(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 107077476 B
(45)授权公告日 2021.09.28

(21)申请号 201580048664.8
(22)申请日 2015.09.21
(65)同一申请的已公布的文献号
申请公布号 CN 107077476 A
(43)申请公布日 2017.08.18
(30)优先权数据
62/054,732 2014.09.24 US
14/755,088 2015.06.30 US
(85)PCT国际申请进入国家阶段日
2017.03.10
(86)PCT国际申请的申请数据
PCT/US2015/051268 2015.09.21
(87)PCT国际申请的公布数据
WO2016/048912 EN 2016.03.31
(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 A·德卡斯罗阿尔维斯
U·A·德什姆克 Y·贝德卡
P·斯卡拉姆 A·玛赫德鲁
(74)专利代理机构 中国贸促会专利商标事务所有限公司 11038
代理人 边海梅
(51)Int.Cl.
G06F 16/25(2019.01)
(56)对比文件
CN 101520861 A,2009.09.02
匿名."Oracle Event Processing NoSQL Database Data Cartridge-11g Release 1 (11.1.1.7)".《URL:https://web.archive.org/web/20130925033627/http://docs.oracle.com/cd/E28280_01/apirefs.1111/e12048/datacartnosql.htm》.2013,
匿名."Oracle Event Processing Hadoop Data Cartridge—11g Release 1 (11 .1 .1 .7)".《URL:https://web.archive.org/web/20130925035955/http://docs.oracle.com/cd/E28280_01/apirefs.1111/e12048/datacarthadoop.htm》.2013,
Francis Liu."HBaseCon 2014:HBase Design Patterns @ Yahoo!".《URL:https://vimeo.com/100518742》.2014,
审查员 张思洋

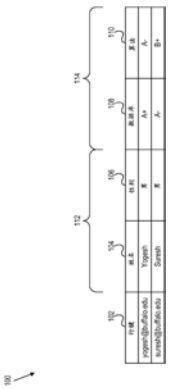
权利要求书3页 说明书34页 附图19页

(54)发明名称

利用动态类型的大数据对事件进行丰富以用于事件处理

(57)摘要

连续事件处理器能够接收事件的连续流,并且能够通过对每个事件应用CEP查询来处理该事件。本文中,公开了使用CQL处理器利用包含在数据存储库中的上下文数据来对源事件进行丰富的方法。在事件处理网络中,事件处理网络表组件被创建以表示数据存储库:该事件处理网络表组件具有映射组件,该映射组件指定从CQL事件特性到与数据存储库中的数据相关联的信息的映射。表组件然后被用作外部关系源,以执行包括以下操作的操作:从数据存储库中选择与事件相对应的数据,以及利用这样的数据对事件进行丰富。



CN 107077476 B

1.一种计算机实现的方法,包括:

创建事件处理网络以表示包括第一数据列类型的数据存储库,所述事件处理网络包括多个分布式节点,每个节点被配置为处理事件流中的事件;

在所述事件处理网络中创建表组件,所述表组件表示所述数据存储库的数据存储库事件,所述数据存储库事件与第一数据列类型相对应;

对于第一事件,使用所述表组件作为外部关系源来执行所述事件处理网络的处理器以执行操作,该操作包括:

从所述数据存储库中选择第一数据列类型的与第一事件相对应的第一上下文数据;

利用第一应用编程接口API方法调用,从所述数据存储库检索第一上下文数据;以及

将第一上下文数据添加到第一事件;

在创建所述事件处理网络之后向所述事件处理网络添加第二数据列类型;以及

对于第二事件,使用所述表组件作为所述外部关系源来执行所述事件处理网络的所述处理器以执行操作,该操作包括:

从所述数据存储库中选择第二数据列类型的与第二事件相对应的第二上下文数据;

利用第二应用编程接口API方法调用,从所述数据存储库检索第二上下文数据;以及

将第二上下文数据添加到第二事件。

2.如权利要求1所述的计算机实现的方法,其中所述第一应用编程接口API方法调用和所述第二应用编程接口API方法调用包括Java方法调用。

3.如权利要求1至2中任一项所述的计算机实现的方法,其中所述数据存储库包括非关系型数据库。

4.如权利要求3所述的计算机实现的方法,其中所述非关系型数据库包括HBase数据库。

5.如权利要求1、2或4中任一项所述的计算机实现的方法,其中所述处理器包括连续查询语言处理器。

6.一种存储计算机可执行指令的计算机可读介质,所述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创建事件处理网络以表示包括第一数据列类型的数据存储库的指令,所述事件处理网络包括多个分布式节点,每个节点被配置为处理事件流中的事件;

使得所述一个或多个处理器在所述事件处理网络中创建表示数据存储库的数据存储库事件的表组件的指令,所述数据存储库事件与第一数据列类型相对应;

使得所述一个或多个处理器使用所述表组件作为外部关系源来实现所述事件处理网络的引擎以执行以下指令的指令:

使得所述一个或多个处理器从所述数据存储库中选择第一数据列类型的与第一事件相对应的第一上下文数据的指令;

使得所述一个或多个处理器利用第一应用编程接口API方法调用,从所述数据存储库检索第一上下文数据的指令;以及

使得所述一个或多个处理器将第一上下文数据添加到第一事件的指令;

使得所述一个或多个处理器在创建所述事件处理网络之后将第二数据列类型添加到

所述事件处理网络的指令;以及

使得所述一个或多个处理器使用所述表组件来实现所述事件处理网络的所述引擎以进一步执行以下指令的指令:

使得所述一个或多个处理器从所述数据存储库中选择第二数据列类型的与第二事件相对应的第二上下文数据的指令;

使得所述一个或多个处理器利用第二应用编程接口API方法调用,从所述数据存储库检索第二上下文数据的指令;以及

使得所述一个或多个处理器将第二上下文数据添加到第二事件的指令。

7.如权利要求6所述的计算机可读介质,其中所述一个或多个计算机系统还被配置为执行使得所述一个或多个处理器创建映射组件的指令。

8.如权利要求7所述的计算机可读介质,其中所述映射组件被配置为指定第一事件的特性和与第一上下文数据相关联的第一信息之间的映射。

9.如权利要求8所述的计算机可读介质,其中与第一上下文数据相关联的第一信息包括与所述数据相关联的族或限定符中的至少一个。

10.如权利要求8或9所述的计算机可读介质,其中所述一个或多个计算机系统还被配置为执行使得所述一个或多个处理器在创建所述表组件之后更新与所述数据相关联的所述信息的模式的指令。

11.如权利要求10所述的计算机可读介质,其中通过添加第二数据列类型在实现所述事件处理网络的所述引擎时动态地更新所述模式。

12.如权利要求10所述的计算机可读介质,其中所述模式定义与所述数据相关联的族类型或限定符类型中的至少一个。

13.如权利要求6-9、11或12中任一项所述的计算机可读介质,其中所述引擎包括连续查询语言引擎。

14.一种计算系统,包括:

存储器,所述存储器存储多个指令;以及

处理器,所述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:

创建事件处理网络以表示包括第一数据列类型的数据存储库,所述事件处理网络包括多个分布式节点,每个节点被配置为处理事件流中的事件;

在所述事件处理网络中创建表示数据存储库的数据存储库事件的表组件,所述数据存储库事件与第一数据列类型相对应;

使用所述表组件作为外部关系源来实现所述事件处理网络的引擎,以:

从所述数据存储库中选择第一数据列类型的与第一事件相对应的第一上下文数据;

利用第一应用编程接口API方法调用,从所述数据存储库检索第一上下文数据;以及

将第一上下文数据添加到第一事件;

在创建所述事件处理网络之后将第二数据列类型添加到所述事件处理网络;以及

使用所述表组件来实现所述事件处理网络的所述引擎以进一步执行:

从所述数据存储库中选择第二数据列类型的与第二事件相对应的第二上下文数据;

