，在配置块404中，channel-1(信道-1)表示EPN中的"orders"(订单)信道，在配置块406中，CQL处理器表示EPN中的"orderProcessor"(订单处理器)处理器，在配置块408中，channel-2(信道-2)表示EPN中的"otutputChannel"(输出信道)信道，并且在配置块410中，socket outbound(出站套接字)410表示EPN中的"outputAdapter"(输出适配器)适配器。

[0162] 如上所述，将事件处理应用表示为"公共应用运行时模型"400使得应用能够在不同的目标事件流处理引擎(例如，326、328或330)中执行而无需用户在应用在目标引擎中执行之前重新编写应用的代码以适应目标引擎的特定物理执行(运行时)环境。

[0163] 返回到图3的讨论，在某些实施例中，由公共应用运行时模型生成器316生成的公共应用运行时模型还可以由公共应用运行时模型优化器318优化。公共应用运行时模型(例如，400)的优化可以包括例如将配置块内的多个阶段组合成单个阶段(例如，将多个持续查询组合成单个经优化的持续查询)或者将单个阶段分成多个阶段以用于并行处理(例如，把持续查询分成可以执行map(映射)和reduce(精简)操作的多个持续查询)。例如，没有重新分区的连贯查询可以用单个CQL进程组合成单个CQL阶段。例如，"select*from orders where orderStatus＝'open'"和"select count(*)from orders group by orderId"这两个阶段可以用"select count(*)from orders group by orderId where orderStatus＝"open"组合成一个CQL阶段。优化器还可以将单个阶段分成多个阶段，以便最大化可缩放性。

例如,完全有状态(fully-stateful)的查询阶段“select count(*)from orders”可以用分区被分为两个阶段,“select count(*)from order group by orderId”和“select sum(*)from counts”。这使用能够使用分区利用部分计数来处理事件的全局计数,然后可以将部分计数相加到全局计数中。

[0164] 在某些实施例中,目标DAG生成器320可以被配置为将经优化的公共应用运行时模型转换成可以由目标事件流处理引擎(326、328、或330)之一执行的应用的一个或多个通用表示。例如,目标DAG生成器320可以被配置为取决于应用将在其中执行的目标事件流处理引擎而将经优化的公共应用运行时模型转换成运行时DAG 322或应用的目标表示324。例如,如果目标事件流处理引擎(例如,326)是由Oracle®管理的Oracle事件处理器(OEP),那么目标DAG生成器320可以被配置为将公共应用运行时模型(或经优化的公共应用运行时模型)中的配置块转换成将由目标事件流处理引擎326执行的应用的目标表示324。在某些实施例中,转换可以包括由目标DAG生成器320对带有具有实际处理逻辑的适当bean的公共应用运行时模型的对象的复制。例如,公共应用运行时模型中的CQL处理器bean的元数据可以被复制到事件处理部件(例如,CQL处理器)的新实例,该实例运行CQL引擎以使用(例如,从公共运行时应用模型中的CQL处理器复制的)给定的查询来处理输入事件。

[0165] 例如,如果目标事件流处理引擎(例如,328)是由Spark®分布式系统管理的分布式事件处理平台,那么目标DAG生成器320可以被配置为将公共应用运行时模型(或经优化的公共应用运行时模型)中的对象(配置块)转换成表示应用的部件的运行时DAG 322。由目标DAG生成器生成的部件的运行时DAG 322的示例在图5中示出。然后,部件的运行时DAG 322由目标事件流处理引擎328转换成目标应用(Spark®CQL应用)。由目标事件流处理引擎328为订单事件处理应用生成的、用于计算按订单状态分组的订单数量的目标应用的示例如下所示:

[0166] 目标应用的示例

[0167]

```
Setup
1. val sparkConf = new SparkConf
2. val sc = new SparkContext(sparkConf)
3. val cc = new CQLContext(sc, Seconds(1))

Event Type, Stream Registration
4. val orderEvent = EventType("orders", Attribute("orderId",INT),
   Attribute("status",CHAR), Attribute("amount", INT) )
5. cc.registerEventType(orderEvent)
6. cc.registerStream(orderEvent)

Load Data
7. val lines = cc.socketTextStream("localhost", 9999)
8. val rows = lines.map(_.split(","))
9. val kv_orders = rows.map(r => (r(1), EventUtil.createTupleValue(orderEvent,
   r(0).toInt, r(1), r(2).toInt) )

Partition Data
10. val orders = rorders.transform( rdd => rdd.partitionBy(new
    OrderPartitioner(numPartitions)).map { case(k,v) => v }
)

CQL Processing
11. val result = cc.cql(orders, "select status, count(*) from orders group by status")

Output
```

[0168]

```
12. val sresult = result.map(x => x.mkString(","))
13. sresult.print
```

[0169] 图5是根据本公开实施例的、由目标DAG生成器生成的部件的运行时有向无环图(DAG)500的示例。在某些实施例中，并且如上所述，当目标应用的执行(运行时)环境是分布式事件处理系统(例如，**Spark®**分布式事件处理系统)时，目标DAG生成器320可以被配置为将公共应用运行时模型(例如，400)中的对象(402、404、406、408和410)转换成表示该应用的部件的运行时DAG 500。

[0170] 在实施例中，部件的运行时DAG包括以下部件，SocketText(套接字文本)502、Map-1(映射-1)504、Map-2(映射-2)506、PartitionBy(按…分区)508、CQL 510、Map-3(映射-3)512和Print(打印)514。(在图4中示出的)SocketInbout 402被转换成SocketText 502、Map-1 504和Map 506。SocketText 502部件包括上面所示的目标应用的示例中的第7行，它从套接字加载字符串。Map-1 504部件包括目标应用的示例中的第8行，它将字符串转换成逗号分隔的值。Map-2 506部件包括目标应用的示例中的第9行，它将逗号分隔的值转换成元组。CQL处理器406和Channel-1 404被转换成PartitionBy 508和CQL 510。PartitionBy 508部件包括目标应用的示例中的第10行，它按照CQL中的标准创建基于组的分区。CQL 510部件包括目标应用的示例中的第11行，它是主要的CQL处理阶段。Channel-2 408和StdoutOutbound 410被转换成Map-3512和Print 51。Map-3 512部件包括目标应用的示例中的第12行，它将元组转换成逗号分隔的字符串，并且Print 514部件包括来自目标应用的示例的第13行，它将输出字符串打印到stdout控制台。

[0171] 图6描绘了可以结合本公开实施例的事件处理系统600的简化高级图。在实施例中，事件处理系统600可以表示由Oracle®管理的Oracle事件处理器(OEP)系统。事件处理系统600可以包括一个或多个事件源(604、606、608)、被配置为提供用于处理事件流的环境的事件处理服务(EPS)602(也称为CQ服务602)，以及或更多事件汇点(610、612)。事件源生成由EPS 602接收的事件流。EPS 602可以从一个或多个事件源接收一个或多个事件流。

[0172] 例如，如图6中所示，EPS 602从事件源604接收第一输入事件流614，从事件源606接收第二输入事件流616，以及从事件源608接收第三事件流618。一个或多个事件处理应用(614、616和618)可以部署在EPS 602上并由EPS 602执行。由EPS 602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事件的处理逻辑来处理经由一个或多个事件流接收到的事件。然后可以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、612)。例如，在图6中，EPS 602将第一输出事件流620输出到事件汇点610，并将第二输出事件流622输出到事件汇点612。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任何一个而不会造成对其它部件的改变。

[0173] 在一个实施例中，EPS 602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，EPS 602可以例如通过使用JRockit Real Time来支持用于处理事件的超高吞吐量和微秒时延。EPS 602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)，其包括用于开发事件处理应用的工具(例如，Oracle CEP Visualizer和Oracle CEP IDE)。

[0174] 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6提供了用于一个这样的事件处理应用614的深入分析。如图6中所示，事件处理应用614被配置为监听输入事件流618，执行包括用于从输入事件618中选择一个或多个值得注意的事件的逻辑的持续查询630，并经由输出事件流622将所选择的值得注意的事件输出到事件汇点612。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。

[0175] 虽然图6中的事件处理应用614被示为监听一个输入流并经由一个输出流输出所选择的事件，但这不是限制性的。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，并经由一个或多个输出事件流将所选择的事件输出到一个或多个事件汇点。相同的查询可以与多于一个事件汇点并与不同类型的事件汇点相关联。

[0176] 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 602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 602提供了特殊的查询机制，该机制使得能够在EPS 602接收到事件时执行事件的处理，而无需存储所有接收到的事件。

[0177] 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持续查询

的形式表达。持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输出。持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以及生成事件的输出流。持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这些出站事件的输入事件流中的事件的数量。

[0178] 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,都可以执行已经由应用针对该事件流向EPS 602注册的持续查询。作为持续查询执行的一部分,EPS 602基于由持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,并且作为持续查询执行的结果输出。

[0179] 可以使用不同语言对持续查询进行编程。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)产品使用的CQL来配置持续查询。Oracle的CQL是一种声明性语言,其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。在某些实施例中,CQL是基于SQL的,其具有添加的支持对流传输事件数据的处理的构造。

[0180] 应当认识到的是,图6中描绘的系统600可以具有除图6中描绘的那些部件之外的其它部件。另外,图6中示出的实施例仅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600可以具有比图6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配置或布置。系统600可以是各种类型,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。

[0181] 图7是图示根据本公开实施例的分布式事件处理系统710的部件的简化框图700。图7中所示的实施例是可以结合本公开实施例的分布式事件处理系统的一个示例。在一些其它实施例中,系统710可以具有比图7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配置或布置。系统710可以是任何类型的计算设备,诸如但不限于移动计算设备、桌面计算设备、瘦客户端计算设备和/或云计算设备、服务器或任何其它数据处理系统。

[0182] 在一些示例中,分布式事件处理系统710可以由软件资源、硬件资源、联网资源和其它资源的预集成和优化组合构成。硬件资源可以包括但不限于服务器、数据存储设备、服务器、打印机等。软件资源可以包括但不限于计算程序、应用(例如,基于云的应用、企业应用或任何其它应用)、计算机程序产品(例如,软件)、服务(例如,基于云的服务)等。数据资源可以包括但不限于任何可访问的数据对象,诸如文件(例如,联网文件或目录信息)、数据库等。

[0183] 在某些实施例中,分布式事件处理系统710可以包括接收器704和计算节点集群708。接收器704可以被配置为接收持续的输入事件流702并将事件流离散化(划分)为一批或多批特定持续时间(例如,X秒)的事件706用于由分布式事件处理系统710中的计算节点集群708进行的后续处理。每批事件在本文中被称为Dstream。在一些示例中,每个Dstream在内部由接收器704表示为弹性分布式数据集(RDD),其是在指定时间段期间(即,在事件批中)摄取的所有输入数据(事件)流的快照。因此,在一些实施例中,输入数据流702被表示为Dstream的序列,其中每个Dstream在内部被表示为RDD,并且每个RDD包括在特定批间隔期间接收到的事件(元组)。在某些示例中,每个RDD表示不可变的分区的元素集合,这些元素

可以存储在高速缓冲存储器中并且在分布式事件处理系统中并行执行。

[0184] 在某些实施例中，计算节点集群704可以被配置为跨越计算节点集群对包含在每个RDD中的数据进行分区，并且针对在应用中定义的一组查询并行地对数据执行操作，并将处理结果提供给分布式事件处理系统的用户。因此，计算节点集群708可以被配置为跨计算节点集群708来分发对RDD中的事件数据的处理，并且快速且实时地向用户提供与针对事件数据执行应用有关的结果。在实施例中，可以使用Apache® Spark Streaming框架来配置分布式事件处理系统710，以执行持续数据流的分布式和实时处理以及事件处理应用的部署。

[0185] 图8是根据本公开一个实施例的过程800的示例流程图，过程800描述了用于处理事件处理应用以生成应用的公共应用运行时模型的一组操作。在某些实施例中，过程800可以由图3中描述的应用处理引擎(314)中的一个或多个部件(例如，316、318和320)执行。过程800通过接收识别应用的信息在802处开始。这种信息可以包括例如描述应用的各种部件(例如，适配器、处理器、流或事件bean)的信息，诸如配置信息、查询信息和其它类型的信息。如上所述，这种信息可以在如图3中所描述的配置文件(308、310和312)中表达。

[0186] 在804处，该过程包括基于识别应用的信息生成应用的“公共应用运行时模型”。在实施例中，为应用生成“公共应用运行时模型”可以包括将应用表示为一组一个或多个配置块，其中每个配置块表示具有相关联的元数据的事件bean。配置块可以包括如关于图4所描述的入站套接字事件bean、出站套接字事件bean、持续查询语言(CQL)处理器事件bean，或者一个或多个信道事件bean。

[0187] 在806处，该过程包括将应用的“公共应用运行时模型”转换成应用的第一通用表示。应用的第一通用表示可以被配置为在多个目标事件处理系统中的第一目标事件处理系统中执行。在一个示例中，将应用的“公共应用运行时模型”转换成应用的第一通用表示可以包括将公共应用运行时模型中的配置块转换成应用的部件的运行时DAG。图5中示出了为应用生成的部件的运行时DAG的示例。

[0188] 在一些实施例中，在808处，该过程包括将应用的第一通用表示发送到第一目标事件处理系统以供第一目标事件处理系统执行。在一个示例中，第一目标事件处理系统是分布式事件处理系统。

[0189] 图9是根据本公开另一个实施例的过程900的示例流程图，该过程900描述用于处理事件处理应用以生成应用的公共应用运行时模型的一组操作。在某些实施例中，过程900可以由图3中描述的应用处理引擎(314)中的一个或多个部件(例如，316、318和320)执行。过程900通过接收识别应用的信息在902处开始。如上所述，这种信息可以包括描述应用的各种部件的配置信息、查询信息和其它类型的信息。

[0190] 在904处，该过程包括基于识别应用的信息生成应用的“公共应用运行时模型”。在实施例中，为应用生成“公共应用运行时模型”可以包括将应用表示为一组一个或多个配置块，其中每个配置块表示具有相关联元数据的事件bean。配置块可以包括，如关于图4所描述的入站套接字事件bean、出站套接字事件bean、持续查询语言(CQL)处理器事件bean，或者一个或多个信道事件bean。

[0191] 在906处，该过程包括将应用的“公共应用运行时模型”转换成应用的第二通用表示。应用的第二通用表示可以被配置为在多个目标事件处理系统中的第二目标事件处理系

统中执行。在一些示例中，第二目标事件处理系统可以与第一目标事件处理系统不同。在一个示例中，将应用的“公共应用运行时模型”转换成应用的第二通用表示可以包括将公共应用运行时模型中的配置块转换成应用的目标表示。

[0192] 在一些实施例中，在908处，该过程包括将应用的第二通用表示发送到第二目标事件处理系统以供第二目标事件处理系统执行。在一个示例中，第二目标事件处理系统是Oracle®事件处理器(OEP)系统。

[0193] 事件数据的序列化和解序列化

[0194] 在某些实施例中，所公开的分布式事件处理系统可以被配置为执行经由持续事件流接收到的事件数据的序列化和解序列化。事件数据的序列化和解序列化使得能够将存储器中的复杂数据对象转换成可以传送到分布式事件处理系统中的计算节点的位序列。事件数据的序列化和解序列化使得分布式事件处理系统中的处理节点能够在由分布式事件处理系统处理事件数据之前高效地存储和表示数据。此外，事件数据的序列化和解序列化减少了在分布式事件处理系统中的处理节点之间交换输入事件和输出事件的时延，并且提高了分布式事件处理系统的整体性能。

[0195] 上述技术可以以多种方式以及在许多上下文中实现。参考下面的图1-10提供了若干示例实现方案和上下文，这些图描述了所公开的分布式事件处理系统可以执行与事件处理应用的部署、处理和执行相关的操作的方式的附加细节。

[0196] 图10是图示根据本公开实施例的分布式事件处理系统的部件的简化框图1000。分布式事件处理系统1000可以与图1中描述的分布式事件处理系统110相同或相似。图10中所示的实施例是可以结合本公开实施例的分布式事件处理系统的一个示例。在其它实施例中，分布式事件处理引擎可以具有比图10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配置或布置。这些部件可以用硬件、固件、软件或其组合来实现。在一些实施例中，软件可以存储在存储器(例如，非瞬态计算机可读介质)中、存储器设备或某种其它物理存储器上，并且可以由一个或多个处理单元(例如，一个或多个处理器、一个或多个处理器核、一个或多个GPU等)执行。因此，图10中所示的实施例是用于实现实施例系统的分布式事件处理引擎的一个示例，并不旨在进行限制。

[0197] 在某些实施例中，分布式事件处理系统1002可以包括接收器1004、应用部署模块1008和计算节点集群1012。接收器1006可以能够如图2所示(例如，从事件源204、206或208)接收持续的输入数据流1004，将输入的数据流分成在本文被称为Dstreams的一批或多批事件1010。如上所述，每个Dstream(即，事件批)包括在指定时间段期间摄取的所有的输入数据(事件)流，并且可以在内部由接收器1006将其表示为RDD对象，RDD对象是不可变的分区的元素集合，这些元素可以在分布式事件处理系统1002中的计算节点集群1012中并行执行。

[0198] 应用部署模块1006可以被配置为部署应用(例如，事件处理应用)以供计算节点集群1012中的计算节点处理和执行。本文描述的应用可以指代分布式事件处理系统的(例如，由用户构建的)计算机程序。例如，应用可以包括词计数应用，该应用计数在输入文本集合内对特定词的引用的数量。可以使用例如读取文本集合并计数每个词在每个文本中出现的次数的一个或多个持续查询来构建这种应用。输入文本可以包含例如来自在线应用(诸如Facebook®或Twitter®)的流中接收的短消息。如上所述，可以使用CQL语言配置持

续查询。例如，为了指定要在词计数流传输应用中执行的词计数任务/操作，用户可以编写CQL查询，该查询可以采用如下形式：SELECT count FROM location GROUP BY word。这种查询可以搜集来自指定位置的所有句子、将来自那些句子的独特词分组成不同的组，然后计数每个组中的词的数量。

[0199] 在某些实施例中，应用部署模块1008可以被配置为从分布式事件处理系统的用户接收识别应用的信息。例如，应用部署模块1008可以被配置为经由应用部署模块1008中的应用设计用户界面接收识别应用的信息。识别应用的信息可以包括在应用中定义的一组一个或多个持续查询。识别应用的信息还可以包括与应用相关联的应用参数。应用参数可以包括例如部署类型参数，其指定节点集群1012上的应用的部署类型(例如，“集群型式”)。附加应用参数可以包括与应用的运行时配置相关的参数(例如，要使用的执行器的数量、并行性参数、存储器的尺寸、高可用性参数等)。

[0200] 在接收到与应用相关的信息后，在某些实施例中，应用部署模块1008然后可以被配置为向计算节点集群1012发送指令以在集群中的计算节点上部署应用。在某些示例中，计算节点集群1012可以被配置为将应用部署到计算节点集群1012上的主计算节点1014。主计算节点1014可以被配置为存储应用的“应用上下文”。“应用上下文”可以包括例如应用的内容，诸如应用的拓扑、调度信息、应用参数等。

[0201] 在某些实施例中，主计算节点1014可以被称为“驱动程序”，或者运行/执行应用的应用主设备。驱动程序可以被定义为运行应用的main()函数并为应用创建“应用上下文”的进程。驱动程序可以负责驱动应用和从资源管理器1016请求资源。资源管理器1016可以是为计算节点集群1012上的计算节点获取资源以执行应用的服务。为了在集群上运行/执行应用，主计算节点1014连接到资源管理器1016，资源管理器1016然后为应用分配资源。一旦连接，主计算节点1014就获取集群中的一个或多个计算节点(也称为工作者节点1018)上的一个或多个执行器。执行器是为应用运行计算并存储数据的进程。主计算节点1014将(例如，由JAR文件定义的)应用代码发送到执行器。基于在应用中定义的变换和动作，主计算节点1014可以将任务1020发送到执行器。

[0202] 在某些实施例中，主计算节点1014可以包括DAG生成器1022、DAG调度器1024、任务调度器1026和应用上下文信息1028。如上所述，应用上下文信息1028可以包括关于应用的信息，诸如应用的拓扑、调度信息、应用参数等。DAG生成器1022可以被配置为基于其从接收器接收的RDD对象来定义和/或创建RDD对象的有向无环图(DAG)。在一些示例中，DAG生成器1022可以将RDD对象的DAG表示为在一定时间间隔期间它已经接收到的所有RDD的RDD谱系图。RDD谱系图中的每个RDD对象都维护指向一个或多个父项的指针以及与该RDD对象与其父项的关系类型有关的元数据。RDD谱系图还识别要为每个RDD对象执行的变换的DAG。当主计算节点1014被请求在应用中运行作业时，DAG生成器1022执行变换的DAG。变换可以识别要对RDD对象执行的一个或多个操作，以将RDD对象中的数据从一种形式变换成另一种形式。例如，这些操作可以被定义为应用部署模块1008所进行的应用生成过程的一部分。当对RDD对象应用操作时，获得具有经变换的数据的新RDD对象。对RDD对象执行的操作的示例可以包括例如map(映射)、filter(过滤)、flatMap(平坦映射)、reduce(精简)、union(合并)、groupByKey(按键分组)、distinct(区分)、join(联接)、collect(收集)、count(计数)等。涉及CQL语言的变换的DAG在本文中可以被称为CQL变换。

[0203] DAG调度器1024被配置为基于由DAG生成器生成的RDD谱系图生成物理执行计划。在一个实施例中,DAG调度器1024通过将RDD谱系图拆分为多个阶段来生成物理执行计划,其中基于需要针对每个RDD对象中的数据执行的变换来识别每个阶段。例如,如果要对RDD对象执行的变换包括map变换和reduce变换,那么可以将map变换分组在一起成为单个阶段,并且可以将reduce变换分组在一起成为另一个阶段。然后,DAG调度器1024将阶段提交给任务调度器1026。

[0204] 任务调度器1026将应用(作业)划分为多个阶段。每个阶段由一个或多个任务组成。在一个实施例中,基于RDD对象中的输入数据的分区的数量来确定特定阶段的任务的数量。例如,并且如上所述,DAG调度器1024可以将所有map操作调度到单个阶段。然后将这些阶段传递给任务调度器1026,并且任务调度器1026经由资源管理器启动任务。然后由执行器节点1018执行任务。任务调度器1026识别计算节点集群1012中的、将针对每个RDD对象(即,正在处理的每批事件)执行在应用(作业)中定义的操作的节点。

[0205] 在某些实施例中,当执行器节点1018接收到RDD对象时,如果需要将RDD对象发送到计算节点集群中的其它执行器(工作者)节点,那么执行器节点1018执行RDD对象中的数据的序列化和解序列化。如上所述,RDD对象中的数据的处理可以涉及执行在应用中定义的一个或多个持续查询。在实施例中,由任务调度器1026识别出的执行器节点(例如,1018)可以调用CQL引擎(诸如CQL处理器230)来执行RDD对象中的数据的处理并向主计算节点1014返回处理的结果。在下面关于图11详细讨论执行器节点1018可以在其处理之前执行RDD对象中的数据的序列化和解序列化的方式。

[0206] 图11描绘了根据本公开实施例的、用于执行RDD对象中的数据的序列化和解序列化的过程的高级数据流。在某些示例中,当计算节点集群中的节点(例如,主节点或执行器节点)从接收器接收到RDD对象以进行处理时,可以由该节点执行图10中的一个或多个操作。在一组操作中,接收器1102接收输入数据流并将输入数据流划分为特定持续时间(例如,X秒)的一批或多批事件(元组)。在一个实施例中,并且如关于图10所描述的,每批事件(元组)可以在内部由接收器表示为RDD对象。DAG生成器1104接收RDD对象并创建RDD对象的DAG 1106。如上面在图10中所述,在某些示例中,RDD对象的DAG 1106包括要针对每个RDD对象执行的CQL变换1108。当请求主计算节点(例如,如图10中所示的1014)在应用中运行作业时,DAG生成器1104执行CQL变换1108以处理由RDD对象表示的一组输入元组。在一些示例中,要处理的一组输入元组可以从CQL变换1108的父变换获得。然后,CQL变换1108的子变换调用如在CQL转换1108中所表示的要针对该组输入元组执行的具体操作。

[0207] 在某些示例中,CQL变换1108针对RDD对象中的该组输入元组调用批序列化器进程1112,以执行RDD对象中的数据的序列化。在实施例中,批序列化器进程1112可以由计算节点集群中的正在处理RDD对象的节点(例如,执行器节点)执行。如上所述,RDD对象中的数据表示经由事件流接收的一批输入元组(事件)。批序列化器进程1112将RDD对象中的数据序列化,并且来自批序列化器进程的结果的经序列化的块通过网络发送到CQL引擎1116,以处理该组输入元组。在某些实施例中,正在处理RDD对象的节点可以调用CQL引擎1116来处理RDD对象中的该组输入元组。例如,CQL引擎1116可以是部署在执行器节点上的事件处理引擎(例如,图6中描述的630)。CQL引擎1116可以被配置为接收该组输入元组,基于在应用中定义的处理逻辑(操作/变换)来处理该组输入元组,并且作为该处理的结果生成一组输出