利用第二应用编程接口API方法调用,从所述数据存储库检索第二上下文数据;以及

将第二上下文数据添加到第二事件。

15.如权利要求14所述的计算系统，其中所述事件处理网络的所述引擎被配置为至少部分地基于映射来识别第一上下文数据或者第二上下文数据。

16.如权利要求15所述的计算系统，其中所述映射被配置为在创建所述表组件之后被更新。

17.如权利要求15或16所述的计算系统，其中所述映射被包括在所述表组件中作为列族。

18.如权利要求14至16中任一项所述的计算系统，其中所述数据存储库包括数据存储位置的集群。

19.如权利要求14至16中任一项所述的计算系统，其中，以下各项中的至少一项成立：所述数据存储库包括非关系型数据库，或者所述引擎包括连续查询语言引擎。

20.一种包括用于执行如权利要求1-5中任何一项所述的方法的构件的装置。

利用动态类型的大数据对事件进行丰富以用于事件处理

[0001] 对相关申请的交叉引用

[0002] 本申请要求于2015年6月30日提交的美国专利申请No.14/755,088(代理人案号88325-937627(153500))的优先权和权益,该申请要求于2014年9月24日提交的美国临时申请No.62/054,732(代理人案号88325-908381(153501US))的优先权益,每个申请的全部公开通过引用被全部结合于此,用于所有目的。

背景技术

[0003] 数据库通常被用在需要存储数据和需要对所存储的数据的查询能力的应用中。因此,现有的数据库最适合于在有限的所存储的数据集合上运行查询。但是,传统的数据库模型不太适合于越来越多的如下现代应用,在这些现代应用中,数据作为数据事件的流而不是有界的数据集合被接收。数据流(也称为事件流)由实时的、潜在连续的事件序列表征。因此,数据或事件流表示无界的数据集合。生成数据流的源的示例包括被配置为发送传感器读数序列的传感器和探测器(例如,射频标识符(RFID)传感器、温度传感器等)、金融报价器、发送网络状态更新的网络监视和流量管理应用、点击流分析工具以及其它。

[0004] 连续事件处理(CEP)是用于处理事件流中的数据的技术。CEP是高度有状态(stateful)的。CEP涉及连续地接收事件,并且在这些事件中找到某种模式。因此,在CEP中涉及大量的状态维护。因为CEP涉及这么多状态的维护,所以对事件流内的数据应用CEP查询的进程总是单线程化的。在计算机编程中,单线程化是一次处理一个命令。

[0005] CEP查询处理通常涉及相对于在事件流内指定的事件连续地执行查询。例如,可以使用CEP查询处理,以便连续观察股票在最近一小时内的平均价格。在这种情景下,可以相对于事件流执行CEP查询处理,其中该事件流包含了这样的事件,其中各个事件在各个时间处指示了股票的当前价格。查询可以汇总过去一小时内的股票价格,并且然后计算这些股票价格的平均值。查询可以输出每个计算出的平均值。随着价格的一小时长的窗口移动,可以连续地执行查询,并且查询可以输出各个不同的平均股票价格。

[0006] 连续事件处理器能够接收事件的连续流,并且能够通过对包含在连续事件流中的每个事件应用CEP查询来处理该事件。这种CEP查询可以被格式化,以符合CEP查询语言的语法,其中CEP查询语言诸如作为结构化查询语言(SQL)的扩展的连续查询语言(CQL)。虽然SQL查询通常(针对用户请求)被应用到已经存储在关系数据库的表中的数据一次,但是CQL查询随着传入事件流中的事件被连续事件处理器接收到而被重复地应用到这些事件。因此,显然需要一种允许高效地执行CEP的系统。

发明内容

[0007] 本文描述的实施例涉及数据库和连续事件处理。根据一些实施例,CQL查询的处理可以跨不同处理节点分布。事件处理机制可以跨多个单独的虚拟机分布。所描述的实施例提供了一种高效地允许用于实时连续数据查询的CEP的系统。

[0008] 根据一些实施例,HBase数据库存储库被用作用于连续查询语言处理器(CQL处理

器)的数据源。这种使用允许由存在于该存储库中的数据对事件进行丰富,这类似于可以如何利用存在于RDBMS表中的数据对事件进行丰富。根据一些实施例,HBase数据库存储库被用作类似于表汇(table sink)特征的数据汇(data sink)。

[0009] 通过参考以下说明书、权利要求书和附图,前述内容以及其它特征和实施例将变得更加清楚。

附图说明

[0010] 图1是示出根据一些实施例的HBase数据存储库中的表的示例的图。

[0011] 图2是示出根据一些实施例的简单事件处理网络的示例的框图。

[0012] 图3是示出根据一些实施例的广播事件处理网络的示例的框图。

[0013] 图4是示出根据一些实施例的负载平衡事件处理网络的示例的框图。

[0014] 图5是示出根据一些实施例的负载平衡事件处理网络的后续状态的示例的框图。

[0015] 图6是示出根据一些实施例的其中通道具有两个消费者的广播事件处理网络的示例的框图。

[0016] 图7是示出根据本发明的实施例的、用于生成可用来请求来自多个资源服务器的服务的单个令牌的技术的示例的流程图。

[0017] 图8是示出根据一些实施例的分区式事件处理网络的示例的框图。

[0018] 图9是示出根据一些实施例的分区式事件处理网络的另一个示例的框图。

[0019] 图10是示出根据一些实施例的扇入(fan-in)事件处理网络的示例的框图。

[0020] 图11是示出根据一些实施例的线形图的示例的图。

[0021] 图12是示出根据一些实施例的散点图的示例的图。

[0022] 图13是示出根据一些实施例的在其中已绘制经平滑的曲线拟合器的散点图的示例的图。

[0023] 图14是示出根据一些实施例的在其中点具有不同尺寸的散点图的示例的图。

[0024] 图15是示出根据一些实施例的雷达图的示例的图。

[0025] 图16绘出了用于实现实施例之一的分布式系统的简化图。

[0026] 图17是根据本公开的实施例的系统环境的组件的简化框图,通过该系统环境,由实施例系统的组件提供的服务可以被提供为云服务。

[0027] 图18示出了在其中可以实现本发明的各种实施例的计算机系统的示例。

[0028] 图19是示出根据一些实施例的叠加在散点图上的表示集群的形状的示例的图。

具体实施方式

[0029] 在以下描述中,为了解释的目的,阐述了具体细节以便提供对本发明的实施例的透彻理解。但是,将清楚的是,可以在没有这些具体细节的情况下实践本发明。

[0030] 用于事件处理应用的处理可以是分布式的。Oracle事件处理(Oracle Event Processing)产品是事件处理器(连续查询语言处理器)的示例。根据一些实施例,CQL查询的处理可以跨不同处理节点分布。例如,每个这样的处理节点可以是单独的机器或计算设备。当跨不同处理节点分发CQL查询的处理时,以某种方式定义对事件进行排序的语义。

[0031] 用于对事件进行排序的初步方法试图在事件流中的事件之间维护先进先出

(FIFO)排序。但是,一些事件流化系统可以涉及多个事件发布者和多个事件消费者。系统中的每个机器可以具有它自己的时钟。在这种情景下,由任何单个机器产生的事件时间戳在整个系统中可能不是决定性的(definitive)。

[0032] 在涉及多个事件消费者的系统内,每个消费者可能具有单独的一组要求。每个消费者可以是连续执行CQL查询的事件处理器。就事件排序而言,每个这样的CQL查询可能具有单独的要求。

[0033] 根据一些实施例,定义了分发流(distribution flow)。每个分发流是在事件生产者和事件消费者之间对事件进行分发的特定方式。一种分发流可以在一组事件消费者之间对事件进行负载平衡。例如,当事件生产者产生第一事件时,负载平衡分发流可以使得第一事件被路由到第一事件消费者。随后,当该事件生产者产生第二事件时,负载平衡分发流可以使得第二事件被路由到第二事件消费者。

[0034] 其它种类的分发流包括分区分发流、扇入分发流、广播流等。取决于正在使用的分发流的类型,并且还取决于正在接收事件的事件消费者的要求,可以使用不同的事件排序技术对事件消费者接收的事件进行排序。

[0035] Map-Reduce(映射-缩减)概览

[0036] 本文对Map-Reduce进行引用,Map-Reduce是用于使用大量计算机器(节点)处理涉及巨大数据集合的可并行化问题的框架。如果所有节点在同一局域网上并且使用类似的硬件,则这些节点被统称为集群。可替代地,如果节点跨地理上和管理上的分布式系统被共享并且使用较多的异构硬件,则这些节点被统称为网格。计算处理可以相对于诸如可能在文件系统中找到的非结构化数据或诸如可能在数据库中找到的结构化数据来执行。Map-Reduce可以利用数据的局部性,从而处理在存储资产上或在存储资产附近的数据,以便减少数据被传输的距离。

[0037] 在“映射(map)”步骤中,主节点接收任务作为输入、将该输入划分为较小的子问题、并且将子问题分发给工作者节点。给定的工作者节点可以重复这种划分和分发,从而导致多级树结构。每个工作者节点处理分配给它的子问题,并将处理的结果传递回其主节点。

[0038] 在“缩减(reduce)”步骤中,主节点收集所有子问题的处理的结果。主节点以某种方式组合这些结果以形成最终输出。最终输出是最初给予主节点执行的任务的产物。

[0039] Map-Reduce允许映射操作和缩减操作的分布式处理。假设每个映射操作独立于其它映射操作,那么所有映射操作可以被并行执行。类似地,如果共享相同键的映射操作的所有输出在同一时间呈现给同一缩减者(reducer),或者如果缩减函数是关联性的,则一组缩减节点可以执行缩减阶段。除了降低产生最终结果所需的总时间之外,并行性还提供了在操作期间从服务器或存储装置的部分故障中恢复的某种可能性。如果一个映射节点或缩减节点故障,那么在输入数据仍然可用的情况下工作可以被重新调度。

[0040] Map-Reduce可以被概念化为五个步骤的并行和分布式计算。在第一步骤中,准备映射输入。Map-Reduce系统指派映射处理器、分配每个映射处理器可以在其上工作的第一输入键值、以及向每个映射处理器提供与该第一输入键值相关联的所有输入数据。

[0041] 在第二步骤中,映射节点执行用户提供的映射代码。映射节点对第一键值中的每一个执行一次映射代码。映射代码的执行生成按照第二键值组织的输出。

[0042] 在第三步骤中,来自第二步骤的输出被混洗(shuffled)到缩减节点。Map-Reduce

系统指派缩减处理器。Map-Reduce系统向每个缩减处理器分配第二键值，其中该处理器将在该第二键值上工作。Map-Reduce系统向每个缩减处理器提供在第二步骤期间产生的、也与该缩减处理器的被分配的第二键值相关联的所有数据。

[0043] 在第四步骤中，缩减节点执行用户提供的缩减代码。缩减节点对在第二步骤期间产生的第二键值中的每一个执行一次缩减代码。

[0044] 在第五步骤中，产生最终输出。Map-Reduce系统收集由第四步生成的所有输出数据，并且按照它们的第二键值对该数据进行排序，以产生最终输出。

[0045] 虽然上述步骤可以被想象为按顺序运行，但是实际上，只要最终输出不受交错的影响，这些步骤就可以被交错。

[0046] 从分发中受益的事件处理场景

[0047] 由于当今要被分析的数据量已经极大地增长，因此可缩放的事件处理机制是非常有用的。在本上下文中的可缩放性可以不仅涉及在执行事件处理中涉及的处理线程的数量的增加，而且还涉及可以并行处理事件的计算机器的数量的增加。本文公开了用于跨多个虚拟机(诸如JAVA虚拟机(JVM))分发事件处理应用的技术。

[0048] 许多不同的事件处理场景很好地适用于分布式执行。这些场景往往具有某些特点。首先，这些场景不是受延迟严格限制的，而是可以涉及例如微秒范围内的延迟。第二，这些场景能够例如按消费者或按区域被逻辑分区。第三，这些场景能够在逻辑上被划分为可以被并行执行的单独的各个组件或任务，使得不存在完全排序约束。

[0049] 可以以分布式方式有用地执行的事件处理场景的一个示例是单词计数场景。在此场景中，系统将传入句子映射为有意义的词语，并且然后将这些词语缩减为(针对每个词语的)计数。在该单词计数场景中执行的工作可以使用Map-Reduce批处理(batching)来执行，但是也可以使用流处理来执行。这是因为，使用流处理可以对实时单词流进行计数，该实时单词流诸如来自Twitter(推特)或另一社交媒体馈送的实时单词流。相对于社交媒体馈送使用流处理允许比使用其它处理方法可能得到的反应更快的反应。

[0050] 如果使用流处理来处置社交媒体馈送(例如对这些馈送中的单词进行计数)，则流处理机制可能经受非常大量的传入单词。为了处置大量的信息，信息的处理可以被分发。单独的计算机器可以订阅不同的社交媒体流，诸如Twitter流。这些机器可以并行处理流、对其中的单词进行计数、并且然后聚合(converge)结果得到的计数以产生完整的结果。

[0051] 可以以分布式方式有用地执行的事件处理场景的另一个示例是矩阵乘法场景。互联网搜索引擎使用的页面排名算法可以将网页的重要性总结作为单个数。这种算法可以实现为一系列级联的大矩阵乘法操作。

[0052] 因为矩阵乘法可以被高度并行化，所以Map-Reduce可以有益于执行涉及矩阵乘法的操作。矩阵乘法操作可以被概念化为自然连接(natural join)，随后是分组和汇总。

[0053] 可以以分布式方式有用地执行的事件处理场景的另一个示例是词语频率-反向文档频率(term frequency-inverted document frequency，TF-IDF)场景。TF-IDF是常常被搜索引擎采用以确定词语的重要性的算法。与期望相反的是，如果词语频繁见于其它文档中，则该词语较不重要，因此，算法的“反向文档频率”方面不太重要。

[0054] 与以上讨论的词计数场景一样，使用流处理能够实时地执行TF-IDF处理是有价值的。与单词计数场景中不同，TF-IDF值的计算涉及为“反向文档频率”计算访问历史文档。对

历史文档的涉及使得TF-IDF场景成为与Hadoop和/或某种索引层(例如HBase)一起使用的良好候选。Hadoop和HBase以及它们在事件流处理中的使用也在本文进行讨论。

[0055] 可以以分布式方式有用地执行的事件处理场景的另一个示例是智能仪表(smart meter)能量消耗场景。今天的家庭通常通过使用位于他们的房屋中的智能仪表来收集他们的能量消耗。通常,这些智能仪表以事件的形式在一天中周期性地(例如,每分钟)输出能量消耗传感器数据。该传感器数据由区域处理中心中的下游管理系统捕获。这些中心使用捕获的数据来计算有用的描述性统计数据,诸如房屋或邻近地区的平均能量消耗。统计数据可以揭示平均能量消耗与该区域的历史数据如何相关。这些运行中的汇总非常适合被分区。因此,分布式分区流可以被有益地应用于此场景。

[0056] 事件处理可以相对于这种信息来执行,以便识别异常值(诸如高于或低于能量消耗的典型范围的家庭)。事件处理可以相对于这种信息来执行,以便尝试预测将来的消耗。识别出的异常值和预测的将来消耗可以被能量提供商使用,以用于差异化定价、促销、以及更高效地控制与他们的合作伙伴的能量购买和销售过程。

[0057] 未在以上具体列举的各种其它场景可以适于以分布式方式进行处理。例如,可以使用分布式事件流处理来执行风险分析,这些风险分析涉及随着衍生物价格的变化实时计算金融投资组合的风险暴露(exposure)。

[0058] 分发事件流以创建流的方法

[0059] 本文公开了用于跨多个事件处理节点分发事件流(例如,源自特定数据源的流)以便于并行事件处理的若干不同技术。每种技术创建不同种类的流。这些技术中的一些总结如下。

[0060] 分区流涉及使用流中的事件的一个或多个属性作为分区标准跨若干单独的计算资源(诸如处理线程或虚拟机(例如,JVM))对事件流进行分区。本文公开了分区流的集群版本。本文还公开了跨处于单节点配置的事件处理网络的线程对经流化的事件进行分区。

[0061] 扇入(fan-in)流涉及将多个先前分发的事件流聚拢回到单个计算资源中。例如,当某个状态要被共同定位(co-located)时,诸如在其中涉及全局汇总的情景下,可以使用扇入流。