元组。

[0208] 在某些实施例中,CQL引擎1116可以在处理RDD对象中的数据之前针对其从批序列化器进程1112接收的经序列化的数据块调用批解序列化器进程1114。这是因为经序列化的块是适合于通过网络传输的二进制格式或有线格式,并且需要将其解序列化为JAVA对象以使CQL引擎1116能够处理。因此,解序列化进程的结果是一组输入元组,该组输入元组处于能够由CQL引擎1116处理的形式。CQL引擎1116处理该组输入元组并基于处理生成一组输出元组。在某些示例中,CQL引擎1116调用另一个批序列化器进程1118,以序列化该组输出元组,并且序列化的结果是输出元组的经序列化的块。当由DAG生成器接收到序列化的该组输出元组时,CQL变换1108针对接收到的输出元组的经序列化的块调用另一个批解序列化器进程1120。批解序列化器进程1120的结果是一组解序列化的输出元组。CQL变换1108将该组输出元组返回到CQL变换中的子变换,以对RDD对象中的数据执行下一组处理操作。在一些实施例中,然后将该组输出元组1110发送到分布式事件处理系统的用户。

[0209] 在某些实施例中,上述批序列化器进程和批解序列化器进程可以由分布式事件处理系统中的节点集群中的正在处理RDD对象的的节点(例如,执行器节点)所执行的软件模块或指令来执行。批序列化器进程和批解序列化器进程执行的操作的附加细节将在下面关于图12-15详细讨论。

[0210] 图12是根据本公开实施例的过程1200的示例流程图,该过程描述通过其可以对一批事件中包括的数据进行序列化的一组操作。在某些实施例中,过程1200可以由分布式事件处理系统中的批序列化器进程(1112)执行。如上所述,在某些实施例中,当主计算节点(例如,如图10中所示的1014)被请求运行在应用中定义的作业/操作时,批序列化器进程可以由主计算节点调用。如上所述,主计算节点识别分布式计算系统1002中的计算节点集群1012中的节点(例如,执行器节点1018),以针对应用中定义的作业/操作处理一批事件并生成作为处理结果的输出事件的集合。在某些实施例中,处理该批事件可以包括该批事件中的数据的序列化。图12的过程描述了一种技术,通过该技术可以序列化一批事件中的数据。图12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12中所示的步骤更多或更少的步骤。

[0211] 在某些实施例中,每当经由(图10中所示的)任务1020接收到一批事件时,图12中描绘的处理就可以由分布式计算系统1002中的计算节点集群1012中的节点执行。该过程在1202处开始,其在1204处从CQL变换1108接收一批事件。在某些示例中,一批事件中的每个事件可以被称为元组,并且该批事件可被称为一批输入元组或一组输入元组。如上所述,经由事件流接收的每个事件符合与事件流相关联的模式,并且该模式识别经由事件流接收的每个事件的一个或多个属性。

[0212] 例如,持续事件流可以表示与公司销售的产品相关的产品相关信息,其中事件流中的每个事件可以表示物品的订单。这种持续事件流可以包括诸如与物品相关的订单标识符、订单状态和订单量之类的属性。用于这种输入流的模式可以表示为S(时间戳,<orderId>,

[0213] <orderStatus>,<orderAmount>)。因此,经由这种流接收的每个事件将通过时间戳和三个属性来识别。在某些实施例中,事件的一个或多个属性可以被表示为一组输入元组(一批事件)中的一个或多个列,因此,在一些示例中,属性可以指代存储一组输入元组中

的元组(事件)的数据值的列。

[0214] 在1206处,该过程包括识别该批事件中的事件的属性(例如,第一属性)。在1208处,该过程包括识别该属性的数据类型。例如,根据上述订单处理流的示例,1206和1208处的过程可以识别出该属性与事件的“orderId”属性对应并且该属性的数据类型是数值数据类型。在1210处,该过程包括确定属性的数据类型是否是数值数据类型。如果识别出的属性的数据类型是数值数据类型,那么在某些实施例中,该过程前进到1212,以确定要对由该属性表示的数据值执行的第一类型的数据压缩。例如,在1212处,该过程可以包括确定将数值压缩技术(例如,基值压缩、精度约简压缩或精度约简值索引)应用到由该属性表示的数据值。在1214处,作为对由该属性存储的数据值应用数值压缩技术的结果,该过程包括针对属性生成一组经序列化的数据值。在图13A、图13B、图13C和图13D中讨论了针对事件的数值属性生成经序列化的数据值的集合的过程。在1216处,该过程包括存储由该属性表示的经序列化的数据值的集合。

[0215] 在某些实施例中,在1218处,该过程包括确定是否存在需要处理的事件的附加属性。如果存在要处理的附加属性,那么该过程循环回到1206以识别事件批中的事件的下一个属性(例如,第二属性),并且针对该下一个属性执行1208-516处的过程。

[0216] 在某些实施例中,在1210处,如果识别出的属性的数据类型未被确定为数值数据类型,那么在某些实施例中,该过程前进到1220,以确定要对由该属性表示的数据值执行的第二类型的数据压缩。例如,继续上述订单处理流的示例,1206和1208处的过程可以识别事件的第二属性与“orderStatus”属性对应并且这个属性的数据类型是非数值数据类型。在这种情况下,该过程继续到1220,以确定要对由该属性存储的数据值执行的第二类型的数据压缩。在实施例中,第二类型的数据压缩可以与第一类型的数据压缩不同。例如,1220处的过程可以确定要对由该属性存储的数据值应用非数值压缩技术(例如,值索引压缩)。在1214处,该过程包括生成由属性表示的经序列化的数据值的集合,作为对由属性存储的数据值应用非数值压缩技术的结果。在图14中讨论了可以针对事件的非数值属性生成经序列化的数据值的集合的过程。在1216处,该过程包括存储由该属性表示的经序列化的数据值的集合。

[0217] 在某些实施例中,该过程可以再次继续到1218,以确定是否存在要识别和处理的事件的任何附加属性。如果存在更多属性,那么该过程循环回到1206,以识别该批事件中的事件的第三属性。然后,1208处的过程可以包括识别第三属性的数据类型,并且1210处的过程可以包括基于第三属性的数据类型确定要对由第三属性存储的数据值执行的第三类型的数据压缩。例如,继续上述订单处理事件流的示例,可以基于属性的数据类型对由“orderAmount”属性存储的数据值执行第三类型的数据压缩。在某些示例中,当已经识别并处理了事件的所有属性时,该过程在1222处结束。

[0218] 图13A是根据本公开实施例的过程1300的示例流程图,该过程描述了用于针对事件的数值属性生成经序列化的数据值的集合的一组操作。在实施例中,过程1300描述了由图12的1214中的过程执行的操作的附加细节。在某些示例中,过程1300通过将当前缓冲区偏移量存储到正在处理其数据值的列(例如,第一个属性)的当前列号于1302处开始。在1304处,该过程包括获得属性的数据类型(列类型)。例如,在这个例子中,属性的数据类型可以被确定为数值属性。在1306处,该过程包括扫描一组输入元组以获得由该属性表示的

最小值、最大值和独特值集合。在1308处，该过程包括计算存储由该范围(最大值-最小值)的属性表示的数据值所需的位的数量。在1309处，该过程包括确定所需的位的数量是否大于属性的数据类型的位的数量的一半，并且独特值集合的尺寸是否小于输入元组的数量/value_index_threshold，其中value_index_threshold是可配置的。在示例中，value_index_threshold可以被配置为值11作为缺省值。如果是，那么该过程继续到1350以执行精度约简索引值压缩技术。在1310处，该过程包括确定所需的位的数量是否小于列的原始数据类型的位的数量。执行1310处的检查以确保经序列化的块的尺寸不会比原始块增加。这是因为，如果覆盖值范围所需的位大于原始数据所需的位，那么使用值索引技术创建的结果块可能大于原始块尺寸。

[0219] 如果所需的位的数量小于列的原始数据类型的位的数量，那么该过程继续到1312以确定该独特值集合的尺寸是否小于输入元组的数量/2。执行这个确定以确保压缩率足够大。如果有太多独特值，那么在某些示例中，将使用值本身而不是值和值的索引。如果确定所需的位的数量小于列的原始数据类型并且该独特值集合的尺寸小于输入元组的数量/2，那么执行1314-626中描述的过程。

[0220] 例如，在1314处，该过程包括将要对由属性表示的数据值执行的第一类型数据压缩存储为精度约简值索引类型的数据压缩。精度约简技术通过查找值的范围来减少表示来自所使用的值的值的位。所需的位将取决于值的范围。在1316处，该过程包括存储属性的最小数据值。在1318处，该过程包括存储每个最小值的位的数量。在1320处，该过程包括针对其数据值当前正被处理的列(例如，属性)的每个数据值执行1322和1324处的操作。例如，在1322处，该过程包括从该独特数据值集合中获得索引。在1324处，该过程包括将索引存储到缓冲区。在已经处理了列的所有数据值之后，在1326处，该过程包括为元组集合中的每个独特值存储(独特值-最小值)。这些值表示从在步骤1324处存储的索引中索引的实际值。

[0221] 在某些实施例中，如果在1312处的过程确定独特值集合的尺寸不小于输入元组的数量/2，那么在一些实施例中，执行下面的图13B中描述的过程1332-638。在某些实施例中，如果1310处的过程确定所需的位的数量不小于列的原始数据类型，那么在一个实施例中，执行图13C中描述的1342-646中的过程。在某些实施例中，通过针对该属性将经序列化的数据值的集合(即，经序列化的数据块)返回到CQL引擎以处理经由事件流接收的该组元组(即，该批事件)，过程在1328处结束。

[0222] 图13B是根据本公开实施例的过程1350的示例流程图，该过程描述了用于使用精度约简压缩技术针对事件的数值属性生成经序列化的数据值的集合的一组操作。在实施例中，过程1350描述了由图13A的1330中的过程执行的操作的附加细节。在某些示例中，过程1350通过将要对由属性表示的数据值执行的数据压缩的类型存储为精度约简压缩而开始于1332。在1334处，该过程包括存储属性的最小数据值。在1336处，该过程包括存储属性的每个数据值的位的数量。在1338处，该过程包括，对于列的每个数据值，仅对所需的位执行位复制(值-最小值)。例如，输入值的集合(10、11、12)将被存储，其中位00用于值0，值0是(10-10(最小值))的结果，01用于值1，值1是(11-10)的结果，并且02用于值2，值2是(12-10)的结果。位值00、01和02的序列可以存储到字节(8位)00010200中并以十六进制值存储为154。

[0223] 图13C是根据本公开实施例的过程1360的示例流程图，该过程描述了用于使用常

规压缩技术针对事件的数值属性生成经序列化的数据值的集合的一组操作。在实施例中，过程1360描述了由图13A的1340中的过程执行的操作的附加细节。在某些示例中，过程1360通过将要对由属性表示的数据值执行的第一类型数据压缩存储为一般压缩类型而开始于1342。在1344处，该过程包括使用诸如zip或gzip的标准压缩技术来压缩列值的阵列。在1346处，该过程包括存储由属性表示的数据值的经压缩的字节。

[0224] 图13D是根据本公开实施例的过程1370的示例流程图，该过程描述了用于使用精度约简值索引压缩技术针对事件的数值属性生成经序列化的数据值的集合的一组操作。在实施例中，过程1370描述由图13A的1350中的过程执行的操作的附加细节。在某些示例中，过程1370通过将要对属性表示的数据值执行的数据压缩的类型存储为精度约简索引值压缩而开始于1372。在1374处，计算差值集合(例如，值-最小值)。在1376处，该过程包括扫描差值集合中的所有值以获得枚举值集合。在1378处，该过程包括为由差值集合表示的每个数据值计算索引集合。在1380处，该过程包括从索引集合计算最小值和最大值。在1382处，该过程包括存储属性的最小数据值。在1384处，该过程包括存储索引值的每个数据值的位的数量。在1386处，该过程包括对于列的每个数据值，仅针对所需的位执行位复制。在1388处，该过程包括存储枚举的差值的集合。

[0225] 图14是根据本公开实施例的过程1400的示例流程图，该过程描述了用于针对事件的非数值属性生成经序列化的数据值的集合的一组操作。在实施例中，过程1400描述当确定要对由属性(例如，非数值属性)表示的数据值执行第二类型的数据压缩时由图12的过程1214执行的操作的附加细节。过程1400通过将当前缓冲区偏移量存储到正在处理其数据值的列(例如，属性)的当前列号而开始于1402。在1404处，获得属性的数据类型(列类型)。例如，在这个例子中，属性的数据类型被确定为非数值属性。在1406处，该过程包括将要对由属性表示的数据值执行的数据压缩的类型存储为值索引压缩。在这种情况下，枚举输入批内的所有可能值，并使用位置索引而不是多次复制值。

[0226] 在1408处，该过程包括扫描所有输入元组以获得该列的枚举值集合。在1410处，该过程包括计算由列表示的每个数据值的索引集合。在1412处，该过程包括针对存储在列中的每个数据值执行下面在1414-716中描述的操作。在1414处，该过程包括从枚举值集合中获得索引。在1416处，该过程包括将索引存储到缓冲区。在1418处，该过程包括存储该枚举值集合。在1420处，该过程通过将针对该属性的经序列化的数据值的集合(即，经序列化的数据块)返回到CQL引擎以处理经由事件流接收的一组元组(即，一批事件)而结束。

[0227] 图15是根据本公开实施例的、可以基于确定事件流中的事件的属性的数据类型来对事件流数据进行序列化的方式的示例。在下面说明的示例中，事件流表示与公司销售的产品相关的产品相关信息。事件流中的每个事件可以表示物品的订单，并且包括与物品相关的诸如订单标识符、订单状态和订单量之类的属性。用于这种订单事件流的模式可以表示为S(时间戳，<orderId>，<orderStatus>，<orderAmount>)。因此，经由这种流接收的每个事件可以通过时间戳和三个属性来识别。作为示例，经由订单事件流接收的一批事件可以包括以下事件和相关联的时间戳：

[0228] …

[0229] (timestamp_N,10,“open”,100)

[0230] (timestamp_N+1,11,“open”,5000)

[0231] (timestamp_N+2,10,“processing”,100)

[0232] (timestamp_N+3,10,“shipped”,100)

[0233] (timestamp_N+4,11,“processing”,5000)

[0234] (timestamp_N+5,10,“closed”,100)

[0235] (timestamp_N+6,11,“shipped”,5000)

[0236] (timestamp_N+7,11,“closed”,5000)

[0237] 如上所述,在某些实施例中,事件的一个或多个属性可以表示为一组输入元组的表示一批事件的的一个或多个列。因此,在一些示例中,属性可以指代存储一组输入元组中的元组(事件)的数据值的列。与经由订单事件流接收的一批事件对应的一组输入元组的示例可以如下面的表-1所示:

[0238]

元组（事件）	属性 1（第 1 列） 订单 ID	属性 2（第 2 列） 订单状态	属性 3（第 3 列） 订单量
e1	10	开放（open）	100
e2	11	开放（open）	5000
e3	10	处理（processing）	100
e4	10	装运（shipped）	100
e5	11	处理（processing）	5000
e6	10	关闭（closed）	100
e7	11	装运（shipped）	5000
e8	11	关闭（closed）	5000

[0239] 表-1

[0240] 在某些实施例中,通过识别事件的每个属性的数据类型并基于属性的数据类型确定要应用于由每个属性表示的数据值的特定类型的压缩技术来对事件批中的事件进行序列化。例如,可以基于确定第一属性是数值属性而将第一压缩技术应用于事件的第一属性(例如,订单id属性),可以基于确定第二属性是非数值属性而将第二压缩技术应用于事件的第二属性(例如,订单状态属性),并且可以基于确定第三属性是数值属性而将第三压缩技术应用于事件的第三属性(例如,订单量属性)。在某些示例中,第一类型的压缩技术、第二类型的压缩技术和第三类型的压缩技术可以彼此不同。

[0241] 在实施例中,列式存储可以用于存储具有相同数据类型(使得列中的值是相同的数据类型)的属性(列)。在某些实施例中,可以使用基值压缩技术或精度约简压缩技术来压缩存储在数值类型的列中的值。通过查找值的范围,精度约简减少了表示来自所使用的值的值的位。所需的位将取决于值的范围。基值压缩使用最小值作为基值,并且对于其它值存储与基值的差异。例如,表示事件批(10,11,10,10,11,10,11,11)中每个事件的“订单ID”的输入值集合可以被压缩为(10,1)、01001011(二进制)或0x4B(十六进制),其表示值(0,1,0,0,1,0,1,1),从而使用从32个位减少到2个位,这是因为最小值是10并且范围是2。对于另一个示例,可以使用精度约简和值索引技术来压缩表示“订单量”的输入值集合。在这种情况

下,表示订单量的输入值集合(100,5000,100,100,5000,100,5000,5000)可以通过使用精度约简和值索引技术压缩为(100,2,0x10,0x4F)和(0,4900)。输入集合可以表示为(0,4900,0,0,4900,0,4900,4900),基值为100。结果集合的值为二进制的00010000、01001111,十六进制的10和4F,其表示具有到基值表(0,4900)的索引的(0,1,0,0,1,0,1,1)(例如,0指向0并且由于基值为100而进而指向100,并且1指向4900并且由于基值为100而进而指向5000)。

[0242] 在某些实施例中,“值索引压缩”技术可以用于处理存储诸如字符串值之类的非数值值的列的值。在这种情况下,我们枚举一批输入中的所有可能值,并且使用位置的索引而不是多次复制值。例如,如果“订单状态”属性(列)的值为(开放,开放,处理,装运,处理,关闭,装运,关闭),那么对应的枚举独特值将为(开放,处理,装运,关闭)。当列的值按顺序存储在线性缓冲区中时,每个值的索引将为(0,5,17,25),因为缓冲区将具有open/0processing/0shipped/0closed/0,其中/0指示字符串标记的结束。在值的线性缓冲区的情况下,最终的压缩结果是(0,0,5,17,5,25,17,25)。

[0243] 图16是根据本公开实施例的过程1600的示例流程图,该过程描述了通过其可以对一批事件中包括的数据进行解序列化的一组操作。在某些实施例中,过程1600可以由分布式事件处理系统中的批解序列化器进程(420)执行。如上所述,在某些实施例中,当请求主计算节点(例如,如图3中所示的314)运行在应用中定义的作业/操作时,批解序列化器进程可以由主计算节点调用。如上所述,主计算节点识别分布式计算系统302中的计算节点集群312中的节点(例如,执行器节点318),以针对应用中定义的作业/操作来处理一批事件并生成作为处理结果的输出事件的集合。在某些实施例中,处理该批事件可以包括该批事件中的数据的序列化和随后的解序列化。图16的过程描述了一种技术,通过该技术可以对一批事件中的数据进行解序列化。图16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16中所示的步骤更多或更少的步骤。

[0244] 在某些实施例中,过程1600通过接收与一批事件(一组输入元组)中的事件的一个或多个属性对应的经序列化的数据值的集合而开始于1602。在1604处,该过程包括处理与该批事件中的事件的一个或多个属性对应的经序列化的数据值的集合,以生成输出事件的集合。在某些示例中,1604处的过程可以包括在1606处基于该经序列化的数据值的集合生成与该属性对应的解序列化的数据值的集合,并且在1608处针对一组一个或多个持续查询来处理与该属性对应的解序列化的数据值的集合以生成输出事件的第一集合。在1610处,该过程包括将该输出事件的集合发送到分布式事件处理系统的用户。

[0245] 关于下面的图17-14详细讨论生成与一批事件中的事件的属性对应的解序列化的数据值的集合的过程1604。具体而言,图17描述通过其可以生成与一批事件中的事件的一个或多个属性对应的解序列化的数据值的集合的过程。图18描述通过其可以生成与事件的非数值属性对应的解序列化的数据值的集合的过程。图19-14描述通过其可以生成与事件的数值属性对应的解序列化的数据值的集合的过程。

[0246] 图17是根据本公开实施例的过程1700的示例流程图,该过程描述为一批事件中的事件的一个或多个属性生成解序列化的数据值的集合的一组操作。在实施例中,过程1700描述图16的1604中的过程的操作的附加细节。在某些示例中,过程1700通过创建元组的阵列而开始于1702。在1704处,该过程包括识别事件的第一列(第一属性)。在1706处,该过程

包括获得正在处理其数据值的列(例如,第一属性)的当前列号的缓冲区偏移量。在1708处,该过程包括读取属性的压缩类型。例如,这涉及读取由批序列化器进程执行了以对属性的数据值进行序列化的数据压缩类型。在1710处,该过程包括确定应用于属性的数据压缩的类型是否是值索引压缩。如果应用于属性的压缩类型是值索引压缩,那么然后该过程继续到1724以执行图18中描述的过程。

[0247] 在1712处,该过程包括确定应用于属性的数据压缩的类型是否是精度约简压缩。如果应用于属性的压缩类型是精度约简压缩,那么该过程继续到1726以执行图19中描述的过程。

[0248] 在某些实施例中,在1714处,该过程包括确定应用于属性的数据压缩类型的类型是否是精度约简值索引压缩。如果应用于属性的压缩类型是精度约简值索引压缩那么该过程继续到1728以执行图20中描述的过程。

[0249] 在某些实施例中,如果该过程确定应用于属性的压缩类型既不是值索引压缩,也不是精度约简压缩或精度约简值索引压缩,那么该过程继续到1716以确定应用于属性的压缩类型是一般压缩。在1716处,该过程包括执行图21中描述的过程。

[0250] 在1718处,该过程包括确定是否存在要处理的附加属性。如果存在要处理的附加属性,那么该过程循环回到1704以识别并处理事件的下一个属性。如果没有要处理的属性,那么在一些实施例中,该过程将元组的阵列返回到CQL引擎以在1720进一步处理。在某些实施例中,该过程在1722结束。

[0251] 图18是根据本公开实施例的的过程1800的示例流程图,该过程描述用于使用值索引压缩生成与一批事件中的事件的数值属性或非数值属性对应的解序列化的数据值的集合的一组操作。在实施例中,过程1810描述了图17中的过程1724的附加细节,并且当应用于属性的数据压缩的类型(例如,在图17中的1710处)被确定为“值索引”压缩时,执行过程1810。作为示例,可以针对订单事件流中的“订单状态”属性(非数值属性)执行过程1800。

[0252] 使用值索引压缩技术的示例在图8的第2列中示出。经压缩的值具有两个数据集合(0,0,5,17,5,25,17,25)和('开放','处理','装运','关闭')。第二个集合被称为value_arrays(值阵列),因为该集合包含实际值。第一个集合被称为index_values(索引值),因为该集合包括存储在value_arrays中的对实际值的索引值。value_index指代index_values中的各个索引值中的每一个。

[0253] 在某些实施例中,过程1800通过将索引值读取到index_values而开始于1802。在1804处,该过程包括将值阵列读取到value_array。在1806处,该过程包括针对与一组输入元组中的属性对应的每个数据值在1808、1810和1812中执行操作。例如,在1808处,该过程包括从index_values[value_index]获得索引(index)。在1810处,该过程包括从value_array[index]获得值。在1812处,该过程包括将该值设置到元组列tuple[value_index]。

[0254] 图19是根据本公开实施例的过程1900的示例流程图,该过程描述用于使用精度约简压缩技术生成与一批事件中的事件的数值属性对应的解序列化的数据值的集合的一组操作。在实施例中,过程1900描述图17中的过程1726的附加细节,并且当应用于属性的数据压缩的类型(例如,在图17中的1712处)被确定为“精度约简”压缩时执行过程1900。作为示例,可以针对订单事件流中的“订单ID”或“订单量”属性(数值属性)执行过程1900,并且经压缩的结果在图8列1中示出为(10,2,0x4B)。在下面描述的一组操作中,在一个示例中,项

“最小值”具有值17，这是值范围的最小值，并且项“位的数量”具有值2，它是表示值范围的位的数量。

[0255] 在某些实施例中，过程1900通过将最小值读取到base_value(基值)而开始于1902。在1904处，该过程包括读取位的数量。在1906处，该过程包括针对与一组输入元组中的属性对应的每个数据值执行1908和1910中的过程。例如，在1908处，该过程包括将值位读取到value_bits。在1910处，该过程包括将base_value+value_bits设置到元组列tuple[value_index]。

[0256] 图20是根据本公开实施例的过程2000的示例流程图，该过程描述用于使用精度约简值索引生成与一批事件中的事件的数值属性对应的解序列化的数据值的集合的一组操作。在实施例中，过程2000描述图17中的过程1728的附加细节，并且当应用于属性的数据压缩的类型(例如，在图17中的1712处)被确定为“精度约简值索引”压缩时执行过程2000。作为示例，可以针对订单事件流中的“订单ID”或“订单量”属性(数值属性)执行过程2000。图8在第3列中示出了对于“订单量属性”的示例结果，其结果为(100,2,0x10,0x4F)和(0,4900)。