[0062] 负载平衡(load-balance)流涉及将流的事件分发到一组消费监听者,这是以总负载以平衡的方式跨该组消费监听者被共享的方式。当前加载较少工作的事件处理节点可以先于当前加载较多工作的事件处理节点被选择,以接收用于处理的新事件。这防止任何一个事件处理节点变得过载而其它事件处理节点还未被充分利用。

[0063] 广播(broadcast)流涉及将流的所有事件向所有消费监听者广播。在这种情况下,所有监听者——诸如事件处理节点——都会接收到所有事件的拷贝。

[0064] 集群域生成

[0065] 为了支持对应于以上讨论的各种流的分布式事件处理网络,一些实施例涉及集群域的生成。在一些实施例中,配置向导或其它工具引导用户通过被配置用于支持分布式流的域的生成。

[0066] 资源弹性

[0067] 在云计算环境中,计算资源可以随着需求增加或减少而动态地增长或收缩。根据一些实施例,部署在云计算环境中的分布式事件处理系统可插入到现有基础设施中。系统

可以动态地增长和收缩当前执行分布式流的计算资源的数量。例如，在越来越高负载的情况下，负载平衡流可以自动产生(spaw)新的计算资源以进一步共享负载。

[0068] 定义为事件处理网络的分布式流

[0069] 对应于流的事件处理网络可以表示为非循环有向图(acyclicdirected graph)。该图可以形式上定义为对(N,C)，其中N是一组节点(顶点)，并且C是N上的表示从源节点到目的地节点的连接(弧)的两位关系(two-place relation)。

[0070] 例如，事件处理网络可以被定义为event processing network1＝({adapter1,channel1,processor,channel2,adapter2},{(adapter1,channel1),(channel1,processor),(processor,channel2),(channel2,adapter2)})(事件处理网络1＝({适配器1,通道1,处理器,通道2,适配器2},{(适配器1,通道1),(通道1,处理器),(处理器,通道2),(通道2,适配器2)}))。事件被定义为表示属性名称和属性值的任何对(PN,PV)的关系P。

[0071] 对于另一个示例，给定表示股票报价器的事件流，可以使用以下定义：e1＝{(price,10),(volume,200),(symbol,'ORCL')};e2＝{(p1,v1),(p2,v2),(p3,v3)},(e1＝{(价格,10),(体量,200),(股票代码,'ORCL')};e2＝{(p1,v1),(p2,v2),(p3,v3)})。由于事件处理网络节点可以包含多于一个事件，因此集合E可以被定义为事件的有序序列(这与在一些其它情况下不同)。

[0072] 对于另一个示例，可以使用以下定义：{processor}＝{e1,e2}({处理器}＝{e1,e2})。事件处理网络的运行时状态S＝(N,E)可以表示为从N到E的两位关系。关系S不是单射的，意味着相同(一个或多个)事件可以存在于多于一个节点中。但是，关系S是满射的，因为事件处理网络中的事件的总集合中的所有事件都在至少一个节点中。

[0073] 对于另一个示例，可以使用以下定义：state＝{(processor,{e1,e2}),(adapter2,{e3})}(状态＝{(处理器,{e1,e2}),(适配器2,{e3})})。这提供了事件流的逻辑模型。该模型可以用物理组件扩充。这种扩充可以通过分配托管事件处理网络的节点的计算资源R来完成。新模型然后变为三位关系S＝(N,R,E)，其中R是集群的所有计算资源的集合。

[0074] 对于另一个示例，可以使用以下定义：state＝{(processor,machine1,{e1,e2}),(adapter2,machine1,{e3})}(状态＝{(处理器,机器1,{e1,e2}),(适配器2,机器1,{e3})})。

[0075] 因此，分布式流可以被定义为函数。这些函数可以将事件处理网络的静态结构、运行时的当前状态以及作为主体的特定计算资源的特定节点作为输入。该函数可以返回运行时状态的新配置，该新配置考虑事件从主体到该主体的连接的流动。形式上，这可以被定义为：distribute-flows:nεN,rεR,C,S→S(分发流:nεN,rεR,C,S→S)。

[0076] 定义了若干函数以支持用于扇入流、负载平衡流、分区流和广播流的模式，后两者具有两个版本：一个用于单个虚拟机情况，而另一个用于集群虚拟机情况。分发流函数是：local-broadcast(n,r,C,S)＝{S-{(n,r,e)}+S'其中$\forall n$[(n,d)εC∧(n,r,e)εS→(d,r,e)εS']}。

[0077] 这些按如下解读：对于在C和S中都存在的所有源n(即，具有连接并且具有事件)，然后对于C中的每个目的地d，生成新的元组状态(d,r,e)。返回当前状态S减去旧的元组(n,r,e)加上新的元组(d,r,e)。在最后步骤中的移除和添加表示事件从源节点到目的地节点

的移动。

[0078] clustered-broadcast(n,r,C,S)＝{S-{(n,r,e)}+S'：∀n[(n,d)εC∧(n,r,e)εS→∀t[(d,t,e)εS'∧tεR]]}。在这种情况下，新的状态S'包括目的地d和资源t的所有有效排列的元组。即，所有计算资源都将接收用于每个被配置的目的地的事件。

[0079] local-partition(n,pnεPN,r,C,S)＝{S-{(n,r,e)}+S'：∀n[(n,d)εC∧(n,r,e)εS→(d,r,e)εS']}。由于线程化未在这个定义中建模，因此本地分区(local-partition)和本地广播(local-broadcast)之间没有区别。但是在实践中情况并非如此，因为线程化按分区进行着色(colored)。

[0080] clustered-partition(n,pnεPN,r,C,S)＝{S-{(n,r,e)}+S'：∀n[(n,d)εC∧(n,r,e)εS→∃!t[(d,t,e)εS'∧t＝p-sched(e,pn)]]}。

[0081] load-balance(n,r,C,S)＝{S-{(n,r,e)}+S'：∃!t[∀n[(n,d)εC∧(n,r,e)εS→(d,t,e)εS'∧t＝lb-sched(R)]]}。

[0082] fan-in(n,r,C,S)＝{S-{(n,r,e)}+S'：∀n[(n,d)εC∧(n,r,e)εS→∃!t[(d,t,e)εS'∧t＝fi-sched(n,R)]]}。

[0083] 如将从以下讨论中将看出的，这最后三者在结构上类似，仅关于它们的调度函数有所不同。事实上，扇入可以被看作具有单个键的分区的特殊情况。

[0084] 调度函数

[0085] 根据一些实施例，定义了调度函数lb-sched、p-sched和fi-sched。这些函数的实现不会改变分布的结构。函数确定资源的调度。函数的默认实现是：lb-sched(R)＝{R→rεR:r＝round-robin(R)}。lb-sched使用常规的轮询(round-robin)算法。在这种情况下，可以维护某个集群状态。

[0086] 根据一些实施例，使用了最小作业(min-jobs)调度，其中最小作业选择到目前为止具有被调度运行的最小数量的作业的资源。

[0087] 根据一些实施例，随机选择目标服务器。考虑大数定律，除了不需要集中化状态之外，这种实施例类似于轮询：p-sched(e,pn)＝{eεE,pnεPN→rεR:r＝hash(prop(e,pn))mod|R|}；以及fi-sched(n,R)＝{nεN,R→rεR:r＝user-configured-server(n,R)}。

[0088] 在一些实施例中，执行到具有处理该事件集合所需的资源的单个服务器的聚合。例如，如果事件是要输出到事件数据网络/JAVA消息传送服务，则这样的服务器可以维护关于事件数据网络/JAVA消息传送服务服务器和目的地配置的信息。

[0089] 在一些实施例中，扇入目标的指定默认情况下被选择为具有最低成员ID的集群成员，这将指示要被配置的成员的第一服务器。

[0090] 示例事件处理网络

[0091] 图2是示出根据一些实施例的简单事件处理网络200的示例的框图。在图2中，事件处理网络200可以使用以下语法来定义：local-broadcast(channel1,machine1,event processing network1,{(channel1,machine1,{e1})})＝{(processor,machine1,{e1})}(local-broadcast(通道1,机器1,事件处理网络1,{(通道1,机器1,{e1})})＝{(处理器,机器1,{e1})})。