[0257] 在下面描述的一组操作中，项“base value”(基值)指代针对列值的基值的值170，项“number of bits”(位的数量)指代针对索引值的位的数量的值2，并且项“index_values”(索引值)指代用作索引值的值(0x10,0x4F)，并且项“value_array”(值阵列)指代表示差值的集合的(0,4900)。

[0258] 在某些实施例中，过程2000通过将最小值读取到base_value而开始于2002。在2004处，该过程包括读取位的数量。在2006处，该过程包括将索引值读取到index_values。在2008处，该过程包括将值阵列读取到value_array。在2010处，该过程包括对于j←0到value_array.length的每个值将value_array[j]设置为value_array[j]+base_value。在2012处，该过程包括针对与一组输入元组中的属性对应的每个数据值执行2014、2016和2018中的过程。例如，在2014处，该过程包括从index_values[value_index]获得索引(index)。在2016处，该过程包括从value_array[index]获得值。在2018处，该过程包括将该值设置到元组列tuple[value_index]。

[0259] 图21是根据本公开实施例的过程2100的示例流程图，该过程描述用于生成与一批事件中的事件的数值属性或非数值属性对应的解序列化的数据值的集合的一组操作。在实施例中，过程2100描述图17中的过程1716的附加细节，并且当应用于属性的数据压缩的类型(例如，在图17中的1716处)被确定为“一般压缩”技术时执行过程2100。作为示例，可以针对订单事件流中的“订单ID”或“订单量”属性(数值属性)执行过程2100。在下面描述的一组操作中，项“value”是指未压缩的值。

[0260] 在某些实施例中，过程2100通过将块解压缩为值阵列而开始于2102。在2104处，该过程包括针对每个数据值将值阵列中的值设置到与一组输入元组中的属性对应的元组的元组列。

[0261] 在基于微批(micro-batch)的事件处理系统中调度和管理多个CEP引擎

[0262] 近年来，已经开发了数据流管理系统(DSMs)，其可以在潜在无界的实时数据流上以持续的方式执行查询。在新的DSMs当中，这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。这种系统的示例是在Spark®平台上运行的

Spark®Streaming应用。

[0263] 由于在其中有状态处理一般很复杂的系统设计的性质,微批处理流处理具有一些缺点。一个这样的缺点是无法执行“型式匹配”操作。型式匹配是流处理系统应当支持的期望的重要特征,并且型式匹配需要高度有状态的处理,以便运行状态机以从无界事件流检测型式。

[0264] 为了支持完全有状态的查询处理,所公开的技术将CQL查询引擎添加到微批处理流处理中。由于集群中有多于一个CQL引擎,因此必须解决与调度、跟踪和维护局部性(locality)相关的问题。

[0265] 图22是可以在其中实现CQL引擎跟踪器中的调度过程的示例系统或体系架构。在一个实施例中,并且如下面的图22所示,在驱动器(主)2206中公开了CQL引擎跟踪器2202部件,其可以在CQL引擎2212和CQL弹性分布式数据集(RDD)2218之间远程通信。为了启动和在调度,CQL引擎跟踪器2202使用两步调度策略来区分不同的系统环境。为了最大化局部性,在一个实施例中,CQL引擎跟踪器2202使用以下亲和度算法。

[0266] 1.所有CQL引擎2212、2214、2216都由CQL引擎跟踪器2202从驱动器2206启动。不设置CQL引擎与优选位置的关联。

[0267] 2.第一CQLRDD 2218没有优选位置信息。

[0268] 3.调度器2204将尝试使用父RDD的优选位置共同定位到父RDD所在的主机。

[0269] 4.CQLRDD 2218的第一次运行将CQL引擎2212与同一主机2208相关联。

[0270] 5.下一个CQLRDD 2220将根据从步骤4设置的关联信息来设置优选位置信息。

[0271] 6.调度器2204将尝试将CQLRDD运行到它被设置到的优选位置。

[0272] 所公开的技术使得能够在微批处理流处理中具有完全有状态的CQL引擎2212、2214、2216,从而维持CQL引擎和CQLRDD之间的局部性,以及用于启动和重新启动CQL引擎的多步调度算法。此外,与其它基于逐事件的流处理系统相比,所公开的本地亲和度算法提供了最大性能。

[0273] 在某些实施例中,所公开的CQL引擎跟踪器2202负责调度、跟踪和重启集群中的CQL引擎。CQL引擎作为集群中的长时间运行的任务来运行,并且可以作为常规流作业被启动。除了遇到故障情况之外,CQL引擎跟踪器2202不返回。

[0274] 在一些实施例中,可以维护CQL引擎跟踪器2202的以下跟踪信息。

[0275] • state:CQLEngineState-INACTIVE,SCHEDULED,ACTIVE

[0276] -这在CQL引擎2212、2214、2216的整个生命周期中从INACTIVE->SCHEDULED->ACTIVE->INACTIVE改变

[0277] • scheduleLocation:TaskLocation

[0278] -初始调度的位置

[0279] • runningExecutor:ExecutorCacheTaskLocation

[0280] -CQL引擎实际运行的执行器2208位置

[0281] • name:String

[0282] -CQL引擎2212、2214、2216的名称

[0283] • endpoint:RpcEndpointRef

[0284] -CQL引擎2212、2214、2216的远程进程调用(RPC)端点,以远程访问它

[0285] • errorInfo:CQLEngineErrorInfo

[0286] -最后已知的错误信息

[0287] 在实施例中,CQL引擎2212、2214、2216的启动流程可以描述如下:

[0288] • 决定要启动的CQL引擎2212、2214、2216的数量

[0289] • 获得执行器的列表2208、2210、2212

[0290] • 运行轮询调度器,以将CQL引擎2212、2214、2216调度到执行器的列表2208、2210、2212

[0291] • 任务调度器2204启动实际的长时间运行的任务

[0292] • 新启动的CQL引擎调用对CQL引擎跟踪器(例如,CQL引擎跟踪器2202)的“register”(“注册”)RPC调用

[0293] 在某些实施例中,CQL引擎局部性亲和度算法可以通过以下过程来描述:

[0294] 1.所有CQL引擎由CQL引擎跟踪器从驱动器启动。没有设置CQL引擎与优选位置的关联。

[0295] 2.第一个CQLRDD没有优选位置信息。

[0296] 3.调度器将尝试使用父RDD的优选位置共同定位到父RDD所在的主机。

[0297] 4.CQLRDD的第一次运行将CQL引擎关联到同一个主机。

[0298] 5.下一个CQLRDD将根据来自从步骤4设置的关联信息来设置优选位置信息。

[0299] 6.调度器将尝试将CQLRDD运行到它设置的优选位置。

[0300] 在实施例中,CQL引擎重启调度过程可以描述如下:

[0301] • 处置两个情况(拒绝,崩溃)

[0302] • 拒绝-如果调度位置与实际位置不同(未能根据调度来开始)

[0303] • 使用旧的所调度的执行器(减去非活动的执行器)或者使用具有schedulePolicy.rescheduleCQLEngine的新调度的执行器,来获得所调度的执行器

[0304] -选择在调度的位置的列表中仍然存活的执行器

[0305] • 用所调度的执行器启动CQL引擎

[0306] 以下流程示出了以上体系架构的数据流:

[0307] 1.驱动器2206中的CQL引擎跟踪器2202为每个CQL引擎启动长时间运行的任务。CQL引擎跟踪器2202将其RPCEndpoint暴露给长时间运行的任务。

[0308] 2.任务调度器2204将长时间运行的任务执行到集群中的执行器2208、2210、2212

[0309] 3.作为长时间运行的任务的一部分,CQL引擎从执行器2208、2210、2212运行。

[0310] 4.CQL引擎将自己注册到具有CQL引擎的CQLEndpoint的驱动器2206中的CQL引擎跟踪器2202。

[0311] 5.作为流传输DAG的一部分,将存在负责CEP处理的CQLRDD。通过咨询CQL引擎跟踪器,CQLRDD由本地CQL引擎或远程CQL引擎处理。通过RPC调用远程CQL引擎

[0312] 启动CQL引擎

[0313] CQL引擎作为集群中的长时间运行的任务来运行。CQL引擎由CQL引擎跟踪器作为常规作业启动,但除了故障或崩溃之外它永远不会返回并保持运行。CQL引擎跟踪器按照下述算法在集群中启动CQL引擎:

[0314] 1.决定要启动的CQL引擎的数量

[0315] 2.获得执行器的列表

[0316] 3.运行轮询调度器(round-robin scheduler),以将CQL引擎调度到执行器的列表

[0317] 4.任务跟踪器启动实际的长时间运行的任务

[0318] 5.新启动的CQL引擎对CQL引擎跟踪器调用“register”RPC调用

[0319] 6.长时间运行的任务仅在CQL引擎崩溃或其它故障时才返回控制。

[0320] 在步骤#1处,决定要启动的集群中的CQL引擎的数量。集群中的CQL引擎的缺省数量与集群中的执行器的数量相同。因此,一个CQL引擎从每个执行器运行。可以配置这个最大数量CQL引擎。

[0321] 在步骤#2处,从集群中检索执行器信息的列表(执行器主机和执行器id)。

[0322] 在步骤#3处,轮询调度器将执行器指派给CQL引擎。

[0323] 在步骤#4处,为每个CQL引擎启动长时间运行的任务。任务调度器使用所调度的执行器信息(执行器主机和id)在所调度的执行器中启动CQL引擎。

[0324] 在步骤#5处,新启动的CQL引擎对CQL引擎跟踪器调用“register”RPC调用。这个步骤发起来自CQL引擎跟踪器的跟踪过程,如下所示。

[0325] 在步骤#6处,CQL引擎的故障或崩溃触发来自CQL引擎跟踪器的恢复过程,如下所示。

[0326] 跟踪CQL引擎

[0327] 在一些实施例中,可以由CQL引擎跟踪器为每个CQL引擎维护以下跟踪信息

[0328] state:CQLEngineState

[0329] scheduleLocation:TaskLocation

[0330] runningExecutor:ExecutorCacheTaskLocation

[0331] name:String

[0332] endpoint:RpcEndpointRef

[0333] errorInfo:CQLFngineErrorInfo

[0334] ‘state’保持CQL引擎的状态。它在CQL引擎的整个生命周期中从INACTIVE->SCHEDULED->ACTIVE->INACTIVE改变。INACTIVE是在CQL引擎被CQL引擎跟踪器跟踪之前的初始状态。SCHEDULED是CQL引擎被调度到要在执行器中执行时的状态。ACTIVE是CQL引擎实际从执行器运行时的状态。

[0335] ‘scheduledLocation’保持执行CQL引擎的调度的位置。

[0336] ‘runningExecutor’保持CQL引擎实际在其处运行的执行器位置。

[0337] ‘name’是作为标识符的CQL引擎的名称。

[0338] ‘endpoint’是与之通信的RPCEndpoint。

[0339] ‘errorInfo’是CQL引擎的最后已知错误信息。

[0340] CQL引擎的恢复

[0341] 长时间运行的任务仅在CQL引擎崩溃或其它故障时才将控制返回到CQL引擎跟踪器。CQL引擎跟踪器使用以下CQL引擎重启调度过程以便重启CQL引擎。从两种情况调用重启调度过程:Rejected(拒绝)和Crashed(崩溃)。

[0342] 崩溃是当运行的CQL引擎崩溃或长时间运行的任务返回时有任何故障的情况。拒绝是当调度的位置与实际位置不同时(例如,无法从调度的执行器启动并通过任务调度器

从不同的执行器启动)的情况。这可能由于来自集群的资源问题而发生。

[0343] 在实施例中,CQL引擎重启调度过程可以描述如下:

[0344] 1.使用集群中的旧的所调度的执行器(减去非活动的执行器)和新执行器获得候选执行器的列表

[0345] 2.在候选执行器的列表中选择仍然存活的执行器

[0346] 3.用所调度的执行器启动长时间运行的任务,该长时间运行的任务启动CQL引擎

[0347] 局部性亲和度算法

[0348] 为了支持水平可缩放性,输入数据集被分区并使用并行化分布式数据处理进行处理。CQL引擎可以使用(queryId,partitionId)与CQL引擎的亲和度或关联来处理多个分区。为了以用于在执行器之间发送数据的最小网络流量来对变换进行优化,需要在最大化局部性的情况下创建这种亲和度。为了最大化局部性,在一个实施例中,CQL引擎跟踪器使用以下亲和度算法。

[0349] 1.所有CQL引擎都由CQL引擎跟踪器从驱动器启动。没有设置CQL引擎与优选位置的关联。

[0350] 2.第一个CQLRDD没有优选位置信息。

[0351] 3.Spark®调度器将尝试使用父RDD的优选位置来共同定位到父RDD所在的执行器。

[0352] 4.CQLRDD向CQL引擎跟踪器调用“getCQLEngine”RPC。

[0353] 5.CQLRDD的分区的第一次计算将(partitionId,queryId)与CQL引擎关联到CQLRDD的同一执行器。

[0354] 6.(partitionId,queryId)到CQL引擎的优选位置映射在CQL引擎跟踪器中维护。

[0355] 7.来自该关联的CQL引擎返回到CQLRDD,并且RDD由该CQL引擎处理。

[0356] 8.下一个CQLRDD将根据从步骤5设置的关联信息来设置优选位置信息。

[0357] 9.Spark®调度器将尝试将CQLRDD运行到其被设置到的优选位置。

[0358] 10.CQLRDD向CQL引擎跟踪器调用“getCQLEngine”RPC,并且(partitionId,queryId)应当已经是同一个执行器。

[0359] 图23是根据本公开实施例的过程2300的示例流程图,该过程描述用于在基于微批处理的事件处理系统内调度和管理多个CEP引擎的一组操作。在实施例中,过程2300描述图22中描述的操作的附加细节。在某些示例中,过程2300通过在CQL引擎的集群中启动第一CQL引擎而开始于2302。可以使用CQL引擎跟踪引擎启动第一CQL引擎以及附加的CQL引擎。在2304处,CQL引擎跟踪引擎还可以调度第一CQL引擎以处理与应用相关的一批持续输入事件流。在2306处,CQL引擎跟踪引擎还可以跟踪要被调度以执行的第一CQL引擎。在2308处,CQL引擎跟踪引擎还可以执行第一CQL引擎来处理该批持续输入事件流以生成与该应用相关的输出事件的集合。

[0360] 使用Group By和Object ID字段的自动数据分区和并行性

[0361] 近年来,已经开发了数据流管理系统(DSMs),该系统可以在潜在无界的实时数据流上以持续的方式执行查询。例如,典型的DSMS可以接收一个或多个数据流、针对数据流注册查询,并在新数据出现在流中时持续地执行查询。由于这种类型的持续查询是长时间运

行的,因此DSMS可以向客户端提供持续的更新结果流。

[0362] DSMS中的典型应用被设计为操作或变换的有向无环图(DAG)形状的“拓扑”。该拓扑充当数据变换管道。

[0363] 包括Apache Storm、Spark Streaming和Flink在内的大多数流处理系统为应用开发人员提供应用编程接口(API),以使用诸如Java、Scala或Clojure之类的不同编程语言来构建拓扑。

[0364] API有利于程序员构建流处理应用,但是由于代码生成层的复杂性,对于诸如为用户生成流处理应用的Stream Analytics之类的代码生成系统而言,它是相对复杂的。

[0365] 图24是示例体系架构2400,其中数据变换的输入管道可以输入到管道分析器2402中并由阶段分类模块2404分类。在一些示例中,Stream Analytics的代码生成层负责通过分析管道阶段来自动确定数据变换管道中的并行性。数据变换管道由各个阶段组成,其中每个阶段根据阶段定义执行具体的变换。聚合器阶段对传入的流数据计算实时聚合。如果可以在节点的集群上处理管道的阶段,那么可以优化数据变换管道处理。

[0366] 为了计算节点的集群上的阶段,期望自动确定阶段操作的并行性特点,然后创建变换的DAG,其中可以通过最大化并行性来在集群节点的集合上完成变换的计算。

[0367] 在一个实施例中,构建数据流管理系统(DSMS),其分析由用户设计的数据变换管道、导出用于各个阶段的分区标准,并生成变换的经优化的DAG,其中每个阶段可以在集群节点的集合上运行。

[0368] 在某些实施例中,以下阶段可以包括在由Stream Analytics平台设计的管道中。

[0369] 1)查询

[0370] 2)业务规则

[0371] 3)空间

[0372] 4)型式

[0373] 管道可以由上述类型的一个或多个阶段组成。

[0374] 样本管道的示例如下所示:

[0375] 输入->查询->查询->空间->输出

[0376] 在一些示例中,用户创建管道以实现期望的业务逻辑。在设计管道时,用户可以选择管道的每个阶段并配置管道的阶段属性。在一些示例中,阶段的配置属性成为阶段元数据。

[0377] 如果阶段类型是查询,那么所公开的技术确定阶段的自动数据分区。为了将用户创建的管道变换成本机运行时变换的DAG,Stream Analytics平台可以执行以下步骤:

[0378] 1)从源到汇点遍历管道。

[0379] 2)对于每个阶段

[0380] i)确定阶段类型

[0381] ii)如果阶段类型是“查询”,那么平台标记是否可以以分布式方式为这个阶段计算变换

[0382] a)确定与查询阶段关联的CQL查询。

[0383] b)将CQL查询解析为口令(token)。

[0384] c)对解析后的查询执行CQL查询的语义分析。

[0385] d)使用各种规则确定查询分类

[0386] 这些规则将持续查询分为以下类别：

[0387] 无状态、半有状态、完全有状态

[0388] e)如果查询是无状态的，那么将该阶段标记为要被分区而没有任何分区属性(标准)2406。以这种方式，阶段将取决于父阶段的分区标准。

[0389] f)如果查询是有状态的，那么将该阶段标记为是未分区的2408。以这种方式，该阶段只以仅在节点的单个集群上执行。

[0390] g)如果查询是半有状态的，那么将该阶段标记为要用分区属性分区2410。分区属性将从步骤2.ii.d的结果中获得。以这种方式，可以在自动确定的分区属性上对阶段计算进行分区。

[0391] 3)对于每个阶段，生成数据变换管道的DAG中的变换。

[0392] i)如果阶段被标记为要被分区而没有任何分区属性2406，那么在DAG中为这个阶段生成变换而没有任何重新分区变换。阶段的分区的数量将通过来自前一阶段的分区的数量确定。

[0393] ii)如果阶段被标记为要用分区属性分区2410，那么在DAG中利用重新分区变换为这个阶段生成变换。针对重新分区变换的输入将是分区属性和分区的数量。重新分区变换将用新的分区标准对传入的事件流重新分区。

[0394] iii)如果阶段被标记为是未分区的2408，那么在DAG中用重新分区变换然后是阶段变换来为该阶段生成变换。针对重新分区变换的输入将是分区属性和将为1的分区的数量。重新分区变换将把已分区/未分区的流重新分区为单个分区。

[0395] 在某些管道中，如果系统没有足够的元数据或者如果不能从查询分析确定分区，那么系统将对象id标记为分区属性2412。

[0396] 如果阶段类型是空间并且DSMS正在处理移动对象的地理位置事件流，其中每个对象具有唯一身份，那么Stream Analytics平台将对象id标记为该阶段的分区属性，并在DAG中用分区变换然后是阶段变换为这个阶段生成变换。

[0397] 所公开技术的实施例提供以下特征：

[0398] -执行管道阶段的元数据扫描，并基于CQL查询分类对阶段进行分类。

[0399] -通过基于持续查询语言对查询执行语义分析来自动确定分区属性。

[0400] -通过使用分区来生成变换的DAG。

[0401] 现有技术涉及为用户使用管道数据变换系统来明确定义管道阶段的并行性特点。如果未指定，那么系统可能处理管道阶段而没有完全利用计算资源。

[0402] 所公开的技术通过分析管道流处理系统的阶段来自动确定数据分区标准。这显著降低了为Stream Analytics平台设计数据处理管道的复杂性。

[0403] 图25是过程2500的示例流程图，其描述用于分布式事件处理系统中的数据分区和并行性的一组操作。在实施例中，过程2500描述图24中描述的操作的附加细节。在某些示例中，过程2500通过确定由事件处理系统处理的持续查询语言(CQL)查询的阶段而开始于2502。在2504处，系统可以被配置为确定与该阶段相关联的阶段类型。在一些示例中，过程2500可以通过至少部分地基于阶段类型确定要为阶段计算的变换而在2506处继续。过程2500还可以在2508处至少部分地基于多个规则来确定CQL查询的分类。在一些示例中，过程

2500可以包括在2510处通过对阶段应用分区标准将阶段标记为分区阶段或未分区阶段。此外，在一些示例中，过程2500可以在2512处至少部分地基于阶段的分区标准生成阶段的数据变换管线的有向无环图(DAG)中的变换。在2514处，过程2500可以至少部分地基于该变换来确定阶段的分区。过程2500还可以至少部分地基于所确定的分区来处理CQL查询。

[0404] 说明性系统

[0405] 图26-28示出了根据各种实施例的用于实现本公开的方面的示例环境的方面。图26描绘了用于实现本公开实施例的分布式系统2600的简化图。在所示实施例中，分布式系统2600包括一个或多个客户端计算设备2602、2604、2606和2608，这些客户端计算设备被配置为通过(一个或多个)网络2610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)等等。服务器2612可以经由网络2610与远程客户端计算设备2602、2604、2606和2608通信地耦合。

[0406] 在各种实施例中，服务器2612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2612还可以提供其它服务，或者软件应用可以包括非虚拟和虚拟环境。在一些实施例中，这些服务可以作为基于web的或云服务或者在软件即服务(SaaS)模型下提供给客户端计算设备2602、2604、2606和/或2608的用户。操作客户端计算设备2602、2604、2606和/或2608的用户又可以利用一个或多个客户端应用与服务器2612交互以利用由这些部件提供的服务。

[0407] 在图26描绘的配置中，系统2600的软件部件2618、2620和2622被示出为在服务器2612上实现。在其它实施例中，系统2600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2602、2604、2606和/或2608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2600不同。因此，图26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，而不是限制性的。

[0408] 客户端计算设备2602、2604、2606和/或2608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)，其运行诸如Microsoft Windows **Mobile®** 之类的软件和/或诸如iOS、Windows Phone、Android、BlackBerry 26，Palm OS之类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，作为示例，运行各种版本的Microsoft **Windows®**、Apple **Macintosh®** 和/或Linux操作系统的个人计算机和/或膝上型计算机。客户端计算设备可以是运行任何各种商用的 **UNIX®** 或类UNIX操作系统(包括但不限于诸如像Google Chrome OS的各种GNU/Linux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提供(一个或多个)网络2610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)。

[0409] 虽然图26中的分布式系统2600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(例如具有传感器的设备等)可以与服务器2612交互。

[0410] 分布式系统2600中的(一个或多个)网络2610可以是对本领域技术人员熟悉的可以利用任何各种可用协议支持数据通信的任何类型的网络,其中各种协议包括但不限于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2610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)2602.11协议套件、**Bluetooth®**、和/或任何其它无线协议下操作的网络)和/或这些和/或其它网络的任意组合。

[0411] 服务器2612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2612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,以维护用于服务器的虚拟存储设备。虚拟网络可以由服务器2612利用软件定义的联网来控制。在各种实施例中,服务器2612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2612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对应。

[0412] 服务器2612可以运行包括以上讨论的任何操作系统的操作系统,以及任何商用的服务器操作系统。服务器2612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可从Oracle、Microsoft、Sybase、IBM(国际商业机器)等商业获得的数据库服务器。

[0413] 在一些实现方案中,服务器2612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2602、2604、2606和2608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持续数据流接收到的**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2612还可以包括经由客户端计算设备2602、2604、2606和2608的一个或多个显示设备显示数据馈送和/或实时事件的一个或多个应用。

[0414] 分布式系统2600还可以包括一个或多个数据库2614和2616。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2614和2616可以驻留在各种位置中。举例来说,数据库2614和2616中的一个或多个可以驻留在服务器2612本地(和/或驻留在服务器2612中)的非瞬态存储介质上。可替代地,数据库2614和2616可以远离服务器2612并经由基于网络的或专用的连接与服务器2612通信。在一组实施例中,数据库2614和2616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2612的功能的任何必要文件可以适当地本地存储在服务器2612上和/或远程存储。在一组实施例中,数据库2614和2616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型数据库,诸如由Oracle提供的数据库。

[0415] 一些附图中描绘的系统可以以各种配置提供。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。

[0416] 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，并以某种方式分配给其客户。例如，云服务提供商(诸如加利福尼亚州Redwood Shores的Oracle Corporation)可以提供各种类型的云服务，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一套按需应用(诸如Oracle Fusion应用)的功能。SaaS服务使客户能够利用在云基础设施系统上执行的应用，而无需客户为应用购买软件。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、网络和其它基本计算资源，用于让客户利用SaaS平台和PaaS平台提供的服务。

[0417] 图27是根据本公开的实施例的系统环境2700的一个或多个部件的简化框图，通过该系统环境2700，由实施例系统的一个或多个部件提供的服务可以作为云服务提供。在所示实施例中，系统环境2700包括可以由用户使用以与提供云服务的云基础设施系统2702交互的一个或多个客户端计算设备2704、2706和2708。客户端计算设备可以被配置为操作客户端应用，诸如web浏览器、专有客户端应用(例如，Oracle Forms)或某种其它应用，这些应用可以由客户端计算设备的用户用来与云基础设施系统2702交互以使用由云基础设施系统2702提供的服务。

[0418] 应该认识到的是，图中描绘的云基础设施系统2702可以具有除了所描绘的那些之外的其它部件。另外，图中所示的实施例仅是可以结合本发明的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2702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配置或布置。

[0419] 客户端计算设备2704、2706和2708可以是与上面针对502、504、506和508所描述的设备类似的设备。

[0420] 虽然示例系统环境2700被示出具有三个客户端计算设备，但是任何数量的客户端计算设备可以被支持。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2702交互。

[0421] (一个或多个)网络2710可以促进客户端2704、2706和2708与云基础设施系统2702之间的数据通信和交换。每个网络可以是本领域技术人员所熟悉的可以使用各种商业上可获得的协议(包括上面针对(一个或多个)网络2710所描述的那些协议)中的任何一种支持数据通信的任何类型的网络。

[0422] 云基础设施系统2702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2712所描述的那些计算机和/或服务器。

[0423] 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许多服务，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、被

托管的办公室(office)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云基础设施系统的用户的需要。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，从云服务提供商的系统经由通信网络(诸如互联网)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。

[0424] 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、被托管的数据库、被托管的Web服务器、软件应用或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那样。例如，服务可以包括通过互联网对云上的远程存储装置进行密码保护的访问。作为另一个示例，服务可以包括基于Web服务的被托管的关系数据库和脚本语言中间件引擎，以供联网的开发人员私有使用。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。

[0425] 在某些实施例中，云基础设施系统2702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递送给客户的应用、中间件和数据库服务产品的套件。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。

[0426] 在各种实施例中，云基础设施系统2702可以适于自动供应、管理和跟踪客户对由云基础设施系统2702供给的服务的订阅。云基础设施系统2702可以经由不同的部署模型来提供云服务。例如，可以依据公共云模型提供服务，其中云基础设施系统2702被销售云服务的组织拥有(例如，被Oracle拥有)，并且服务对一般公众或不同行业的企业可用。作为另一个示例，可以依据私有云模型来提供服务，其中云基础设施系统2702仅针对单个组织操作，并且可以为该组织内的一个或多个实体提供服务。还可以依据社区云模型来提供云服务，其中云基础设施系统2702和由云基础设施系统2702提供的服务由相关社区中的若干组织共享。云服务还可以依据混合云模型被提供，该混合云模型是两个或更多个不同模型的组合。

[0427] 在一些实施例中，由云基础设施系统2702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其它服务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2702提供的一个或多个服务。云基础设施系统2702然后执行处理以提供客户的订阅订单中的服务。

[0428] 在一些实施例中，由云基础设施系统2702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落入SaaS类别的云服务。例如，SaaS平台可以提供在集成开发和部署平台上构建和递送按需应用套件的能力。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销售绩效管理、企业集成和业务灵活性的解决方案的服务。

[0429] 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

可以被配置为提供落入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)能够在共享的公共体系架构上整合现有应用以及充分利用平台提供的共享服务来构建新应用的能力的服务。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架构系统提供的PaaS服务,而无需客户购买单独的许可和支持。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。

[0430] 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。在云基础设施系统中,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,并且Java云服务可以为客户提供部署Java应用的平台。

[0431] 各种不同的基础设施服务可以由云基础设施系统中的IaaS平台提供。基础设施服务促进底层计算资源(诸如存储装置、网络和其它基础计算资源)的管理和控制,以供客户利用由SaaS平台和PaaS平台提供的服务。

[0432] 在某些实施例中,云基础设施系统2702还可以包括基础设施资源2730,用于向云基础设施系统的客户提供用于提供各种服务的资源。在一个实施例中,基础设施资源2730可以包括预先集成和优化的硬件(诸如服务器、存储装置和联网资源)的组合,以执行由PaaS平台和SaaS平台提供的服务。

[0433] 在一些实施例中,云基础设施系统2702中的资源可以由多个用户共享并且根据需要动态重新分配。此外,可以将资源分配给在不同时区的用户。例如,云基础设施系统2730可以使在第一时区中的第一组用户能够在指定的小时数内利用云基础设施系统的资源,并且然后使相同资源能够被重新分配给位于不同时区的另一组用户,从而使资源的利用率最大化。

[0434] 在某些实施例中,可以提供由云基础设施系统2702的不同部件或模块以及由云基础设施系统2702提供的服务共享的多个内部共享服务2732。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、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。

[0435] 在某些实施例中,云基础设施系统2702可以提供云基础设施系统中的云服务(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2702接收到的客户订阅等的能力。

[0436] 在一个实施例中,如图中所绘出的,云管理功能可以由一个或多个模块提供,诸如订单管理模块2720、订单编排模块2722、订单供应模块2724、订单管理和监视模块2726,以及身份管理模块2728。这些模块可以包括一个或多个计算机和/或服务器或者使用一个或多个计算机和/或服务器来提供,这些计算机和/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。

[0437] 在示例性操作2734中,使用客户端设备(诸如客户端设备2704、2706或2708)的客

户可以通过请求由云基础设施系统2702提供的一个或多个服务并且下订阅由云基础设施系统2702供应的一个或多个服务来的订单来与云基础设施系统2702交互。在某些实施例中，客户可以访问云用户界面(UI)(云UI 2712、云UI 2714和/或云UI 2716)并经由这些UI下订阅订单。云基础设施系统2702响应于客户下订单而接收到的订单信息可以包括识别客户以及客户想要订阅的云基础设施系统2702供应的一个或多个服务的信息。

[0438] 在客户下订单之后，经由云UI 2712、2714和/或2716接收订单信息。

[0439] 在操作2736处，订单存储在订单数据库2718中。订单数据库2718可以是由云基础设施系统2718操作和与其它系统元件一起操作的若干数据库之一。

[0440] 在操作2738处，订单信息被转发到订单管理模块2720。在一些情况下，订单管理模块2720可以被配置为执行与订单相关的计费和记账功能，诸如验证订单、以及在验证后预订订单。

[0441] 在操作2740处，将关于订单的信息传送到订单编排模块2722。订单编排模块2722可以利用订单信息为客户下的订单编排服务和资源的供应。在一些情况下，订单编排模块2722可以使用订单供应模块2724的服务来编排资源的供应以支持所订阅的服务。

[0442] 在某些实施例中，订单编排模块2722使得能够管理与每个订单相关联的业务过程并应用业务逻辑来确定订单是否应该进行到供应。在操作2742处，在接收到新订阅的订单时，订单编排模块2722向订单供应模块2724发送分配资源并配置履行订阅订单所需的那些资源的请求。订单供应模块2724使得能够为客户订购的服务分配资源。订单供应模块2724提供在由云基础设施系统2700提供的云服务和用于供应用于提供所请求的服务的资源的物理实现层之间的抽象层。因此，订单编排模块2722可以与实现细节(诸如服务和资源是否实际上即时供应或预先供应并仅在请求后才分配/指派)隔离。

[0443] 在操作2744处，一旦供应了服务和资源，就可以通过云基础设施系统2702的订单供应模块2724向客户端设备2704、2706和/或2708上的客户发送所提供的服务的通知。在操作2746处，订单管理和监视模块2726可以管理和跟踪客户的订阅订单。在一些情况下，订单管理和监视模块2726可以被配置为收集订阅订单中的服务的使用统计信息，诸如，所使用的存储量、传输的数据量、用户的数量，以及系统运行时间量和系统停机时间量。

[0444] 在某些实施例中，云基础设施系统2700可以包括身份管理模块2728。身份管理模块2728可以被配置为提供身份服务，诸如云基础设施系统2700中的访问管理和授权服务。在一些实施例中，身份管理模块2728可以控制关于希望利用由云基础设施系统2702提供的服务的客户的信息。这样的信息可以包括认证这些客户的身份的信息以及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)执行哪些动作的信息。身份管理模块2728还可以包括对关于每个客户的描述性信息以及关于如何和由谁来访问和修改这些描述性信息的管理。

[0445] 图28图示了可以用于实现本公开实施例的示例性计算机系统2800。在一些实施例中，计算机系统2800可以用于实现上述各种服务器和计算机系统中的任何一种。如图28所示，计算机系统2800包括各种子系统，包括经由总线子系统2802与多个外围子系统通信的处理单元2804。这些外围子系统可以包括处理加速单元2806、I/O子系统2808、存储子系统2818和通信子系统2824。存储子系统2818可以包括有形计算机可读存储介质2822和系统存储器2810。

[0446] 总线子系统2802提供用于使计算机系统2800的各种部件和子系统按照期望彼此通信的机制。虽然总线子系统2802被示意性地示为单条总线，但是总线子系统的可替代实施例可以利用多条总线。总线子系统2802可以是若干种类型的总线结构中的任何一种，包括存储器总线或存储器控制器、外围总线和利用任何各种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，等等。

[0447] 处理子系统2804控制计算机系统2800的操作并且可以包括一个或多个处理单元2832、2834等。处理单元可以包括一个或多个处理器，其中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2804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2804的处理单元中的一些或全部可以利用定制电路来实现，诸如专用集成电路(ASIC)或现场可编程门阵列(FPGA)。

[0448] 在一些实施例中，处理子系统2804中的处理单元可以执行存储在系统存储器2810中或计算机可读存储介质2822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2810中和/或计算机可读存储介质2822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2804可以提供上述各种功能，用于响应于使用型式而动态地修改文档(例如，网页)。

[0449] 在某些实施例中，可以提供处理加速单元2806，用于执行定制的处理或用于卸载由处理子系统2804执行的一些处理，以便加速由计算机系统2800执行的整体处理。

[0450] I/O子系统2808可以包括用于向计算机系统2800输入信息和/或用于从或经由计算机系统2800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2800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示设备、结合到显示器中的触摸板或触摸屏、滚轮、点拨轮、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用户能够控制输入设备并与其交互的诸如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、提供用于接收利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，当拍摄图片和/或进行菜单选择时的“眨眼”)并将眼睛姿势转换为到输入设备(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用户能够通过语音命令与语音识别系统(例如，**Siri®**导航器)交互的语音识别感测设备。

[0451] 用户接口输入设备的其它示例包括但不限于三维(3D)鼠标、操纵杆或指示杆、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、数字摄像机、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。

[0452] 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)、诸如利用液晶显示器(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2800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。

[0453] 存储子系统2818提供用于存储由计算机系统2800使用的信息的储存库或数据存储。存储子系统2818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结构。当由处理子系统2804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2818中。软件可以由处理子系统2804的一个或多个处理单元执行。存储子系统2818也可以提供用于存储根据本公开使用的数据的储存库。

[0454] 存储子系统2818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28所示，存储子系统2818包括系统存储器2810和计算机可读存储介质2822。系统存储器2810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。在一些实现方案中，包含帮助在诸如启动期间在计算机系统2800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2804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2810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)。

[0455] 作为示例而非限制，如在图28中所绘出的，系统存储器2810可以存储应用程序2812，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2814和操作系统2816。作为示例，操作系统2816可以包括各种版本的Microsoft **Windows®**、Apple **Macintosh®**和/或Linux操作系统、各种商用 **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®** 260S和 **Palm®** OS操作系统的移动操作系统。

[0456] 计算机可读存储介质2822可以存储提供一些实施例的功能的编程和数据结构。当由处理子系统2804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2818中。作为示例，计算机可读存储介质2822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)盘或其它光学介质的光盘驱动器。计算机可读存储介质2822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2822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DRAM和基于闪存存储器的SSD的组合的

混合SSD。计算机可读介质2822可以为计算机系统2800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。

[0457] 在某些实施例中，存储子系统2800也可以包括计算机可读存储介质读取器2820，其可以进一步连接到计算机可读存储介质2822。可选地，与系统存储器2810一起和组合，计算机可读存储介质2822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。

[0458] 在某些实施例中，计算机系统2800可以提供对执行一个或多个虚拟机的支持。计算机系统2800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配存储器、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，其可以与由计算机系统2800执行的其它虚拟机执行的操作系统相同或不同。相应地，多个操作系统可以潜在地由计算机系统2800并发地运行。每个虚拟机一般独立于其它虚拟机运行。

[0459] 通信子系统2824提供到其它计算机系统和网络的接口。通信子系统2824用作用于从计算机系统2800的其它系统接收数据和向其发送数据的接口。例如，通信子系统2824可以使计算机系统2800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和发送信息到客户端设备。此外，通信子系统2824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。

[0460] 通信子系统2824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1724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、或其任意组合)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2824可以提供有线网络连接(例如，以太网)。

[0461] 通信子系统2824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2824可以以结构化和/或非结构化的数据馈送2826、事件流2828、事件更新2830等形式接收输入通信。例如，通信子系统1724可以被配置为实时地从社交媒体网络的用户和/或诸如**Twitter®**馈送、**Facebook®**更新、诸如丰富站点摘要(RSS)馈送的web馈送的其它通信服务接收(或发送)数据馈送2826，和/或来自一个或多个第三方信息源的实时更新。

[0462] 在某些实施例中，通信子系统2824可以被配置为以持续数据流的形式接收本质上可能是持续的或无界的没有明确结束的数据，其中持续数据流可以包括实时事件的事件流2828和/或事件更新2830。生成持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。

[0463] 通信子系统2824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2826、事件流2828、事件更新2830等，其中所述一个或多个数据库可以与耦合到计算机系统2800的一个或多个流数据源计算机通信。

[0464] 计算机系统2800可以是各种类型中的一种，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。

[0465] 由于计算机和网络不断变化的性质，对图28中绘出的计算机系统2800的描述旨在仅仅作为具体示例。具有比图28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理解实现各种实施例的其它方式和/或方法。

[0466] 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，虽然已使用特定系列的事务和步骤描述了本公开的实施例，然而，对本领域技术人员应当清楚的是，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或结合使用。

[0467] 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内。本公开的实施例可以只用硬件、或只用软件、或利用其组合来实现。本文描述的各种进程可以在同一处理器或以任何组合的不同处理器上实现。相应地，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、或其任意组合来实现。进程可以利用各种技术来通信，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。

[0468] 因而，说明书和附图应当在说明性而不是限制性的意义上考虑。然而，将清楚的是，在不背离权利要求中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进行限制。各种修改和等效物都在以下权利要求的范围之内。修改和变化包括所公开特征的任意相关组合。

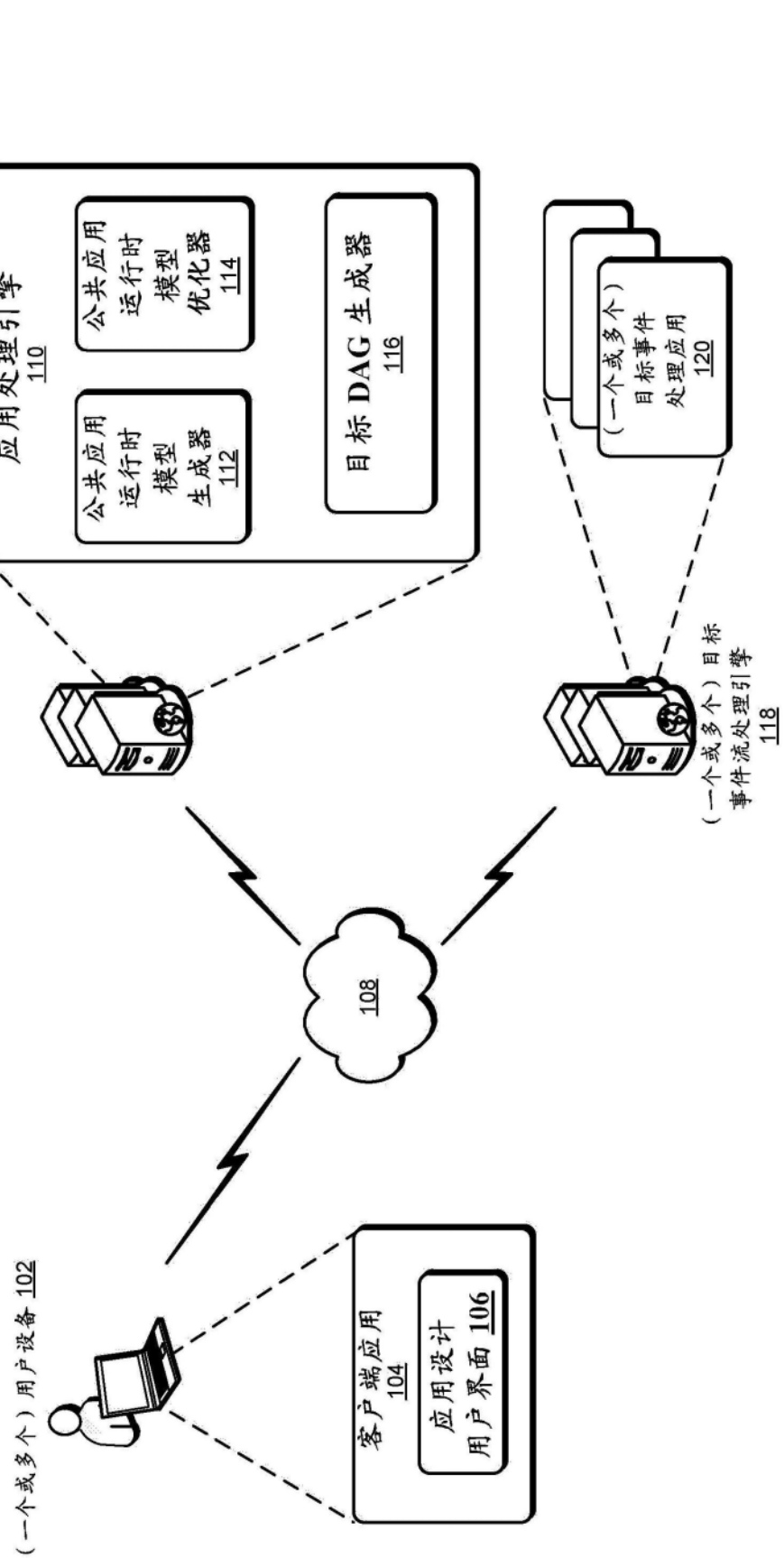

100
应用处理引擎
110
公共应用
运行时
模型
生成器
112
公共应用
运行时
模型
优化器
114
目标 DAG 生成器
116
(一个或多个)
目标事件
处理应用
120
(一个或多个)目标
事件流处理引擎
118
108
(一个或多个)用户设备 102
客户端应用
104
应用设计
用户界面 106

图1

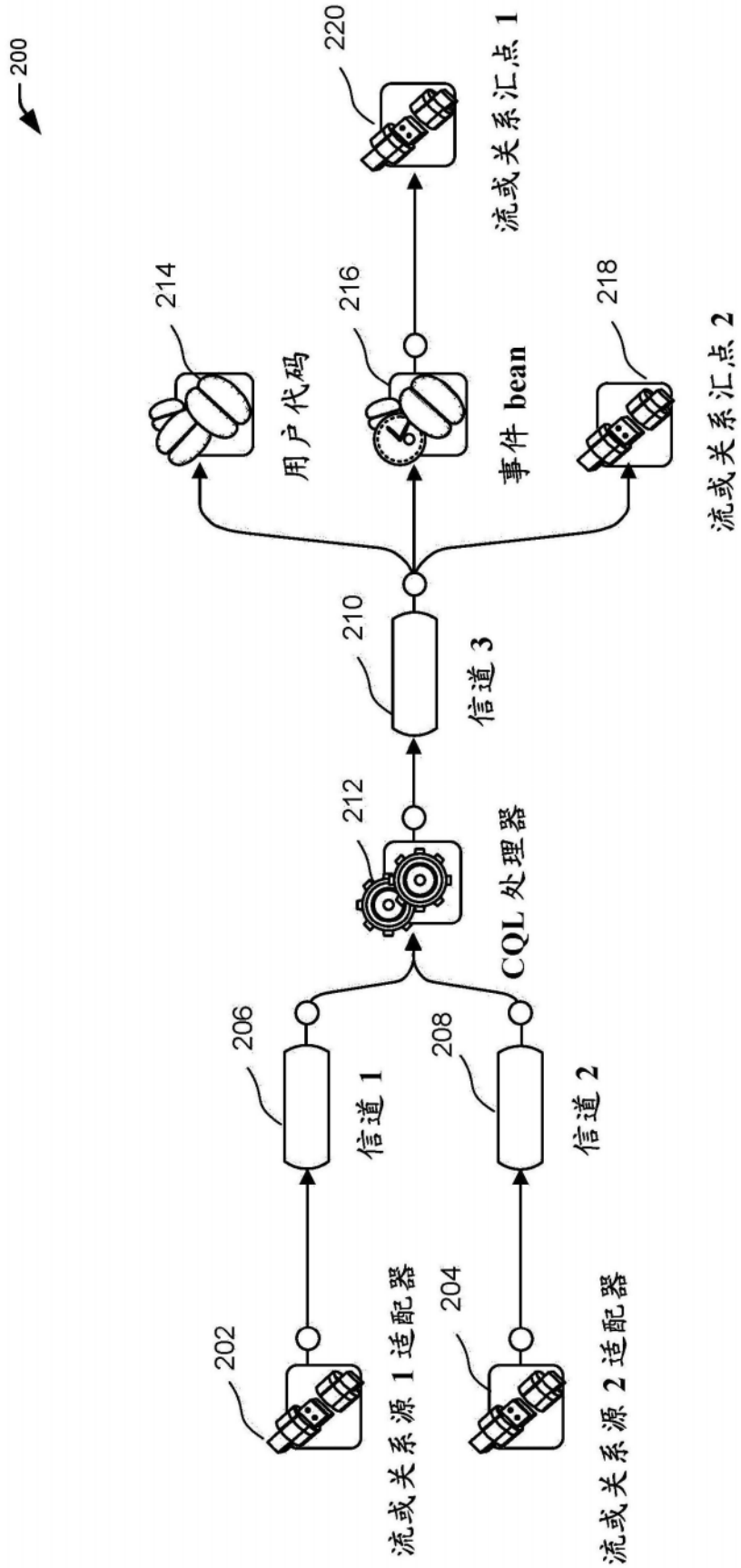

200
202
流或关系源 1 适配器
204
流或关系源 2 适配器
206
信道 1
208
信道 2
CQL 处理器
212
210
信道 3
214
用户代码
216
事件 bean
218
流或关系汇点 2
220
流或关系汇点 1

图2

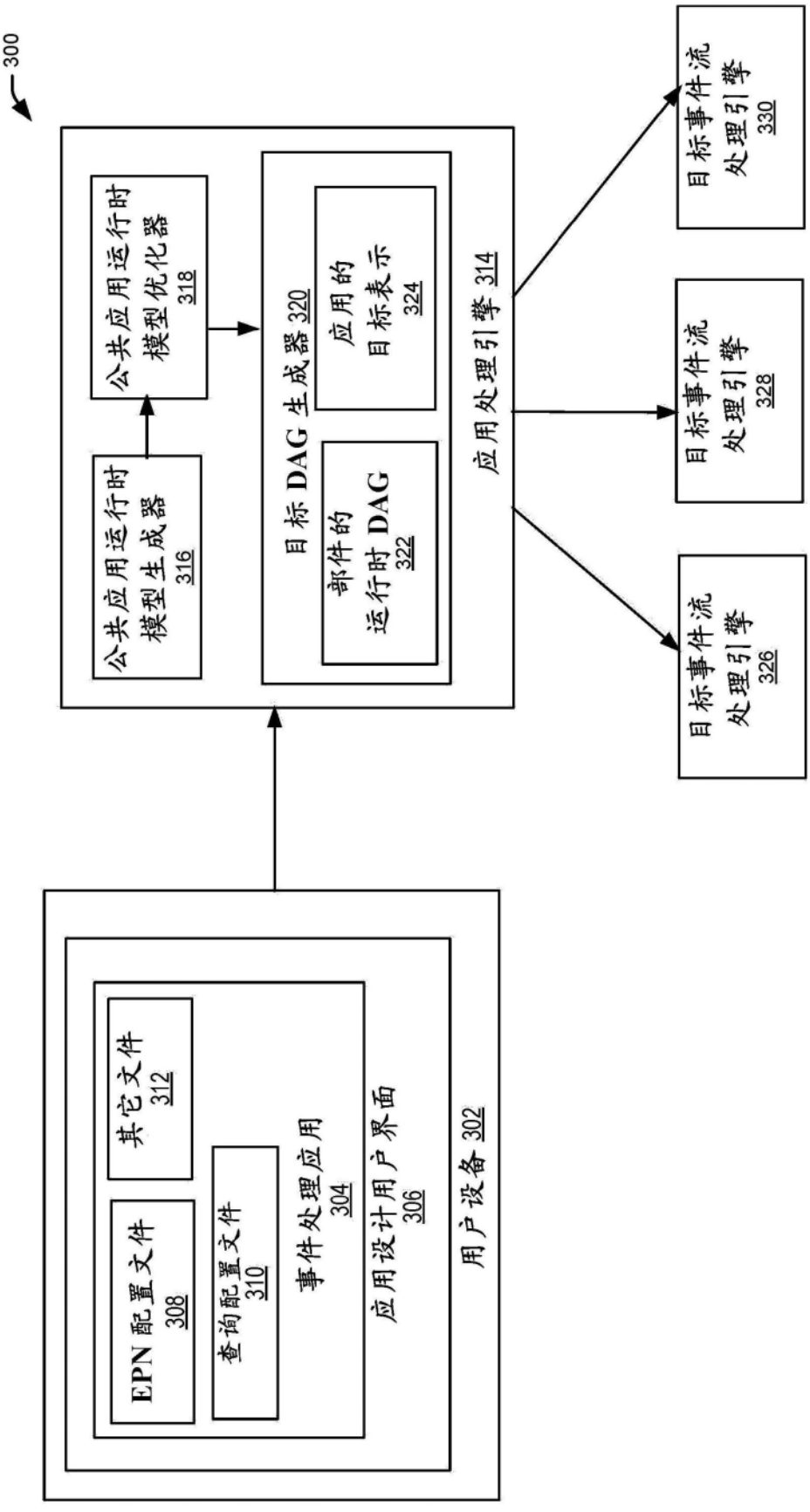

300
用户设备 302
应用设计用户界面 306
事件处理应用 304
EPN 配置文件 308
其它文件 312
查询配置文件 310
应用处理引擎 314
公共应用运行时 模型生成器 316
公共应用运行时 模型优化器 318
目标 DAG 生成器 320
部件的 运行时 DAG 322
应用的 目标表示 324
目标事件流 处理引擎 326
目标事件流 处理引擎 328
目标事件流 处理引擎 330