[0092] 以下讨论的是一些集群情况，其中R＝{machine1,machine2,machine3}(R＝{机器

1,机器2,机器3})。图3是示出根据一些实施例的广播事件处理网络300的示例的框图。在图3中,事件处理网络300可以使用以下语法来定义:clustered-broadcast(channel1,machine1,event processing network1,{(channel1,machine1,{e1})})＝{(processor,machine1,{e1}),(processor,machine2,{e1}),(processor,machine3,{e1})}(clustered-broadcast(通道1,机器1,事件处理网络1,{(通道1,机器1,{e1})})＝{(处理器,机器1,{e1}),(处理器,机器2,{e1}),(处理器,机器3,{e1})})。在这种情况下,同一事件e1被分发到计算资源机器1、机器2和机器3中的全部。

[0093] 图4是示出根据一些实施例的负载平衡事件处理网络400的示例的框图。在图4中,事件处理网络400可以使用以下语法来定义:load-balance(channel1,machine1,event processing network1,{(channel1,machine1,{e1})})＝{(processor,machine2,{e1})}(load-balance(通道1,机器1,事件处理网络1,{(通道1,机器1,{e1})})＝{(处理器,机器2,{e1})})。在进行负载平衡的情况下,机器2的状态在执行对于{e1}的作业时被改变。

[0094] 图5是示出根据一些实施例的负载平衡事件处理网络500的后续状态的示例的框图。图5示出了在将e1发送到机器2之后图4的负载平衡事件处理网络的状态。在图5中,事件处理网络500可以使用以下语法来定义:load-balance(channel1,machine1,event processing network1,{(channel1,machine1,{e2,e3})})＝{(processor,machine3,{e2,e3})}(load-balance(通道1,机器1,事件处理网络1,{(通道1,机器1,{e2,e3})})＝{(处理器,机器3,{e2,e3})})。在这种情况下,e2和e3两者都被发送到同一机器。这是因为它们两者同时存在于源节点中,因此将它们保持在一起是有意义的。

[0095] 在一些事件处理网络中,单个通道可以具有两个消费者,诸如event processing network 2＝({adapter1,channel1,processor1,processor2,channel2,channel3,adapter2,adapter3},{(adapter1,channel1),(channel1,processor1),(processor1,channel2),(channel2,adapter2),(channel1,processor2),(processor2,channel3),(channel3,adapter3)})(事件处理网络2＝({适配器1,通道1,处理器1,处理器2,通道2,通道3,适配器2,适配器3},{(适配器1,通道1),(通道1,处理器1),(处理器1,通道2),(通道2,适配器2),(通道1,处理器2),(处理器2,通道3),(通道3,适配器3)}))。在处于这种情景下的集群广播的情况下,所有机器中的所有处理器都接收事件。

[0096] 图6是示出根据一些实施例的、其中通道具有两个消费者的广播事件处理网络600的示例的框图。在图6中,事件处理网络600可以使用以下语法来定义:clustered-broadcast(channel1,machine1,event processing network2,{(channel1,machine1,{e1})})＝{(processor1,machine1,{e1}),(processor2,machine1,{e1}),(processor1,machine2,{e1}),(processor2,machine1,{e1}),(processor,1machine3,{e1}),(processor2,machine3,{e1})}(clustered-broadcast(通道1,机器1,事件处理网络2,{(通道1,机器1,{e1})})＝{(处理器1,机器1,{e1}),(处理器2,机器1,{e1}),(处理器1,机器2,{e1}),(处理器2,机器1,{e1}),(处理器,1机器3,{e1}),(处理器2,机器3,{e1})})。在机器(例如,机器1)内,事件(例如,e1)到其消费监听者(例如,处理器1、处理器2)的分派可以取决于排序要求同步地(即,同一线程)或异步地(即,不同线程)发生。

[0097] 图7是示出根据一些实施例的、其中通道具有两个消费者的负载平衡事件处理网络700的示例的框图。在这种情况下,存在多个监听者。在图7中,事件处理网络700可以使用

以下语法来定义:load-balance(channel1,machine1,event processing network2,{(channel1,machine1,{e1})})＝{(processor1,machine1,{e1}),(processor2,machine1,{e1})}(load-balance(通道1,机器1,事件处理网络2,{(通道1,机器1,{e1})})＝{(处理器1,机器1,{e1}),(处理器2,机器1,{e1})})。在这种情况下,事件被发送到单个成员的所有监听者。换句话说,只有到达的下一个事件将被负载平衡到不同的服务器或机器。

[0098] 分区场景和扇入场景可以被认为再次使用简单事件处理网络。图8是示出根据一些实施例的分区式事件处理网络800的示例的框图。在图8中,事件处理网络800可以使用以下语法来定义:partition(channel1,machine1,event processing network1,{(channel1,machine1,{e1(p1,1)})}):-{(processor,machine1,{e1(p1,1)})}(partition(通道1,机器1,事件处理网络1,{(通道1,机器1,{e1(p1,1)})}):-{(处理器,机器1,{e1(p1,1)})})。接下来,事件e2可以被认为在与事件e1相同的分区上,但事件e3在不同的分区上。这可以使用以下语法来定义:partition(channel1,machine1,event processing network1,{(channel1,machine1,{e2(p1,1)})}):-{(processor,machine1,{e2(p1,1)})};partition(channel1,machine1,event processing network1,{(channel1,machine1,{e3(p1,2)})}):-{(processor,machine2,{e3(p1,2)})}(partition(通道1,机器1,事件处理网络1,{(通道1,机器1,{e2(p1,1)})}):-{(处理器,机器1,{e2(p1,1)})};partition(通道1,机器1,事件处理网络1,{(通道1,机器1,{e3(p1,2)})}):-{(处理器,机器2,{e3(p1,2)})})。

[0099] 根据一些实施例,在分区式事件处理网络中,事件到达所有机器而不是仅到达某些机器。图9是示出根据一些实施例的分区式事件处理网络900的示例的框图。在一些实施例中,处理器可以具有馈送到该处理器中的多个上游通道。这种境况类似于处理多个事件。

[0100] 图10是示出根据一些实施例的扇入事件处理网络1000的示例的框图。在图10中,事件处理网络1000可以使用以下语法来定义:fan-in(channel1,machine1,event processing network1,{(channel1,machine1,{e1})}):-{(processor,machine1,{e1})};fan-in(channel1,machine2,event processing network1,{(channel1,machine2,{e2})}):-{(processor,machine1,{e2})}(fan-in(通道1,机器1,事件处理网络1,{(通道1,机器1,{e1})}):-{(处理器,机器1,{e1})};fan-in(通道1,机器2,事件处理网络1,{(通道1,机器2,{e2})}):-{(处理器,机器1,{e2})})。在扇入的情况下,事件被一起聚拢在机器1中。

[0101] 排序和查询处理语义

[0102] 在一些实施例中,事件不是瞬间从一个节点出现到所有其它目的地节点中;在这种实施例中,不是始终维护完全次序。可以在某些场景中安全地放宽排序要求,而不破坏分布模型的语义和查询处理模型的语义。

[0103] 存在两个维度要被考虑,即,针对单个事件的机器目的地之间的排序的维度,以及在发到目的地时事件本身的排序的维度。每个维度都可以被单独考虑。

[0104] 对于目的地排序,在负载平衡网络、分区网络和扇入网络中,由于存在单个目的地(即,$\exists!t$),所以目的地排序是不适用的。在集群广播网络中,由于一般的广播性质,不需要设想排序保证。

[0105] 对于事件排序，在负载平衡网络中，由于不能保证事件一开始将被发送到同一资源，所以保证事件的排序没有益处，因此下游查询处理在一些实施例中不会试图使用应用时间排序。此外，在一些实施例中，下游查询处理不依赖于接收所有事件，并且因此是无状态的。落入此标准中的查询的类型是过滤和流连接(1-n流关系连接)。

[0106] 在集群广播网络中，所有服务器具有完整的事件集合并且因此具有处理的全状态(full state)，因此在这种网络中，在每个服务器(即目的地)的上下文中保持次序。

[0107] 在分区网络中，对特定的目的地——排序在分区内得到保证。这允许下游查询处理同样地利用分区排序约束。在一些实施例中，即使存在正在馈送要被分区的事件的多个上游节点(如在以上结合图9所述的情况之一中)，此排序也得到保证。