图3

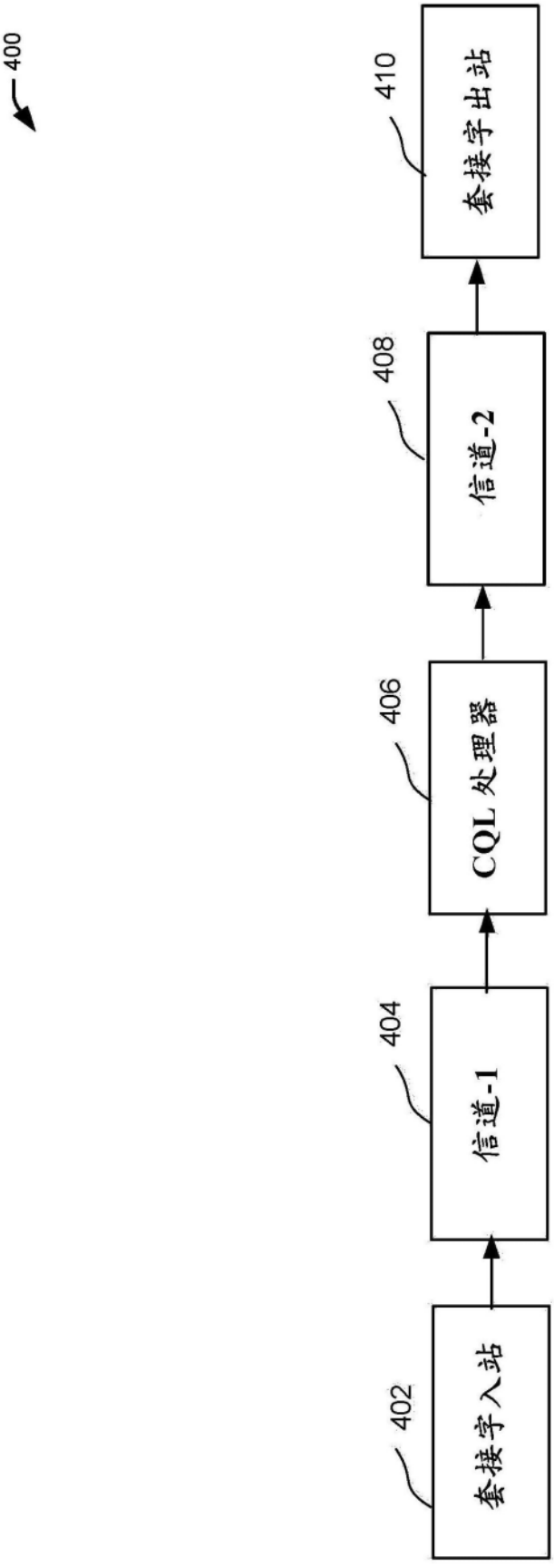


图4

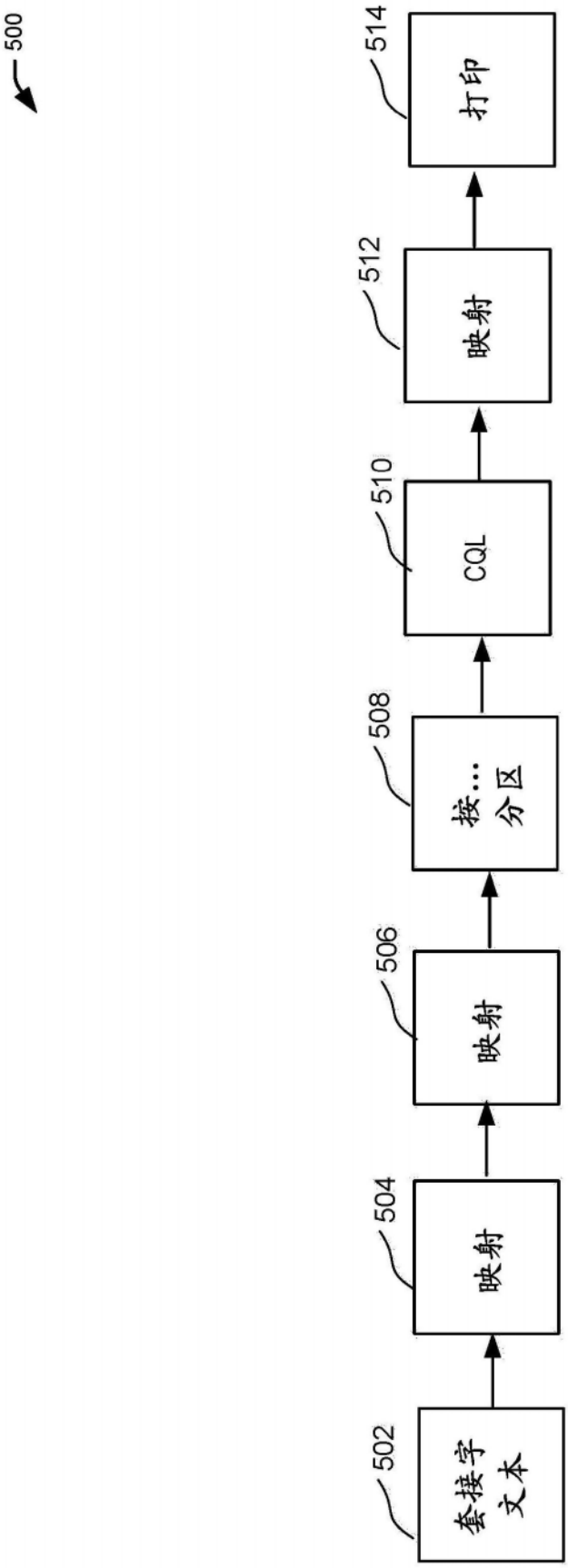


图5

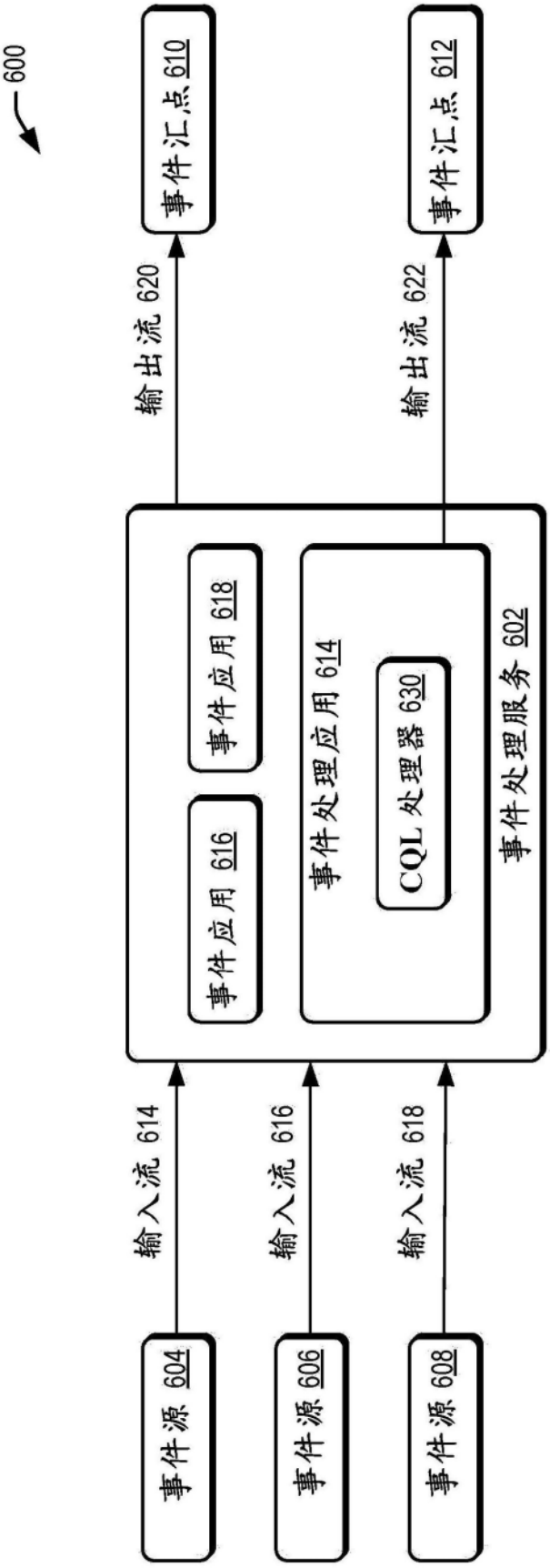

600
事件汇点 610
事件汇点 612
输出流 620
输出流 622
事件应用 618
事件应用 616
事件处理应用 614
CQL 处理器 630
事件处理服务 602
输入流 614
输入流 616
输入流 618
事件源 604
事件源 606
事件源 608

图6

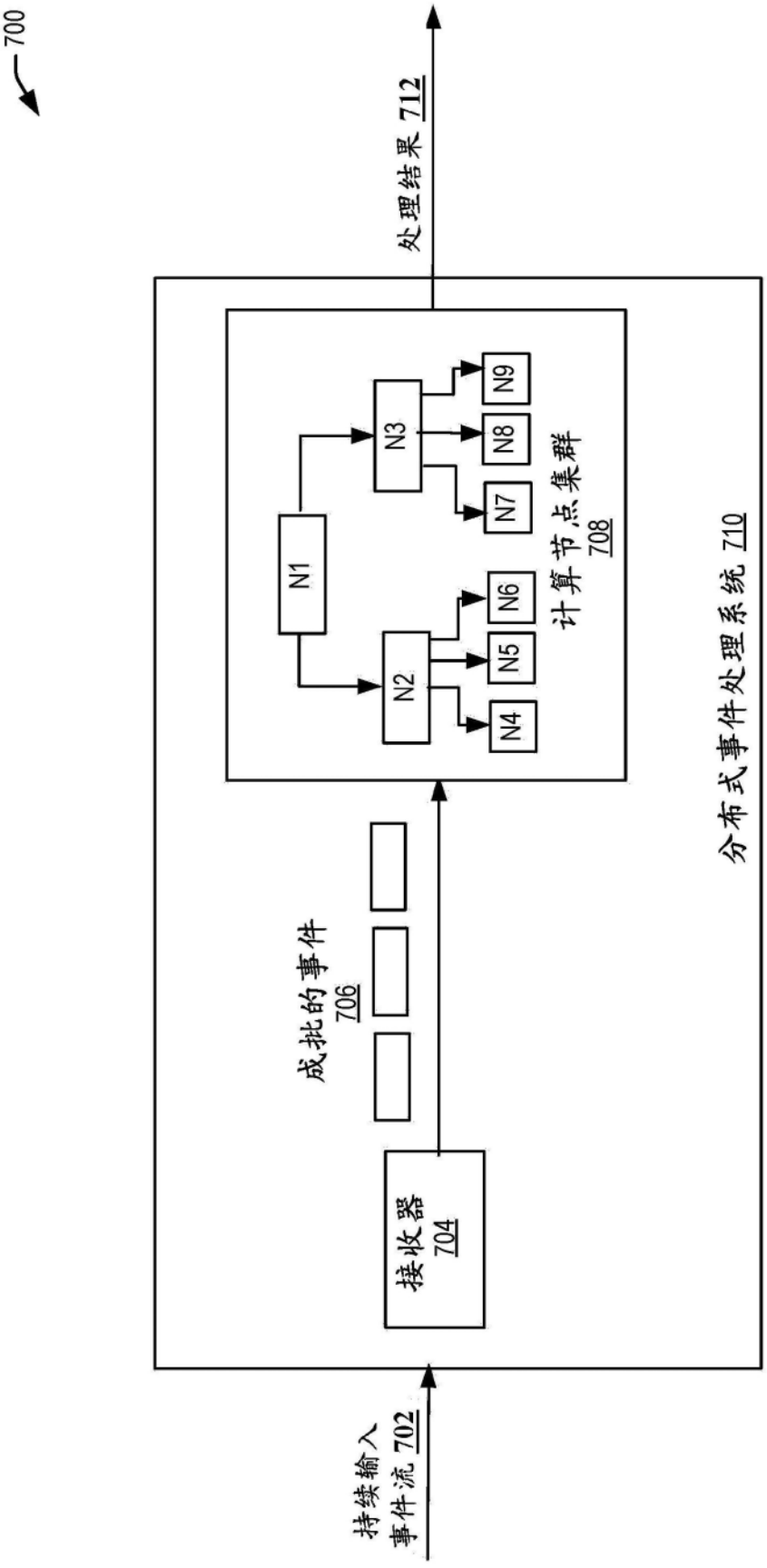

700
处理结果 712
分布式事件处理系统 710
计算节点集群 708
N1
N2
N3
N4
N5
N6
N7
N8
N9
成批的事件 706
接收器 704
持续输入事件流 702

图7

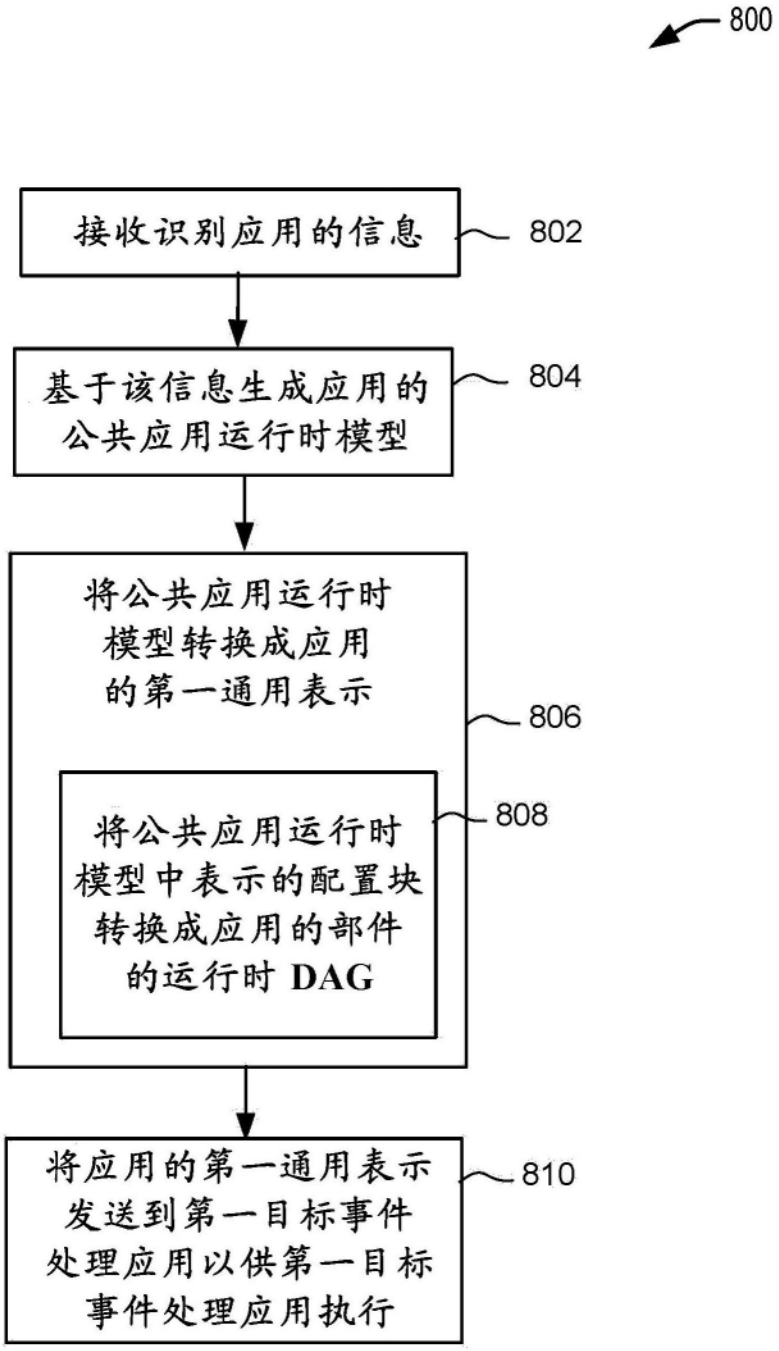

800
接收识别应用的信息
802
基于该信息生成应用的公共应用运行时模型
804
将公共应用运行时模型转换成应用的第一通用表示
806
将公共应用运行时模型中表示的配置块转换成应用的部件的运行时 DAG
808
将应用的第一通用表示发送到第一目标事件处理应用以供第一目标事件处理应用执行
810

图8

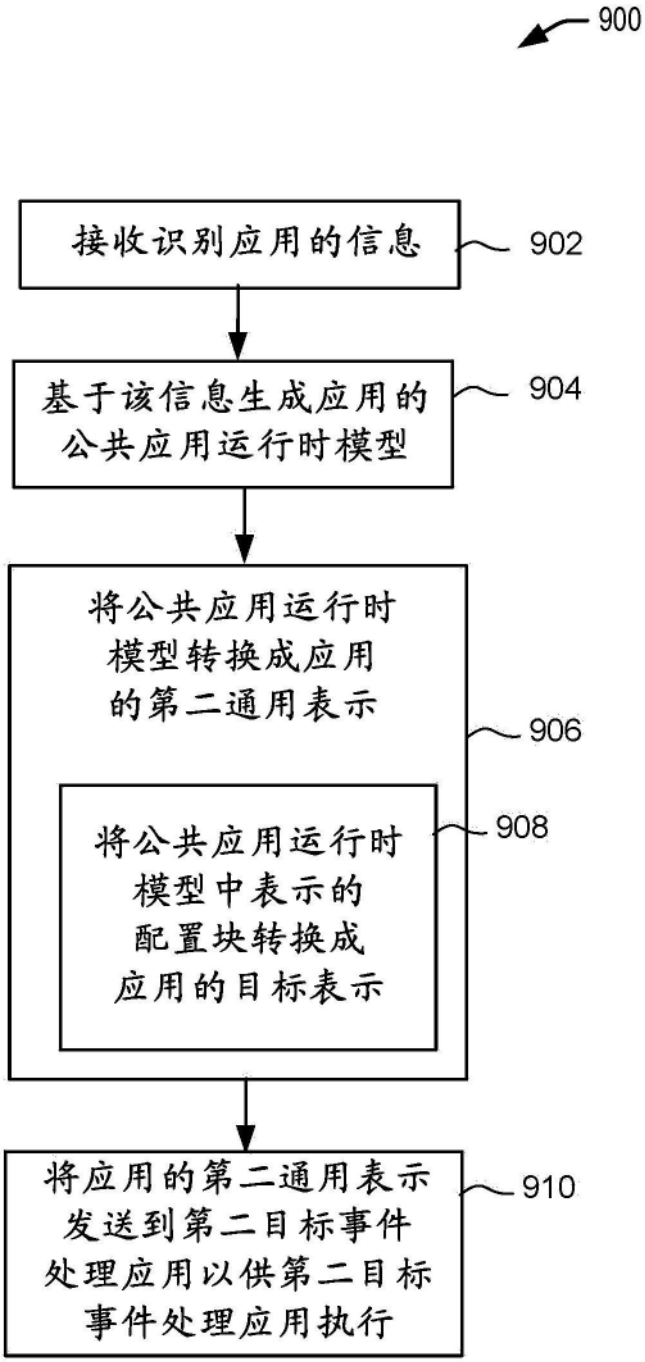

900
接收识别应用的信息
902
基于该信息生成应用的公共应用运行时模型
904
将公共应用运行时模型转换成应用的第二通用表示
906
将公共应用运行时模型中表示的配置块转换成应用的目标表示
908
将应用的第二通用表示发送到第二目标事件处理应用以供第二目标事件处理应用执行
910

图9

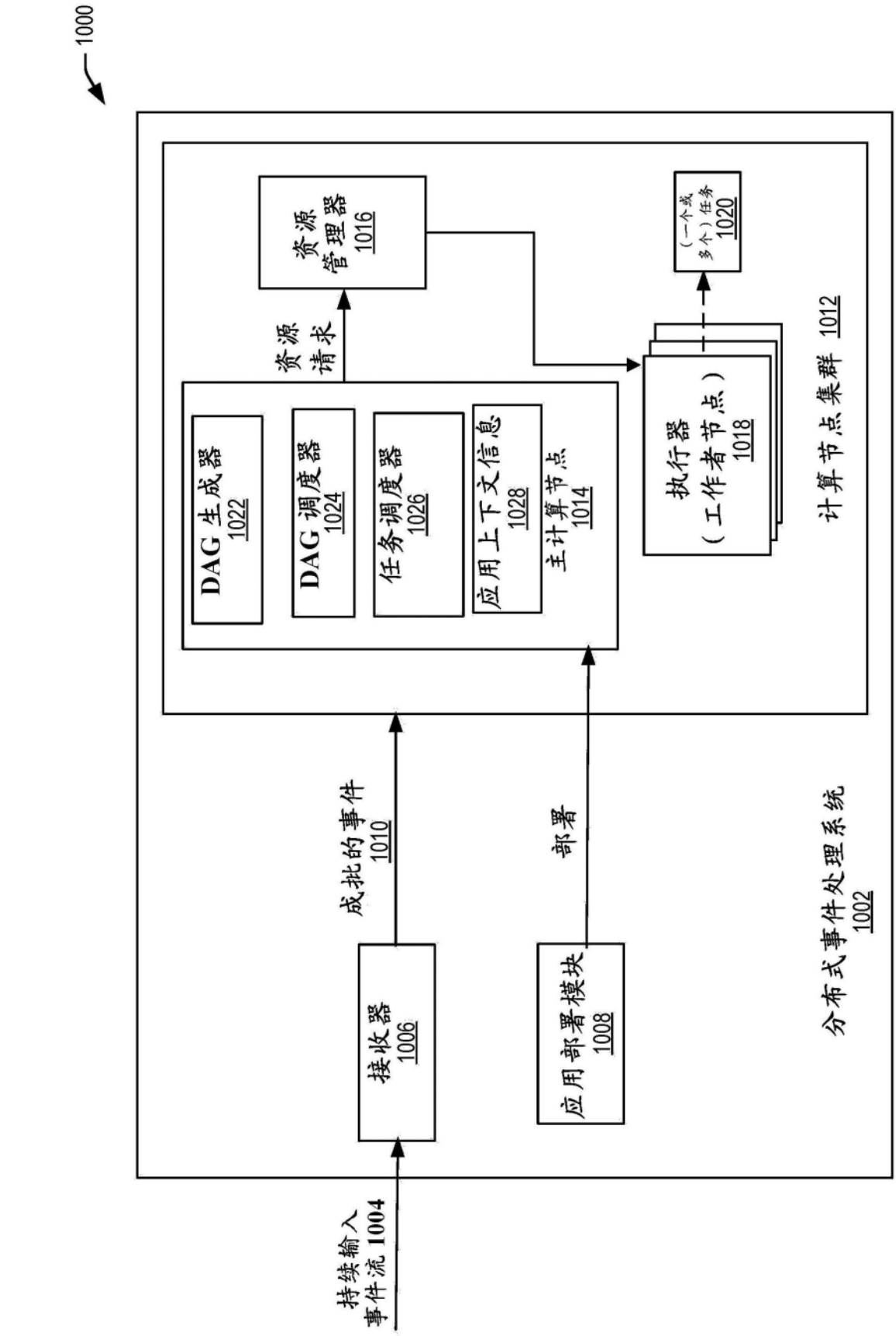

1000
分布式事件处理系统 1002
计算节点集群 1012
持续输入事件流 1004
接收器 1006
应用部署模块 1008
成批的事件 1010
部署
主计算节点 1014
DAG 生成器 1022
DAG 调度器 1024
任务调度器 1026
应用上下文信息 1028
资源请求
资源管理器 1016
执行器（工作者节点）1018
（一个或多个）任务 1020