[0108] 在扇入网络中，做出关于事件一开始如何被派生(fork)的确定，如下：如果事件是负载平衡的，则不存在排序保证，并且扇入函数也不施加任何次序。以下是在一个实施例中如果事件被分区则所发生的情况的示例：

[0109] 输入：{{t4,b},{t3,a},{t2,b},{t1,a}}

[0110] 分区a：{{t3,a},{t1,a}}

[0111] 分区b：{{t4,b},{t2,b}}

[0112] 调度1：{{t4,b},{t3,a},{t1,a},{t2,b}}

[0113] 调度2：{{t4,b},{t1,a},{t2,b},{t3,a}}

[0114] 调度3：{{t4,b},{t3,a},{t2,b},{t1,a}}

[0115] 在上游分区的情况下，扇入可能最终以与原始输入的次序不同的次序来排列事件。为了避免这种情况，尽管事件是从不同源接收到的，扇入网络也对它们进行排序。

[0116] 为了应对这些不同的场景，在不同场景中使用不同的语义。在无序场景的情况下，在事件之间不需要按照事件的时间戳的排序保证。

[0117] 在部分分区有序场景的情况下，根据一些实施例，保证事件按照其时间戳在源和目的地节点对的上下文中以及在分区的上下文中是有序的(即，a<=b)。换句话说，来自不同上游服务器的事件不能保证是有序的，并且去往不同分区的事件同样不能保证是有序的。

[0118] 在完全分区有序场景的情况下，保证事件按照其时间戳跨所有源和目的地节点对并且在分区的上下文中是有序的(即，a<=b)。为了能够支持这种模式，可以跨集群施加单一视角的时间戳。应用时间戳可以用于这种情况。

[0119] 在部分有序场景的情况下，在一些实施例中，保证事件按照其时间戳在源和目的地节点对的上下文中是有序的(即，a<=b)。

[0120] 在完全有序的情况下，在一些实施例中，保证事件按照其时间戳跨所有源和目的地节点对是有序的(即，a<=b)。

[0121] 已经按照最小约束到最大约束呈现了这些约束。为了支持这些不同的约束，在一些实施例中，使用以下附加配置。应用时间戳记(stamped)的属性是要用于完全次序标准的事件属性。超时属性指示在继续之前等待上游事件的时间。乱序(out-of-order)策略指示如果事件确实乱序到达则这些事件是应该被丢弃、被提出作为错误还是被发送到死信(dead-letter)队列。

[0122] 在一些实施例中，每个分发流可以与不同的排序约束集合一起使用。在负载平衡

网络中，约束可以是无序的。在广播网络中，输入流的所有事件可以以它们在广播通道上被接收到的相同次序传播到网络的所有节点。每个节点可以维护全状态，并且因此，监听广播通道的每个节点对于任何时间戳具有完全相同的状态。因此，这些节点中的每一个节点的监听器下游可以以完全次序接收输出事件，并且在本地广播中约束可以是完全有序的。对于集群广播，跨网络的事件递送可能导致事件的无序化，但是按照定义，递送应该是完全有序的，使得网络可以满足有序递送的要求。在分区式网络中，每个节点可以维护部分状态并且接收事件的子集。接收到的子流的事件处于与在输入流中观察到的次序相同的次序中。因此，输入流被跨分区(每个节点上一个分区)排序，并且在分区式网络中约束可以是分区有序的。在集群分区中，跨网络的事件递送可能导致事件的无序化，但是再次地，按照定义，递送应该是完全有序的，使得网络可以满足有序递送的要求。在扇入网络中，约束可以是无序的、部分有序的或完全有序的。

[0123] 此外，如果目的地节点是CQL处理器并且其查询是已知的，则可以从这些查询中推断出分发排序约束。例如，如果所有查询都被配置为分区有序的，则分发流也可以至少被设置为部分分区有序的。

[0124] 部署计划

[0125] 在一些实施例中，计算资源跨节点共享。为了允许更好地共享资源，可以利用一组约束要求对节点进行注释，诸如‘memory>1M(存储器>1M)’或‘thread>3(线程>3)’，并且反过来，可以利用一组能力对计算资源进行注释，诸如‘memory＝10M(存储器＝10M)’或‘thread-pool＝10(线程池＝10)’。

[0126] 例如，要求可以被表示为requirements:{processor1}＝{threads>3}(要求:{处理器1}＝{线程>3})。能力可以被表示为capabilities:{machine1}＝{max-thread-pool＝10,cpu＝8}(能力:{机器1}＝{最大线程池＝10,cpu＝8})。

[0127] 在将资源调度到节点期间，系统试图将要求与能力匹配，并且通过这样做，系统随着能力被分配给节点动态地减少和增加能力的当前值。例如，调度可以被表示为Schedule-1:{processor1}＝{threads>3,computing-resource＝machine1};Schedule-1:{machine1}＝{max-thread-pool＝10,current-thread-pool＝7}(调度-1:{处理器1}＝{线程>3,计算资源＝机器1};调度-1:{机器1}＝{最大线程池＝10,当前线程池＝7})。

[0128] 此外，集群本身的总能力可以例如通过向集群添加新的计算资源以应对应用负载的增加而改变。这被知晓为计算弹性。例如，在t＝0:{cluster}＝{machine1,machine2}({集群}＝{机器1,机器2})，但是在t＝1:{cluster}＝{machine1,machine2,machine3}({集群}＝{机器1,机器2,机器3})。系统应对这些动态的资源变化。

[0129] 可能存在系统的操作者想要将节点手动分配到具体计算资源的情况。这可以通过将‘computing-resource(计算资源)’视为要求本身来支持。例如，Requirements:{processor1}＝{threads>3,computing-resource＝machine1}(要求:{处理器1}＝{线程>3,计算资源＝机器1})。这种部署要求的指定被知晓为部署计划，并且可以被包括在应用元数据内。

[0130] 集群成员配置和域配置

[0131] 在Hadoop中，在Map-Reduce系统开始处的映射函数被复制到分布式任务并且被并行执行，每个函数读取单独的输入数据，或最常见地每个函数读取输入数据的块。流处理是

类似的。上游节点(即,入站适配器)各自订阅不同的流或流的不同分区。这意味着在一些实施例中,分布式事件处理网络允许入站适配器并行工作。没有必要使辅助节点(secondary nodes)中存在的入站适配器保持暂停状态。

[0132] 在一些实施例中,每个入站适配器可以订阅不同的流或流的不同分区。这可以通过在事件处理中使用集群成员(Clustered Member)设施来完成,其中成员能够发现它是否是主要的,并且成员在集群中成员与唯一ID相关联并且因此可以使用这个ID作为到流或流分区配置的键。

[0133] 辅助成员也可以选择不订阅任何事件,在这种情况下,系统的输入方面不是并行执行的。

[0134] 成本复杂性和批处理

[0135] 分布式系统中的通信成本可以轻易地超过处理数据本身的成本。事实上,Hadoop中的一个常见问题是尝试在具有太多的缩减者并因此增加通信成本和具有太少的缩减者并因此具有与键相关联的太多元件并且因此每缩减者没有足够存储器之间找出最佳折衷。

[0136] 为了便于理解这种成本和用于应对它的机制,提供了以下内容。延迟度量被计算为事件的总延迟与事件的通信延迟的比率。这是针对事件的某个采样速率进行的,并且可以在运行时动态地打开和关闭。在一些实施例中,存在使用Batching API(批处理API)一起发送的事件确实在整个分发过程中被分批在一起的保证。

[0137] 行为视点

[0138] 在一些实施例中,高速缓存一致性(cache coherence)可以用于消息传送以及分区二者。协作的语义按照流而变化,并且可以利用特定高速缓存方案(scheme)、高速缓存键和过滤的组合来实现。发送者(源)将事件插入(即,put())到高速缓存中,并且接收者(目标)从高速缓存中移除(即,get/delete)该事件。具有Filtered Events(经过滤的事件)的MapListener(映射监听者)API可以用来保证正确的接收者获得正确的事件集合。然而,如果事件是要作为单独的动作被接收到并且然后被删除,那么这会导致两个单独的网络操作。因此,在一些实施例中,允许事件按照其自身的价值(merit)过期(expire)。这样,高速缓存一致性就会批处理事件的删除并在适当的时候做这件事。

[0139] 在一些实施例中,同一高速缓存服务对于每种流类型的所有应用共享。例如,可以存在用于所有复制流的单个高速缓存服务、用于分区流的另一个高速缓存服务等。由于锁定可以每条目地进行,因此由一个应用中的一个通道处置一个事件不会影响其它应用,并且这避免了高速缓存在单个服务器中的扩增(proliferation)。

[0140] 在广播流的情况下,由于所有事件要被所有成员接收,因此可以使用复制高速缓存方案(Replicated Cache Scheme)。高速缓存键是成员ID、应用名称、事件处理网络阶段名称(例如,通道名称)、以及事件时间戳(无论它是基于应用的还是基于系统的)的散列。

[0141] CacheKey=hash(memberld,applicationName.stageName,eventTimestanp)

[0142] 高速缓存值是添加有应用ID、事件处理网络阶段ID以及目标ID的(一个或多个)原始事件的包装器(wrapper),其中目标ID被设置为-1以表示所有成员。应用ID和阶段ID是原始的应用名称和阶段名称的散列值,它们是用户设置的串。包装器可以包括事件时间戳(如果不是基于应用属性的)和事件种类(即,插入、删除、更新、心跳(heartbeat))。

[0143] CacheValue={applicationId,stageId,eventTimestampeventKind,

sourceEvents}

[0144] 在一些实施例中,所有集群成员注册MapListener,其中过滤器(Filter)被设置到应用ID、以及用于所讨论的广播通道的事件处理网络阶段ID。这意味着当充当发送者角色的成员将事件放入到广播高速缓存时,充当接收者角色的所有成员可以在MapListener.entryInserted(MapEvent)上被回调。

[0145] 如果流被设置为无序的,则MapListener是异步的并且可以使用一致性线程用于下游处理。如果流被设置为有序的,则可以注册SynchronousMapListener(同步映射监听者),并且该事件可以被立即移交到单一通道线程用于下游处理。在实施例中,这样做是因为整个映射可以被同步,因此每个通道的工作被入队,线程立即返回,使得其它通道可以接收它们的事件。发送者的原始成员节点可以通过MapListener接收事件。

[0146] 在负载平衡流的情况下,目标计算资源可以通过找到集群中成员的总数并且生成[0,总数]之间的随机数来选择。换句话说,不是利用最后使用的成员保持某个集群状态,而是可以使用随机化来实现负载平衡。键和值可以类似于广播情况。然而,MapListener可以被注册,其中过滤器被设置到应用ID、阶段ID和目标ID,其中目标ID的值是随机选择的成员ID。换句话说,在一些实施例中,只有随机选择的目标将接收事件。根据一些实施例,负载平衡流仅支持无序情况,因此仅使用异步监听者。

[0147] 在扇入流的情况下,目标计算资源可以由用户按照某个配置机制直接指定。这种用户定义的目标ID可以在高速缓存值包装器中设置,但是除此之外语义与负载平衡情况类似。扇入流支持完全排序。在这种情况下,除了使用同步映射监听者之外,还可以将通道配置为使用应用时间戳,并且在接收者中事件可以被重新排序,直到某个用户可配置的超时值。超时值例如可以在几秒的范围内,并且可以基于在较低延迟和较大几率的乱序事件之间的折衷。

[0148] 在一些实施例中,优化使用保证高速缓存键保持有序的散列。接收者然后可以使用过滤器,其中该过滤器检索从最近到过去某个时间的范围内的特定通道的所有条目。在这种情况下,可以使用连续查询映射(Continuous Query Map)。可以使用相同的超时配置定期地检查该映射。

[0149] 在扇入流的情况下,可以利用分区式高速缓存方案(PartitionedCache Scheme)的固有分区支持。高速缓存数据相关度(affinity)可以被设立为关联到由应用ID、阶段ID和所配置的分区事件属性(值)(例如,用于事件属性‘symbol’的值‘ORCL’)组成的(分区)键。这可以通过使用KeyAssociation(键关联)类来完成。高速缓存键可以保持(例如,与时间戳)相同。然而,在一些实施例中,所有键具有到之前刚刚描述的分区键的关联,从而确保分区保持共同定位。

[0150] 如果高速缓存一致性被用来将数据放置在最佳位置中,则不选择目标成员,并且因此不使用MapListener而使用EntryProcessor(条目处理器),其中过滤器被设置到应用ID、阶段ID和分区事件属性值。在这种情况下,源节点调取EntryProcessor,并且保证在数据驻留于其中的成员中执行EntryProcessor实现,因此避免了将数据拷贝到已经显式地选择的目标成员。高速缓存一致性可以使用其确定对充分利用它的内部构件来按集群成员和数据尺寸确定正确的分区数量进行优化。

[0151] 可调取的任务隐式地为它正在处理的条目获取锁。这以及数据被共同定位的事实

意味着条目可以在处理结束时在不导致另一个网络操作的情况下被删除(而不是让条目过期)。如果流被配置为无序的,那么任务可以被尽快移交到通道的多线程执行器。如果流是有序的,那么再次地,任务可以被移交到通道,但是是移交到单一通道线程。如果流是分区有序的,那么移交可以按分区发生。分区可以从键关联中确定,并且然后用于对用于执行的正确线程进行索引。换句话说,线程可以按分区着色。

[0152] 关于容错,如果当发送者发布事件时成员是关闭的,并且如果接收者正在使用MapListener,则该事件在成员返回运行时没有被接收到。解决这一点的一种方法是使用MapListener与连续查询映射的组合。在这种情况下,可以在事件被彻底处理后尽快删除事件,而不是缓慢地删除。如果成员接收到事件并且在完成事件的处理之前关闭(go down),则该事件被重新处理,这意味着该事件在它被彻底处理之前不从高速缓存中删除。

[0153] 如果分区数据迁移到不同的服务器,则可以针对分区是否被迁移的通知监听实况事件(Live Events)。在一些实施例中,这种境况被提出作为错误以让用户知道在新窗口过去之前状态为丢失。例如,在10分钟窗口中间,那么只有在下一个窗口中,状态才是有效的。

[0154] 结构视点

[0155] 在一些实施例中,经过分发流的所有事件都是可串行化的。就配置而言,以下可以被添加到通道组件。流类型:本地或集群分区、本地或集群广播、负载平衡、扇入;排序:无序、部分分区有序、完全分区有序、部分有序、完全有序;分区属性:String(串);应用时间戳:long(长整型);完全排序超时:long(长整型)。

[0156] 部署视点

[0157] 一致性高速缓存配置可以被包括作为服务器部署/配置的一部分。这可以包括用于不同分发流中的每一个分发流的不同高速缓存方案的配置。

[0158] 设计时考虑

[0159] 在一些实施例中,在集成开发环境(IDE)内,开发者可以从调色板(palette)中选择不同风格的通道:常规通道、广播通道、分区通道或扇入通道。调色板中不同的通道风格是对用户的视觉线索。它们可以实现为配置,以便允许用户改变运行时分发流而不必在IDE中创作应用。在分区通道的情况下,IDE可以通过提示用户将有序的事件属性集合用作分区标准来进行跟踪。该有序的事件属性集合可以如在通道的事件类型中所定义的那样存在。同样,任何其它通道专用配置可以相应地是可配置的。

[0160] 管理、操作和安全

[0161] 分布式通道风格以及关联到它的任何配置可以呈现在管理控制台中,作为事件处理网络图视图中的通道和通道配置的一部分。管理/监视控制台还可以提供用于可视化集群的计算资源的完整网络的机制。这是集群的部署视图。

[0162] 此外,一个方面涉及能够理解计算资源中的源节点到另一个计算资源中的目的地节点之间的运行时交互或映射。这构成了集群的连接视图。在这种情况下,不仅示出了运行时连接,而且还示出了已经经过连接的事件的数量。

[0163] 另一个有用的监视工具提供保护特定事件的能力。例如,用户可以确保事件e1＝{(p1,1),(p2,2)}已经经过机器3中的处理器2。在一些实施例中,提供了用于监护事件的机制。