图10

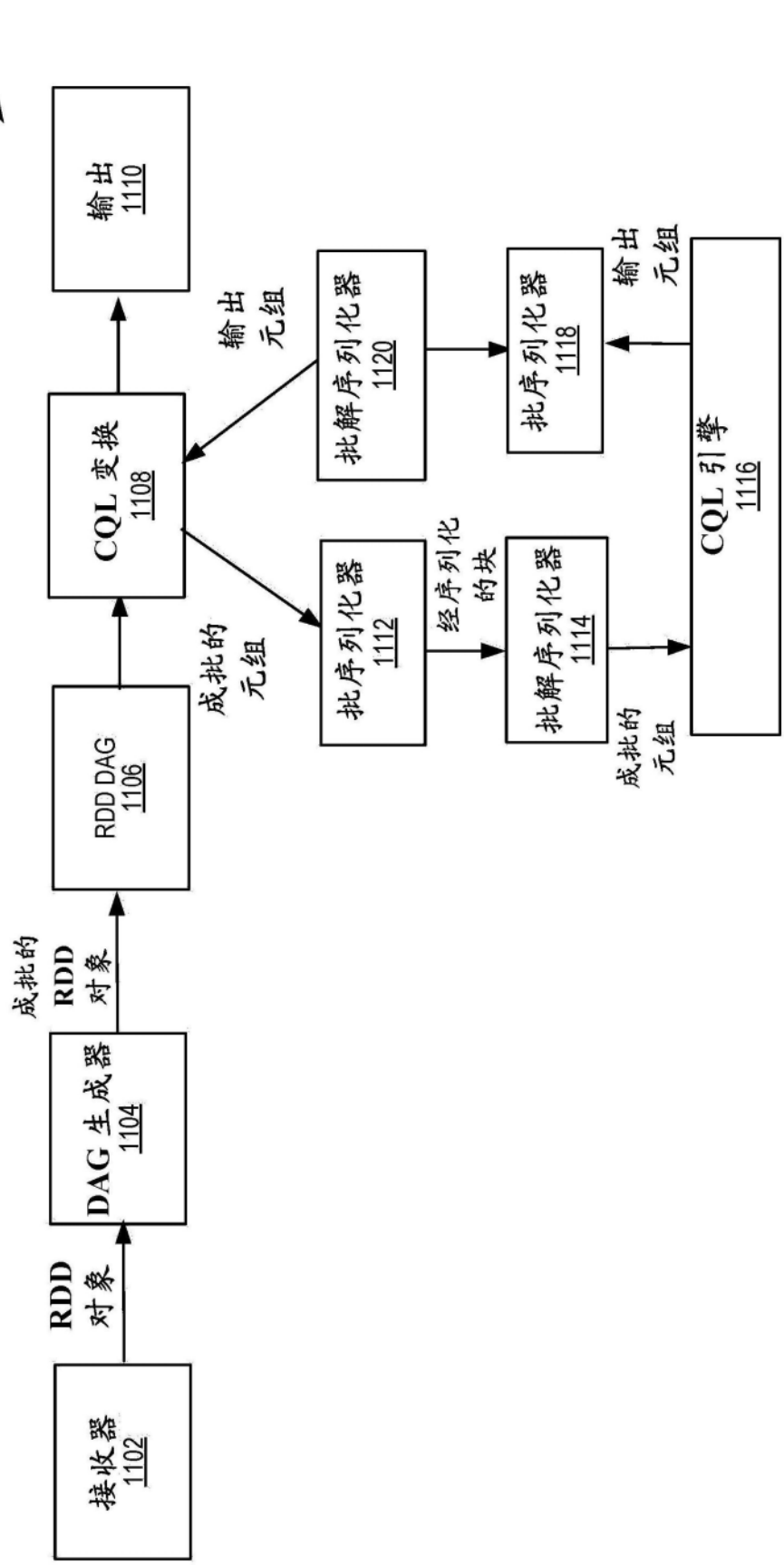

1100
接收器 1102
RDD 对象
DAG 生成器 1104
成批的 RDD 对象
RDD DAG 1106
CQL 变换 1108
输出 1110
成批的元组
批序列化器 1112
经序列化的块
批解序列化器 1114
成批的元组
CQL 引擎 1116
输出元组
批序列化器 1118
输出元组
批解序列化器 1120

图11

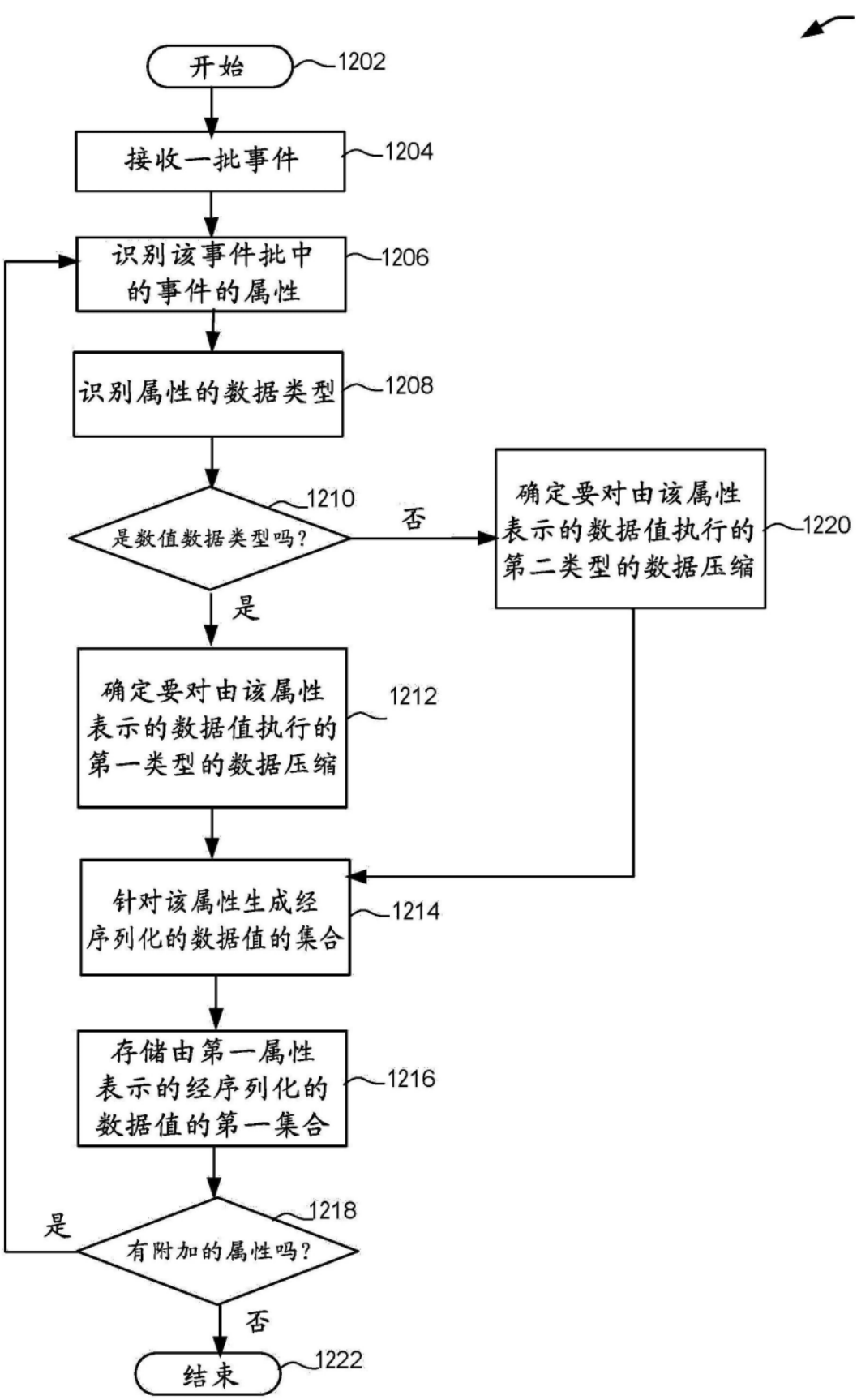

1200
开始
1202
接收一批事件
1204
识别该事件批中的事件的属性
1206
识别属性的数据类型
1208
是数值数据类型吗?
1210
否
确定要对由该属性表示的数据值执行的第二类型的数据压缩
1220
是
确定要对由该属性表示的数据值执行的第一类型的数据压缩
1212
针对该属性生成经序列化的数据值的集合
1214
存储由第一属性表示的经序列化的数据值的第一集合
1216
有附加的属性吗?
1218
是
否
结束
1222

图12

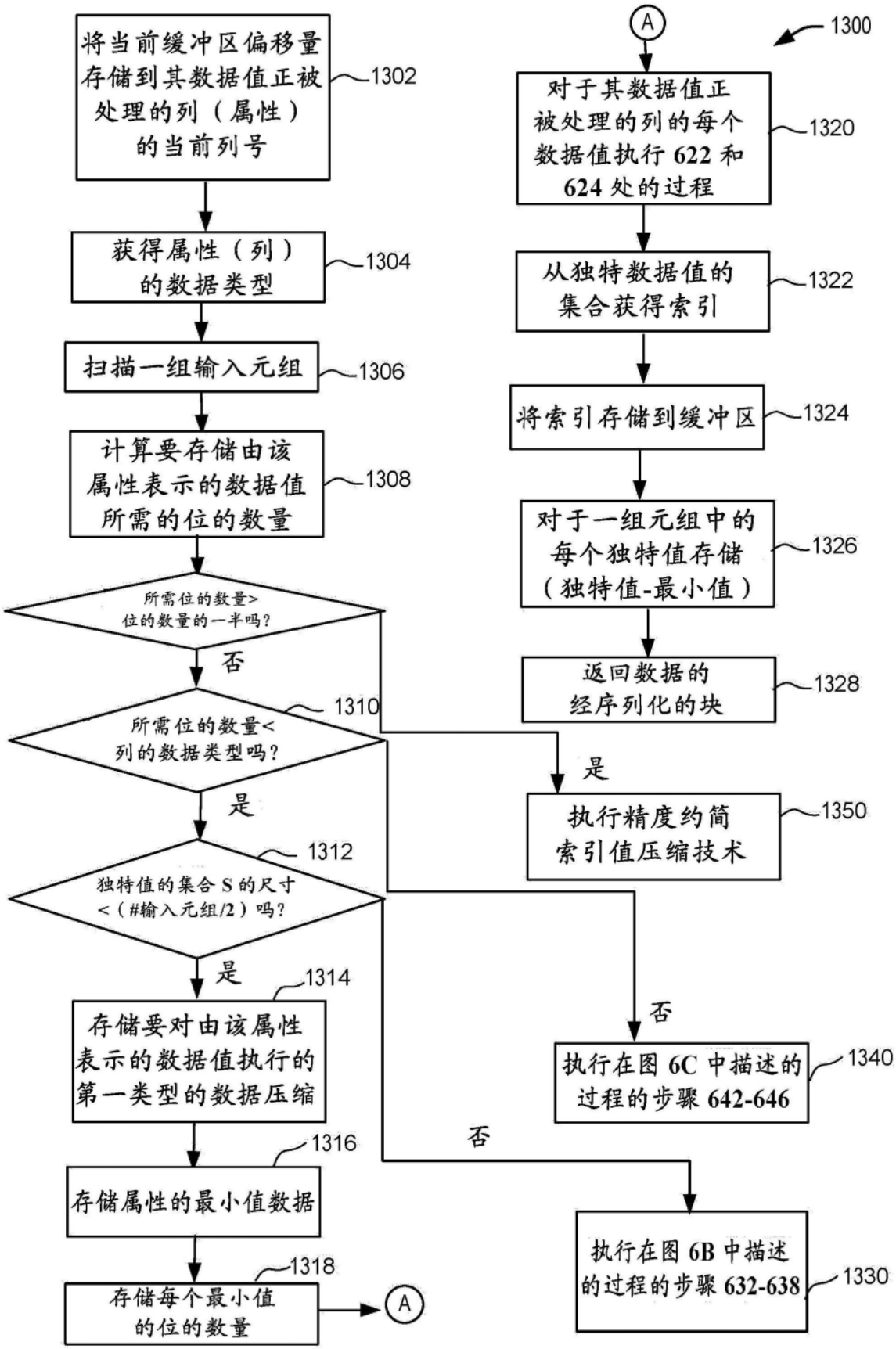

1300
将当前缓冲区偏移量存储到其数据值正被处理的列（属性）的当前列号
1302
获得属性（列）的数据类型
1304
扫描一组输入元组
1306
计算要存储由该属性表示的数据值所需的位的数量
1308
所需位的数量>位的数量的一半吗?
否
所需位的数量<列的数据类型吗?
1310
是
独特值的集合 S 的尺寸<（#输入元组/2）吗?
1312
是
存储要对由该属性表示的数据值执行的第一类型的数据压缩
1314
存储属性的最小值数据
1316
存储每个最小值的位的数量
1318
A
A
对于其数据值正被处理的列的每个数据值执行 622 和 624 处的过程
1320
从独特数据值的集合获得索引
1322
将索引存储到缓冲区
1324
对于一组元组中的每个独特值存储（独特值-最小值）
1326
返回数据的经序列化的块
1328
是
执行精度约简索引值压缩技术
1350
否
执行在图 6C 中描述的过程的步骤 642-646
1340
否
执行在图 6B 中描述的过程的步骤 632-638
1330

图13A

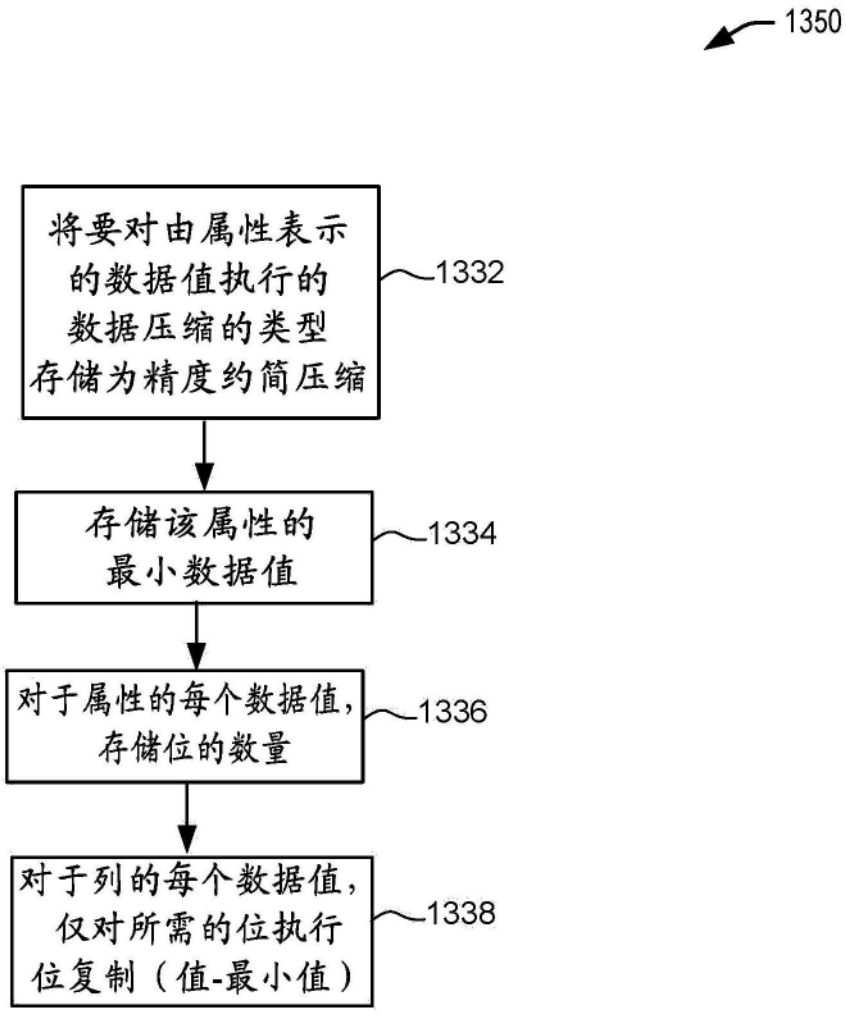

1350
将要对由属性表示的数据值执行的数据压缩的类型存储为精度约简压缩
1332
存储该属性的最小数据值
1334
对于属性的每个数据值，存储位的数量
1336
对于列的每个数据值，仅对所需的位执行位复制（值-最小值）
1338

图13B

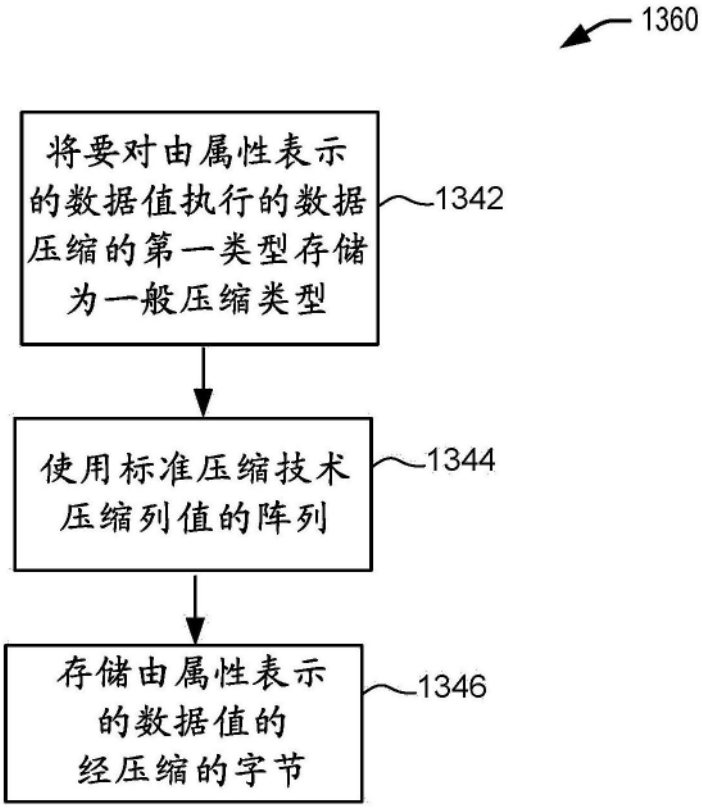

1360
将要对由属性表示的数据值执行的数据压缩的第一类型存储为一般压缩类型
1342
使用标准压缩技术压缩列值的阵列
1344
存储由属性表示的数据值的经压缩的字节
1346

图13C

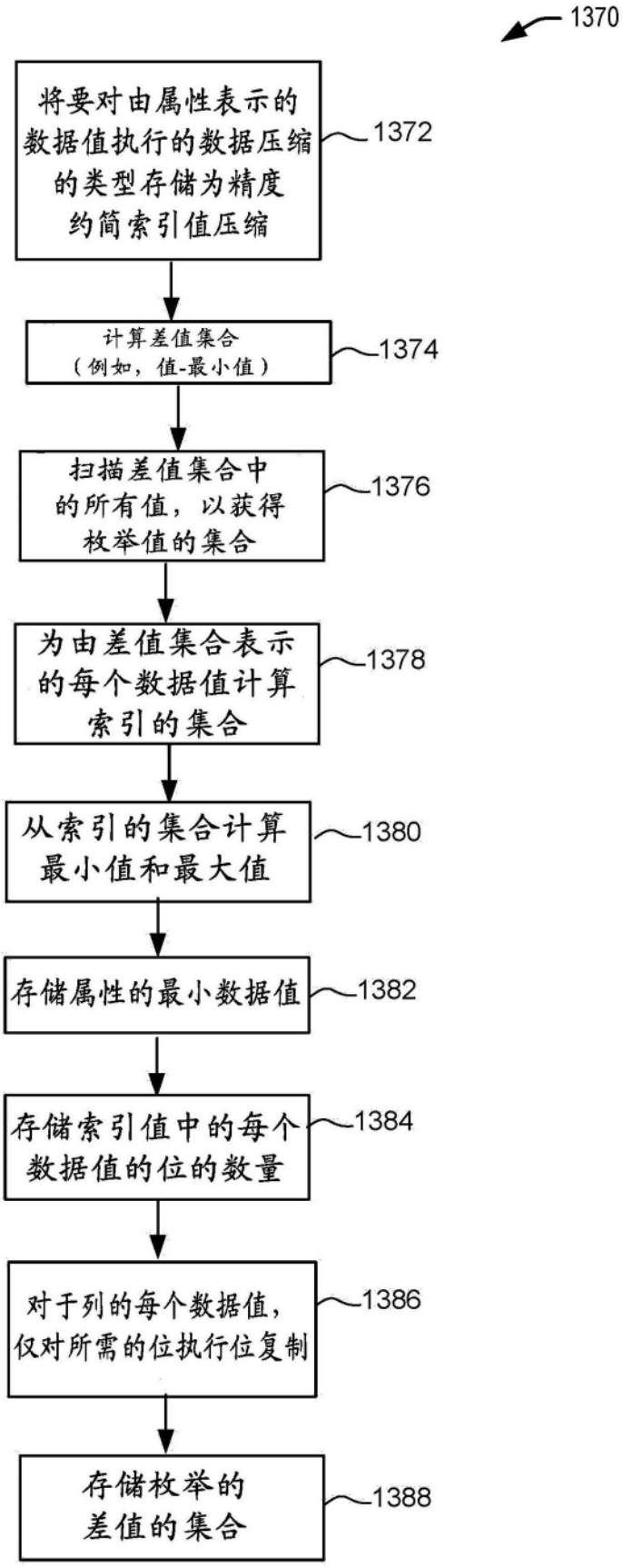

1370
将要对由属性表示的数据值执行的数据压缩的类型存储为精度约简索引值压缩
1372
计算差值集合（例如，值-最小值）
1374
扫描差值集合中的所有值，以获得枚举值的集合
1376
为由差值集合表示的每个数据值计算索引的集合
1378
从索引的集合计算最小值和最大值
1380
存储属性的最小数据值
1382
存储索引值中的每个数据值的位的数量
1384
对于列的每个数据值，仅对所需的位执行位复制
1386
存储枚举的差值的集合
1388

图13D

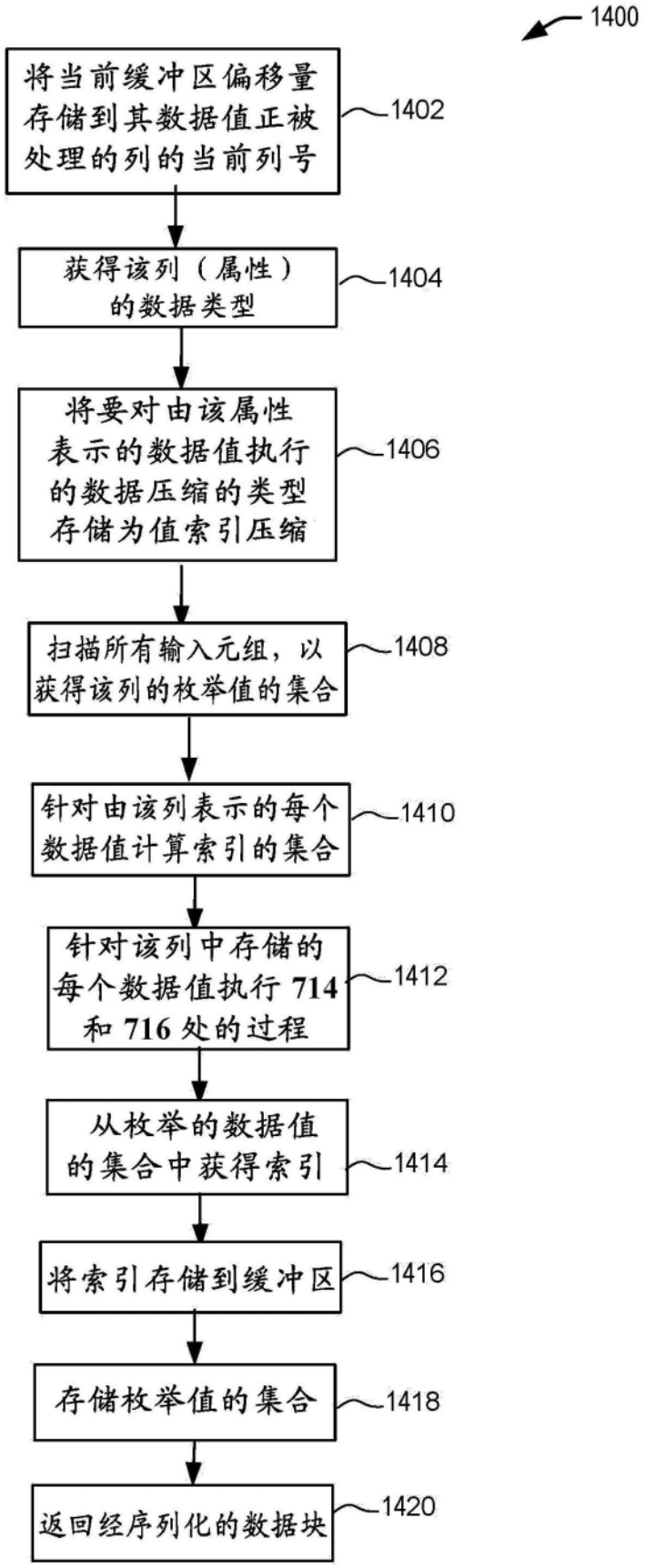

1400
将当前缓冲区偏移量存储到其数据值正被处理的列的当前列号
1402
获得该列（属性）的数据类型
1404
将要对由该属性表示的数据值执行的数据压缩的类型存储为值索引压缩
1406
扫描所有输入元组，以获得该列的枚举值的集合
1408
针对由该列表示的每个数据值计算索引的集合
1410
针对该列中存储的每个数据值执行 714 和 716 处的过程
1412
从枚举的数据值的集合中获得索引
1414
将索引存储到缓冲区
1416
存储枚举值的集合
1418
返回经序列化的数据块
1420

图14

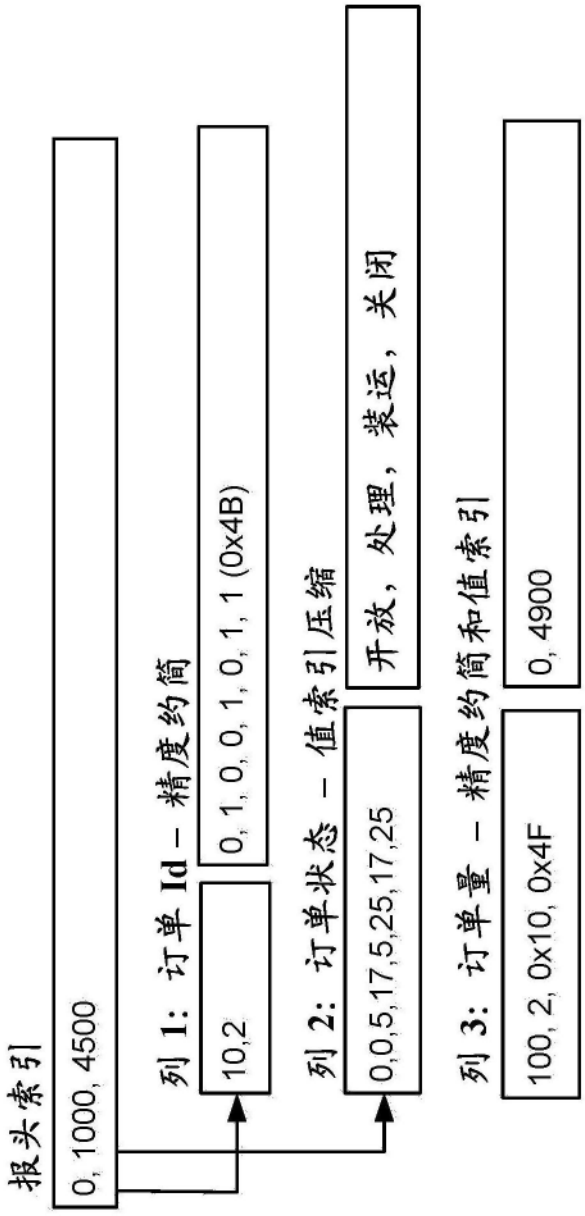

报头索引
0, 1000, 4500
列 1: 订单 Id – 精度约简
10,2
0, 1, 0, 0, 1, 0, 1, 1 (0x4B)
列 2: 订单状态 – 值索引压缩
0,0,5,17,5,25,17,25
开放，处理，装运，关闭
列 3: 订单量 – 精度约简和值索引
100, 2, 0x10, 0x4F
0, 4900

图15

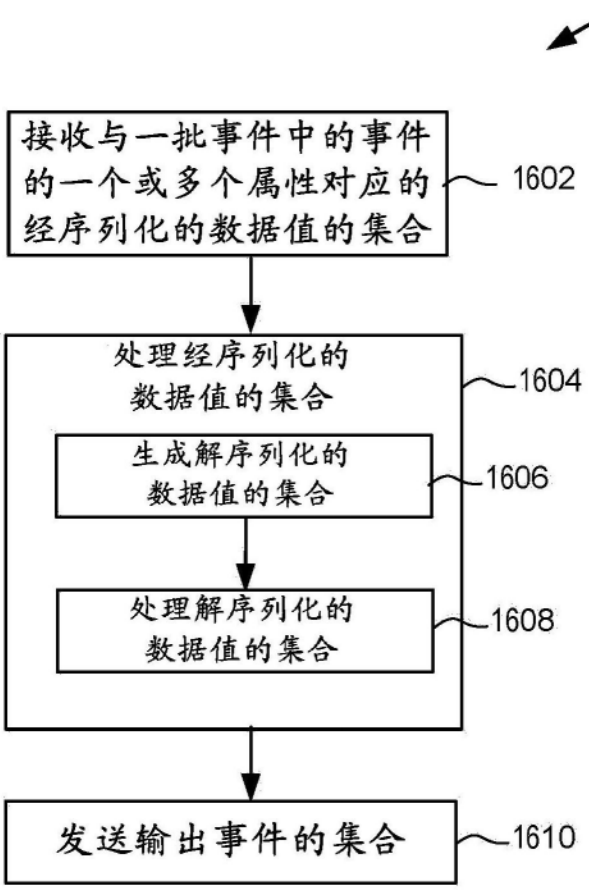

1600
接收与一批事件中的事件的一个或多个属性对应的经序列化的数据值的集合
1602
处理经序列化的数据值的集合
1604
生成解序列化的数据值的集合
1606
处理解序列化的数据值的集合
1608
发送输出事件的集合
1610

图16

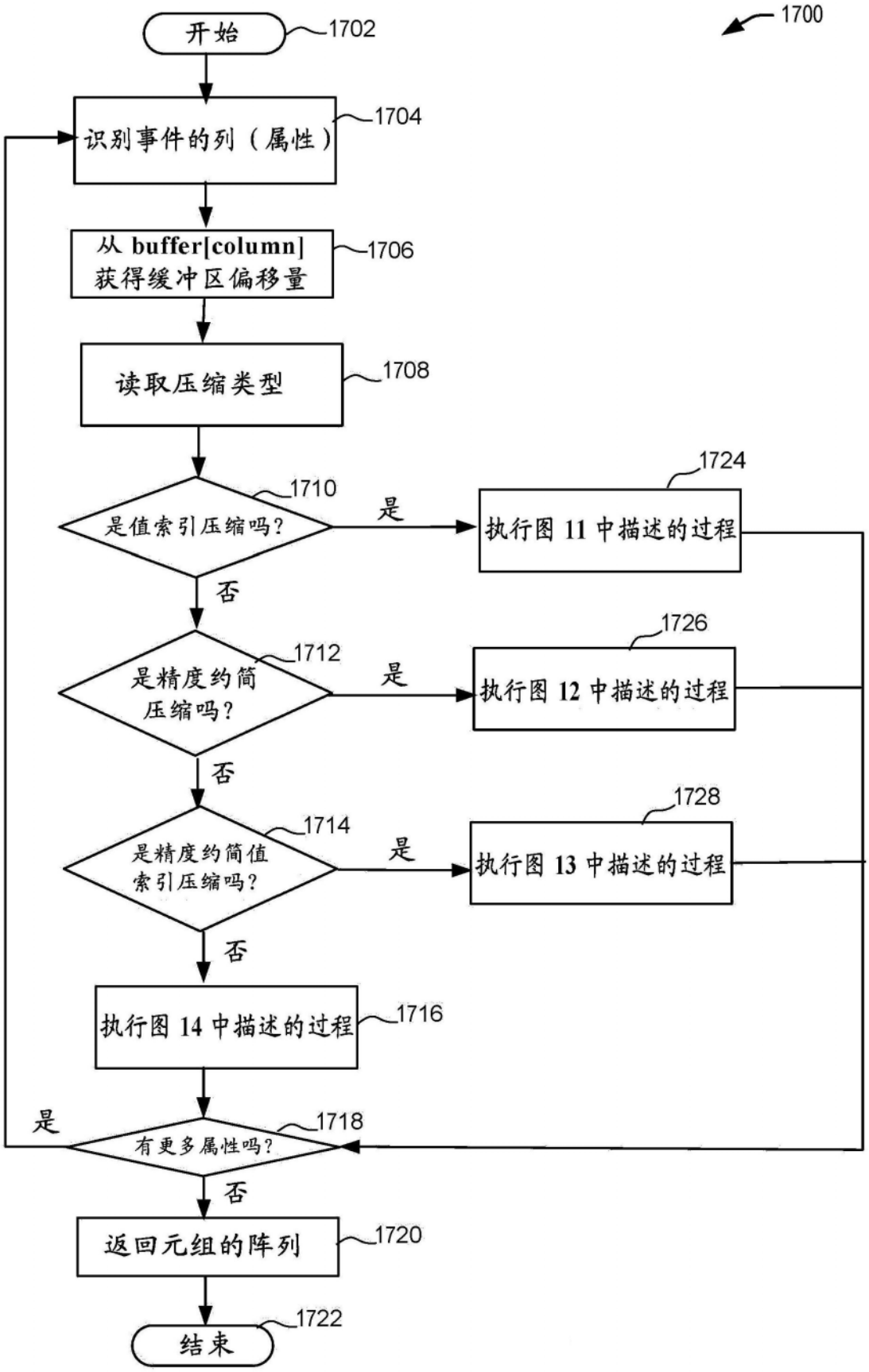

1700
开始
1702
识别事件的列（属性）
1704
从 buffer[column] 获得缓冲区偏移量
1706
读取压缩类型
1708
是值索引压缩吗？
1710
是
执行图 11 中描述的过程
1724
否
是精度约简压缩吗？
1712
是
执行图 12 中描述的过程
1726
否
是精度约简值索引压缩吗？
1714
是
执行图 13 中描述的过程
1728
否
执行图 14 中描述的过程
1716
是
有更多属性吗？
1718
否
返回元组的阵列
1720
结束
1722

图17

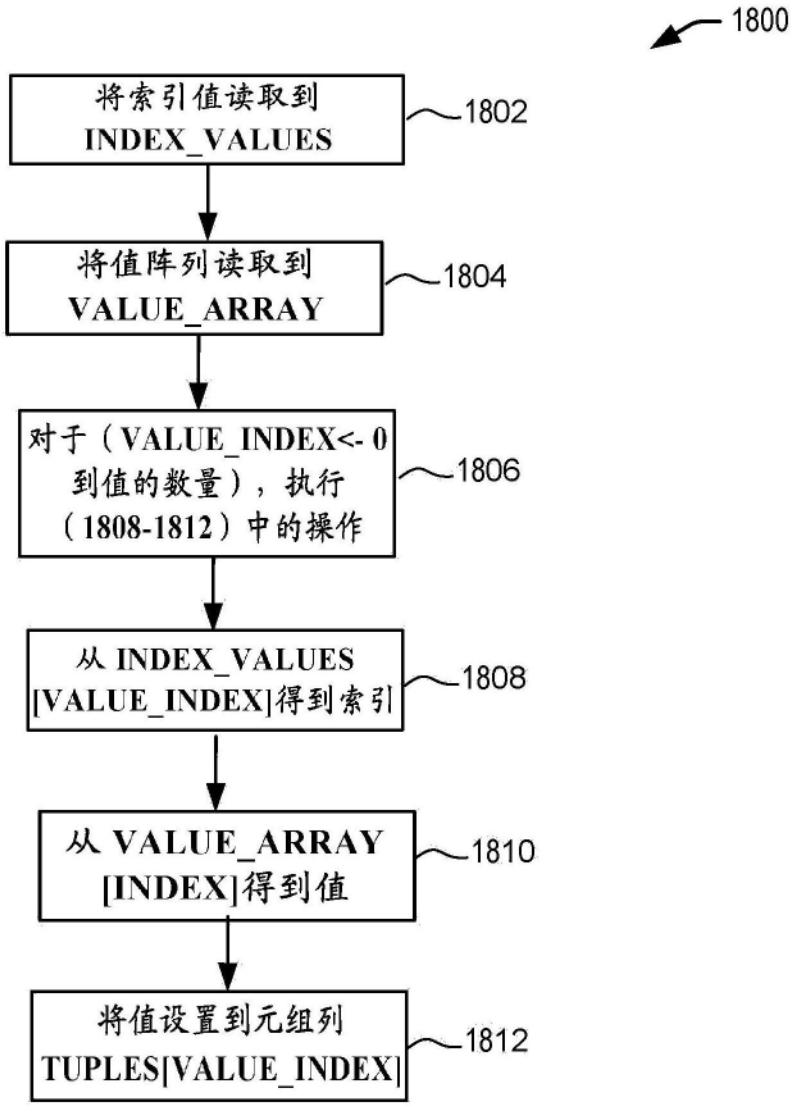

1800
将索引值读取到 INDEX_VALUES
1802
将值阵列读取到 VALUE_ARRAY
1804
对于（VALUE_INDEX<- 0到值的数量），执行（1808-1812）中的操作
1806
从 INDEX_VALUES [VALUE_INDEX]得到索引
1808
从 VALUE_ARRAY [INDEX]得到值
1810
将值设置到元组列 TUPLES[VALUE_INDEX]
1812

图18

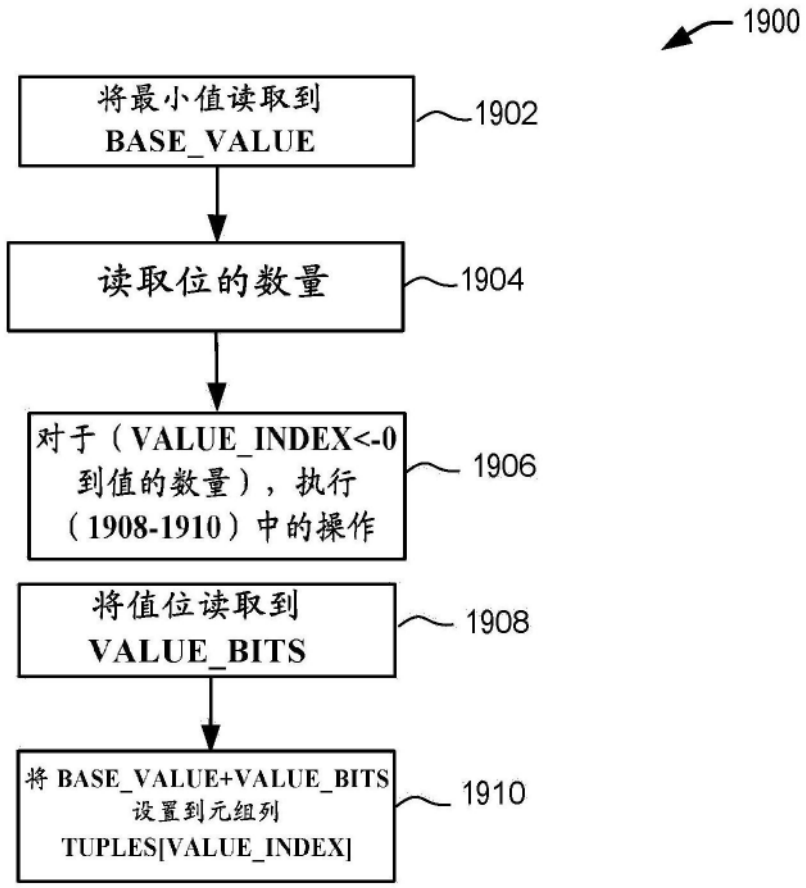

1900
将最小值读取到
BASE_VALUE
1902
读取位的数量
1904
对于（VALUE_INDEX<-0
到值的数量），执行
（1908-1910）中的操作
1906
将值位读取到
VALUE_BITS
1908
将 BASE_VALUE+VALUE_BITS
设置到元组列
TUPLES[VALUE_INDEX]
1910

图19

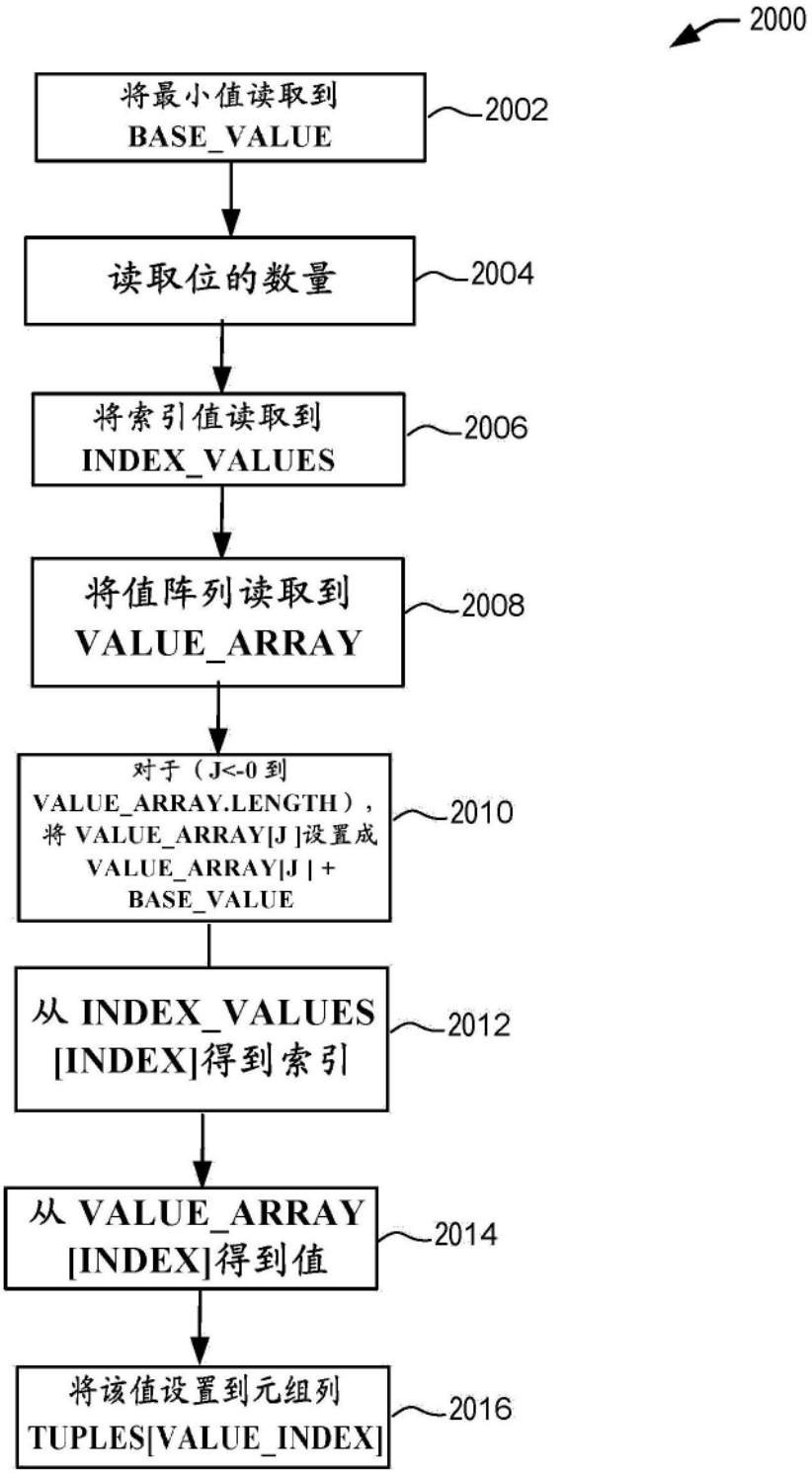

2000
将最小值读取到 BASE_VALUE
2002
读取位的数量
2004
将索引值读取到 INDEX_VALUES
2006
将值阵列读取到 VALUE_ARRAY
2008
对于（J<-0 到 VALUE_ARRAY.LENGTH），将 VALUE_ARRAY[J]设置成 VALUE_ARRAY[J] + BASE_VALUE
2010
从 INDEX_VALUES [INDEX]得到索引
2012
从 VALUE_ARRAY [INDEX]得到值
2014
将该值设置到元组列 TUPLES[VALUE_INDEX]
2016

图20

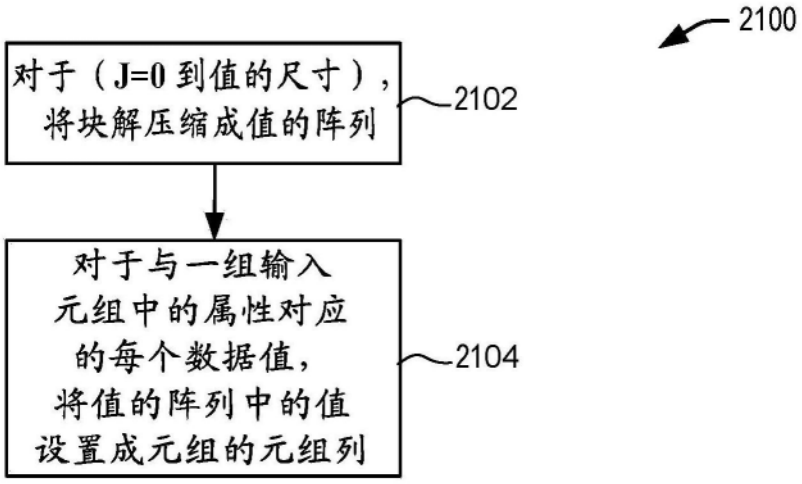

2100
对于（J=0 到值的尺寸），将块解压缩成值的阵列
2102
对于与一组输入元组中的属性对应的每个数据值，将值的阵列中的值设置成元组的元组列
2104

图21

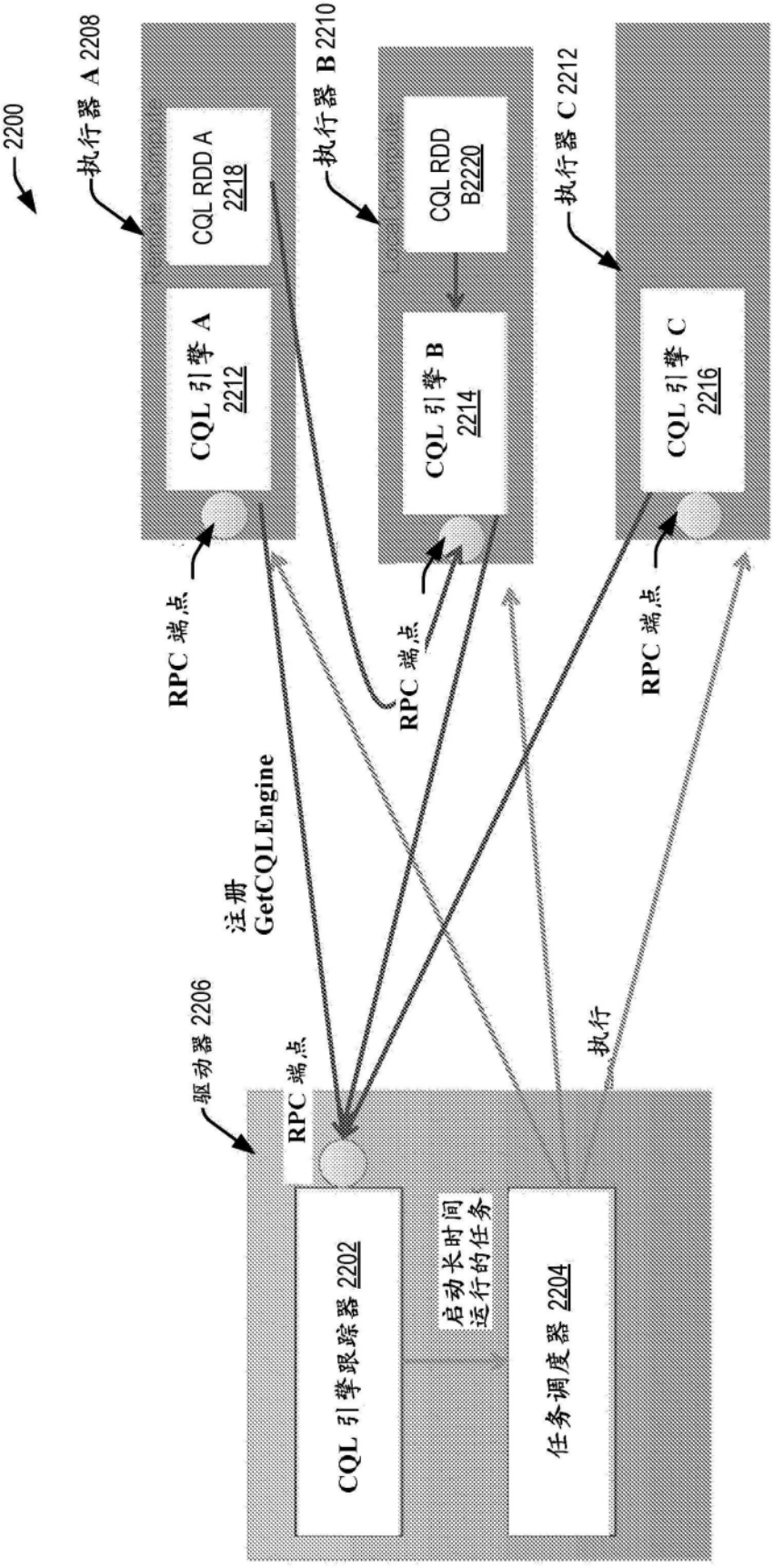

2200
执行器 A 2208
执行器 B 2210
执行器 C 2212
CQL RDD A 2218
CQL RDD B2220
CQL 引擎 A 2212
CQL 引擎 B 2214
CQL 引擎 C 2216
RPC 端点
RPC 端点
RPC 端点
注册 GetCQLEngine
驱动器 2206
RPC 端点
执行
CQL 引擎跟踪器 2202
启动长时间运行的任务
任务调度器 2204

图22

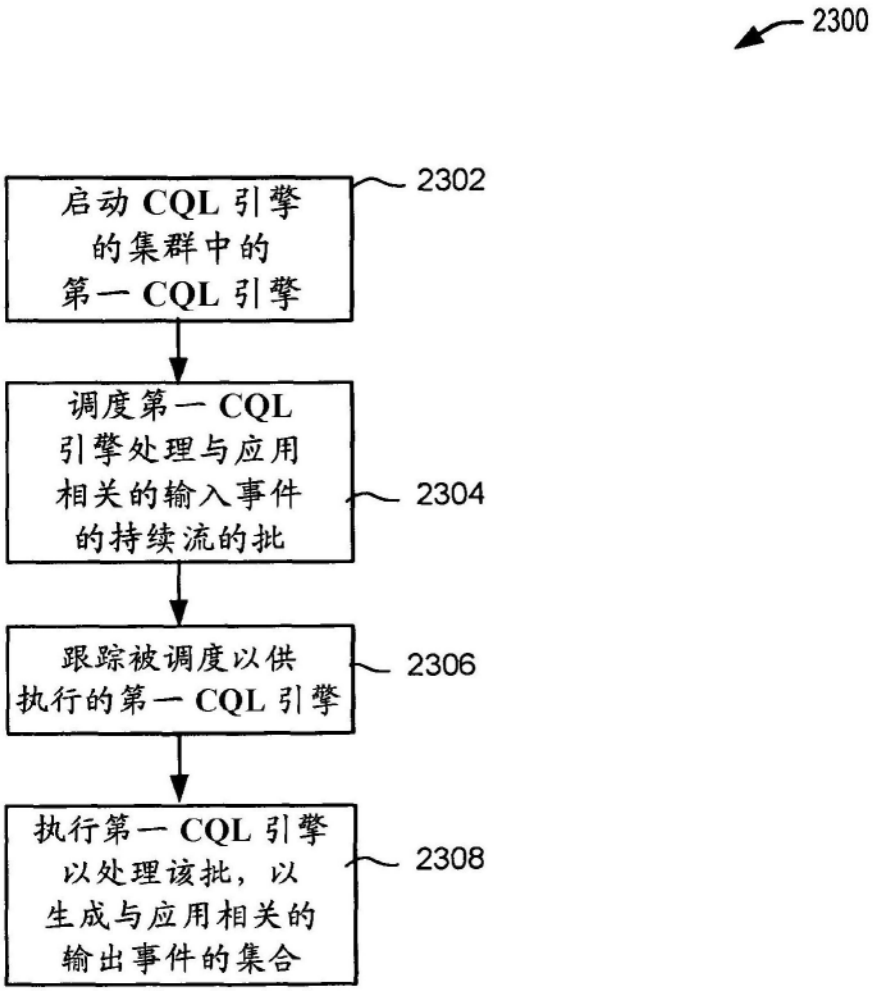

2300
启动 CQL 引擎的集群中的第一 CQL 引擎
2302
调度第一 CQL 引擎处理与应用相关的输入事件的持续流的批
2304
跟踪被调度以供执行的第一 CQL 引擎
2306
执行第一 CQL 引擎以处理该批，以生成与应用相关的输出事件的集合
2308

图23

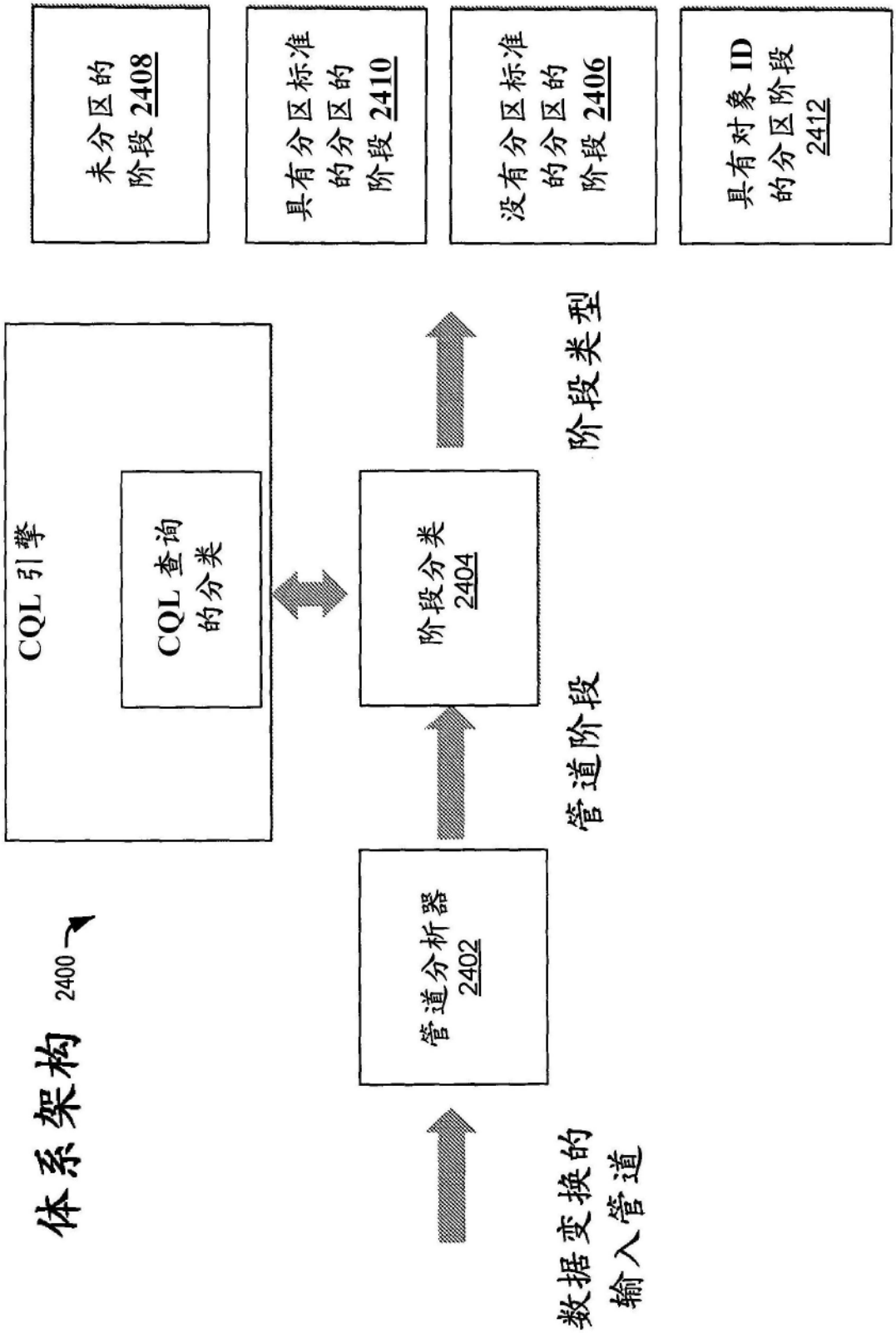

体系架构 2400
数据变换的输入管道
管道分析器 2402
管道阶段
阶段分类 2404
阶段类型
CQL 引擎
CQL 查询的分类
未分区的阶段 2408
具有分区标准的分区的阶段 2410
没有分区标准的分区的阶段 2406
具有对象 ID 的分区阶段 2412

图24

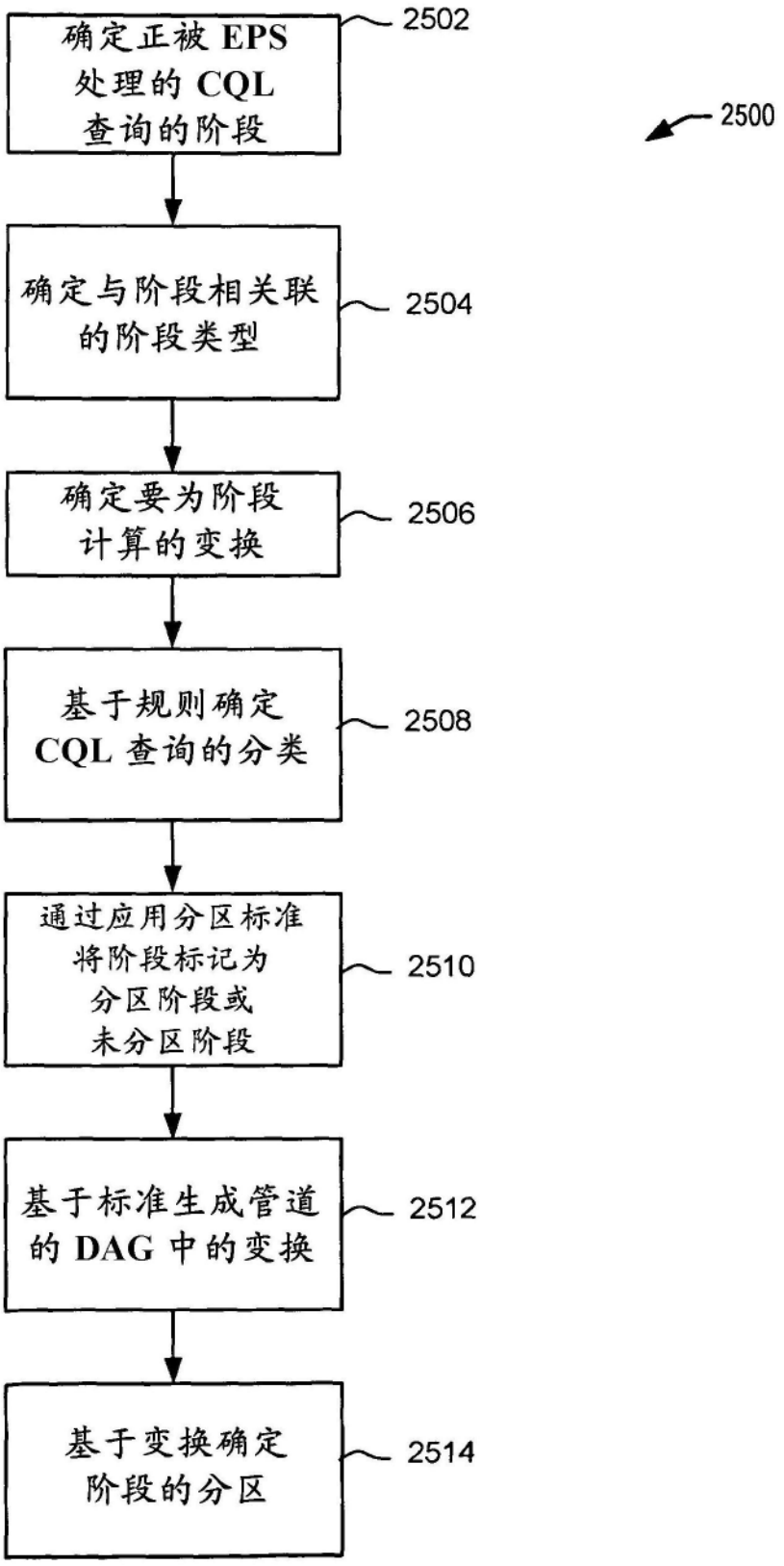

确定正被 EPS 处理的 CQL 查询的阶段
2502
2500
确定与阶段相关联的阶段类型
2504
确定要为阶段计算的变换
2506
基于规则确定 CQL 查询的分类
2508
通过应用分区标准将阶段标记为分区阶段或未分区阶段
2510
基于标准生成管道的 DAG 中的变换
2512
基于变换确定阶段的分区
2514

图25

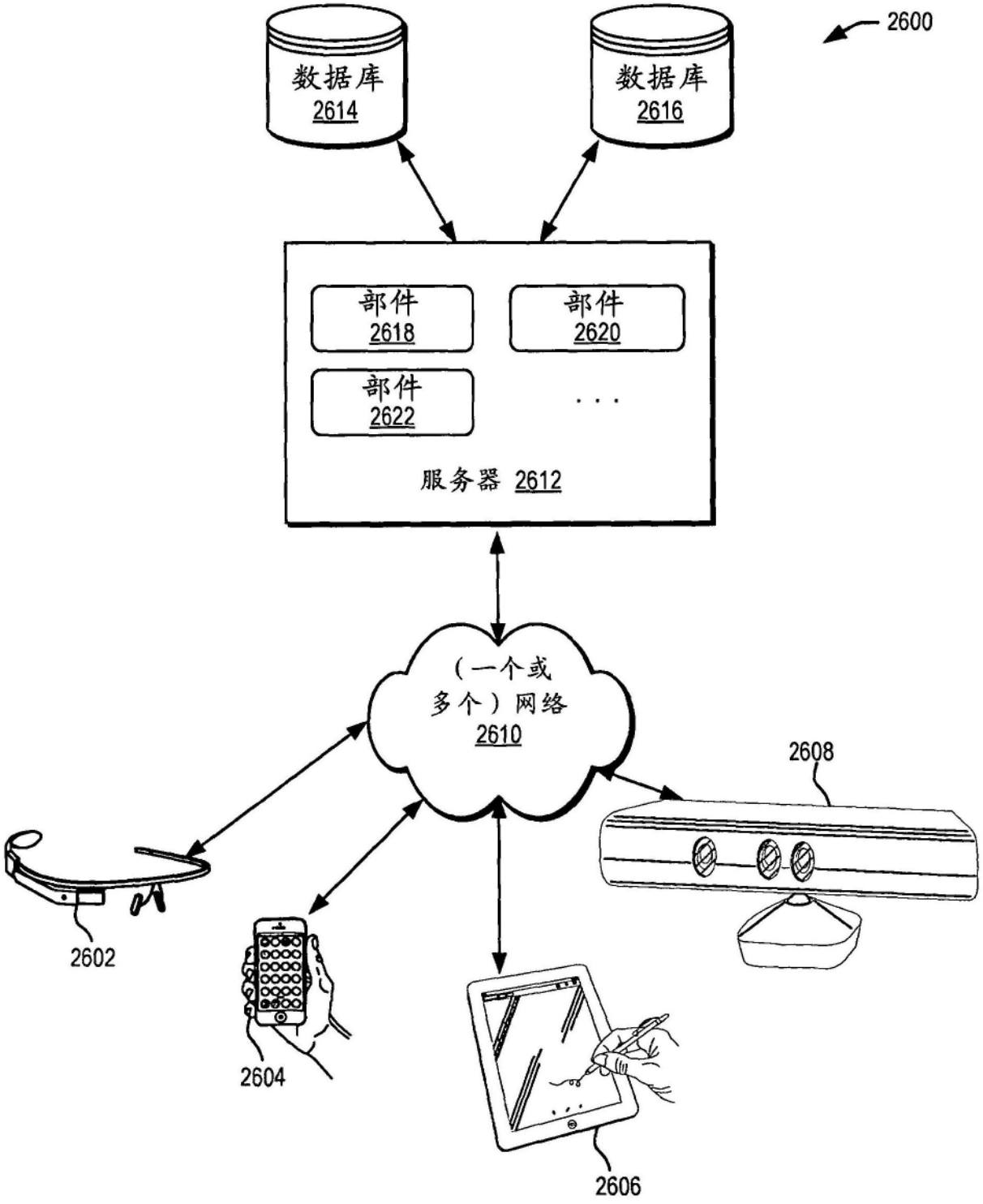

2600
数据库
2614
数据库
2616
部件
2618
部件
2620
部件
2622
. . .
服务器 2612
（一个或
多个）网络
2610
2608
2602
2604
2606

图26

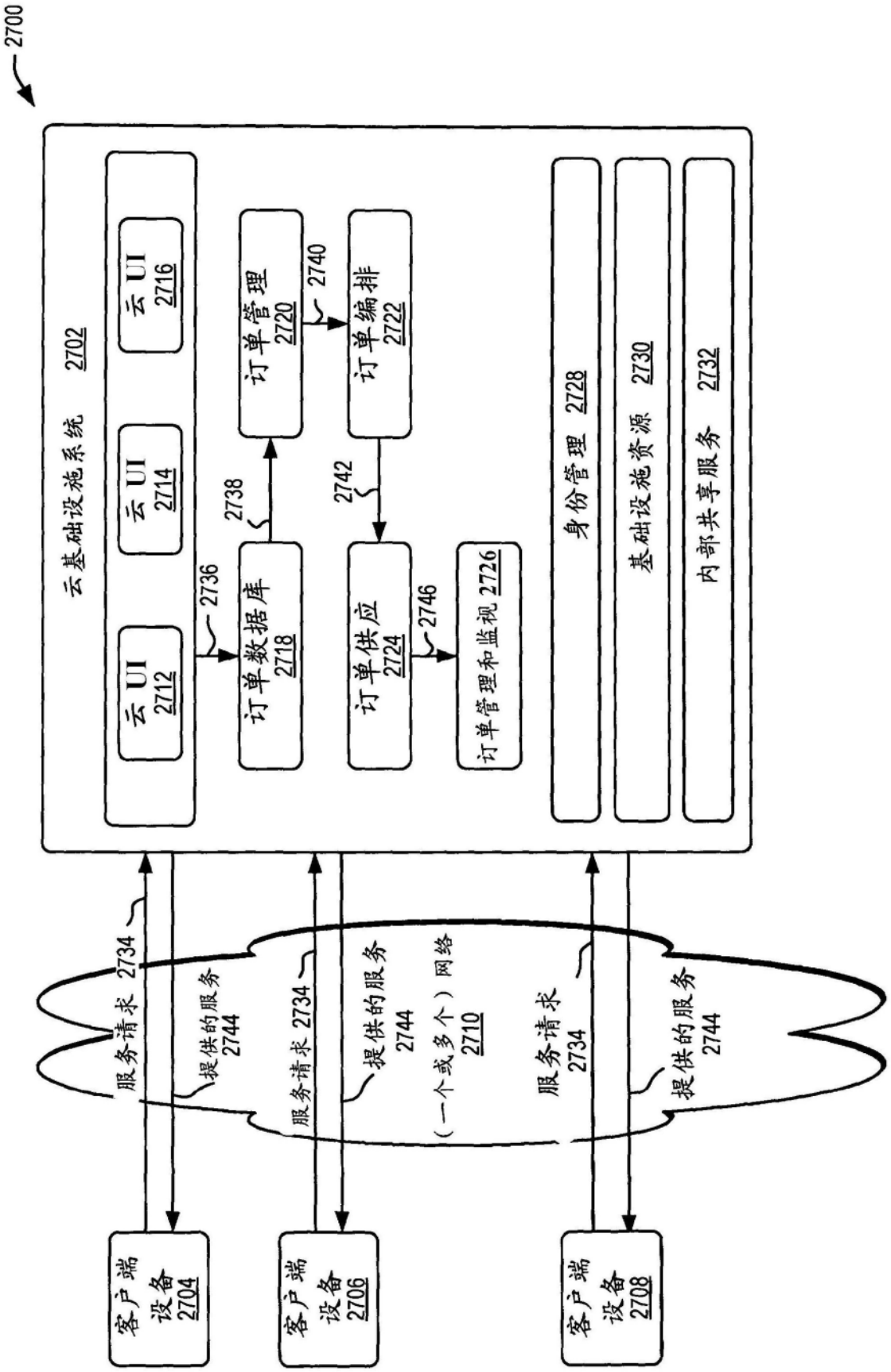

2700
云基础设施系统 2702
云 UI 2712
云 UI 2714
云 UI 2716
订单数据库 2718
订单管理 2720
订单编排 2722
订单供应 2724
订单管理和监视 2726
身份管理 2728
基础设施资源 2730
内部共享服务 2732
2734
2736
2738
2740
2742
2744
2746
服务请求 2734
提供的服务 2744
（一个或多个）网络 2710
客户端设备 2704
客户端设备 2706
客户端设备 2708

图27

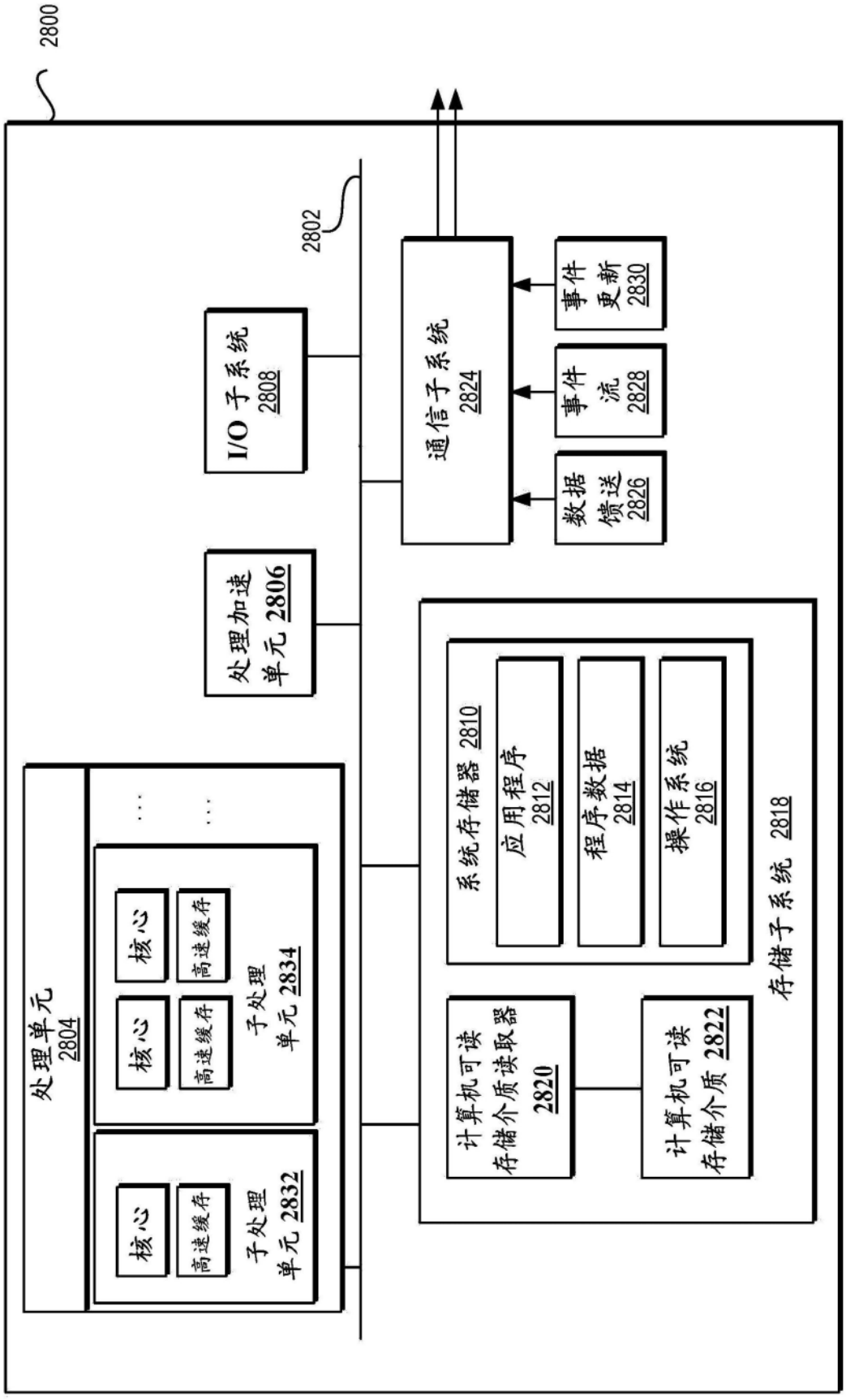

2800
2802
I/O 子系统 2808
处理加速单元 2806
通信子系统 2824
事件更新 2830
事件流 2828
数据馈送 2826
处理单元 2804
核心
高速缓存
子处理单元 2832
子处理单元 2834
系统存储器 2810
应用程序 2812
程序数据 2814
操作系统 2816
存储子系统 2818
计算机可读存储介质读取器 2820
计算机可读存储介质 2822

图28