[0164] 基于事件数据的绘图

[0165] 一些实施例允许在流化数据中实时识别境况，诸如威胁和机会。这些境况可以通过可视化图来识别。下面描述的是可以用来可视化数据的各种图(graphs and plots)。一些这样的图不是监视图，而更确切地是勘探图。勘探图可以是可配置的，以便允许不同境况的试试看精确锁定(try-and-see pin-pointing)。

[0166] 根据一些实施例，绘图机制接收时间系列数据作为输入。时间系列数据是随时间变化的数据。时间系列数据可以由以下类型的图来表示：线形图、散点图和雷达图。条形图和饼图可以用来表示分类数据。

[0167] 线形图是可视化时间系列数据的一种方式。X轴可以用来表示时间的推移，并且因此允许数据随着时间向前推移的自然滚动。Y轴可以用来查看因变量--感兴趣的变量--随着时间推移的变化。图11是示出根据一些实施例的线形图1100的示例的图。

[0168] 因变量可以是输出事件的属性中的任何属性。例如，因变量可以是股票事件的属性‘price(价格)’，或者是对源应用摘要和条件(Summaries and Conditions)所产生的属性‘Sum(price)>10(求和(价格)>10)’。线形图适合连续变量。在一些实施例中，对于线形图，允许选择作为数值的事件属性。在一些实施例中，不允许在Y轴中选择Interval(间隔)、DateTime(日期时间)、Boolean(布尔)、String(串)、XML等类型的属性。输出事件的第一数值属性可以被初始地选择为Y轴。用户被允许将此更改为任何其它数值属性。

[0169] 如以上所提到的，X轴可以指定时间系列。该轴可以使用转换为HH:MM:SS:毫秒(小时:分钟:秒:毫秒)格式的、输出事件到达的系统时间戳，并使用已知的滑动(slide)标准进行滑动，用于评估查询和更新实况流(Live Stream)表格输出表。这可能是在1/10秒到1/2秒的范围内。在一些实施例中，可选地，可以使用如由CQL指示的输出事件的实际时间戳或基本动作时间(element time)。在应用时间戳记的查询的情况下，时间戳表示应用时间，应用时间可以与系统时间显著不同。例如，针对实际时间的每个小时，应用时间可能推移1个计时分度(tick)。

[0170] 分析流化数据的另一个方面是理解其变量之间的相关性。相关性示出了变量对的协方差，并且范围从-1(逆强相关)，0(无相关性)到1(直接强相关)。例如，汽车的重量与其每加仑英里数(MPG)直接相关。但是，汽车的重量和其颜色之间没有相关性。为了支持这种相关性，一些实施例允许将第二条线(以不同颜色)绘制到线形图。该第二条线表示第二变量，第二变量可以是由用户选择的第二事件属性。

[0171] 每条线(由其x和y对的集合构成)被称为数据系列。绘制两条线允许用户对变量是具有直接线性度还是间接线性度进行可视化。此外，可以计算变量的相关系数并在图中呈现该相关系数。相关系数可以使用CQL函数correlate()来计算。相关系数可以使用颜色渐变来呈现，其中绿色是直接相关的、红色是间接相关的、并且灰色意味着无相关性。

[0172] 在一些实施例中，向用户提供选择附加变量(即，事件属性)以在线形图中绘制为线(系列)的机制，其中附加变量上至某个方便的最大值(例如，在5和10之间)。当用户选择输出事件的属性时，他可以选择计算出的变量，诸如分类属性的计数结果。然而，由于相关以成对方式进行，并且相关可能是繁重的(taxing)，所以一些实施例仅允许一次使两个变量相关。如果图具有多于两个变量，则用户可以通知哪两个变量应该被用来计算相关系数。

[0173] 在一些实施例中，可选地，可以与相关系数的计算一起计算结果的置信度。这让用户知道有多大可能随机样本将产生相同的结果。

[0174] 相关性是与方差和协方差相联系的。在一些实施例中，可选地，图的可见特征示出了图内表示的分布是否是正态分布。

[0175] 在一些实施例中，可以自动确定前N个相关的成对变量。

[0176] 当时间维度不太重要时，则可以通过散点图中的表示来更好地理解数据。图12是示出根据一些实施例的散点图1200的示例的图。在这种情况下，用户可以向X轴和Y轴两者分配不同的事件属性。然而，在一些实施例中，两者都被约束为数值变量。在一些实施例中，默认情况下，可以从输出事件(类型)中选择作为数值的前两个属性。X轴表示解释变量，并且Y轴表示响应变量。因此，指示响应的属性或计算出的字段(诸如totalX或sumOfY或outcomeZ)可以是用于自动分配到Y轴的良好候选。

[0177] 在时间系列的情况下，新值在图的右侧进入并且旧值在左侧退出。但是，这种行为不会变换到散点图，因为新点可能出现在任何地方。因此，在一些实施例中，新点被赋予一些视觉线索。例如，新点可以最初以红色或蓝色绘制，不像其它现有点，随着新点到达逐渐淘汰(phasing out)。

[0178] 在图中的最旧点开始被自动移除之前，在一个时间处图中可以维持各种数量的点。存在执行这种移除的若干种方法。一种技术保持尽可能多的点，只要这些点不使可视化变差(或混乱)。另一种技术保持理解数据所需的最少事件集合。

[0179] 在一些实施例中，可以在散点图中绘制以某种形式限制点或对点进行塑形的线。一种技术在Y轴中的所有值之上绘制一条线表示最大值，并且在Y轴中的所有值之下绘制另一条线表示最小值。另一种技术绘制涵盖所有点的多边形。这向用户给出了关于值在哪里的受限视图。

[0180] 另一种技术绘制经平滑的曲线拟合器(即，局部加权线性回归(lowess))。图13是示出根据一些实施例的在其中已绘制经平滑的曲线拟合器的散点图1300的示例的图。这种技术可用于预测性在线处理。如线形图的情况中，可以提供所讨论的两个变量的相关系数。经平滑的线和回归拟合也可以指示线性度。

[0181] 散点图很好地适用于支持被表示为点的尺寸的第三维度的可视化。图14是示出根据一些实施例的、在其中点具有不同尺寸的散点图1400的示例的图。在这种情况下，用户可以向‘size(尺寸)’维度分配第三事件属性。机制可以以避免使图混乱的方式对尺寸进行缩放。

[0182] 除了X轴被绘制为表示时间周期(诸如24小时周期)的圆形使雷达图看起来像雷达之外，雷达图与线形图是类似的。图15是示出根据一些实施例的雷达图1500的示例的图。雷达图对于发现特定境况是否是循环的是有用的，并且因此是处理时间系列的有用工具。例如，这种图可以用来确定股票的价格是否通常在工作日的开始或结束时高，或者用来确定在星期五销售的机票的数量是否高于在该周中的任何其它日子。

[0183] 雷达图的响应变量也可以是数值的。找到X轴的正确尺度是雷达图的考虑。如果窗口范围被定义，那么它可以用作雷达图的默认循环。否则，循环可以从毫秒到小时变化。在图中附加的线可以表示不同响应变量。如线形图一样。

[0184] 数值变量不是可以在图中被可视化的唯一变量。分类变量(无论它们是定类的(例如，男性、女性)还是定序的(例如，高、中、低))也可以在图中进行可视化。分类变量通常被分析为频率，诸如例如欺诈的数量(因素是“欺诈”、“没有欺诈”)、黄金客户与常规客户的比

率、上星期被观看的前5个电影等。这种变量通常在条形图和饼图中进行可视化。在某些情况(例如前n个)下,可能存在计算频率所涉及的高CPU/存储器消耗。因此,在一些实施例中,这种计算不在后台中执行。在一些实施例中,执行以下操作:

[0185] 1.选择条形图(或饼图)

[0186] 2.将串类型的事件属性分配给X轴

[0187] 3.将计数结果(或任何其它数值属性)分配给Y轴

[0188] 计数结果可以按照用户的定义(例如使用窗口范围)来更新,因此不需要从条形图中显式地清除值。也可以选择饼图。可以交由用户来确保跨类别的总量为100%。

[0189] 线形图适用于涉及寻找时间系列中的一般趋势和数值变量的场景。散点图适用于涉及寻找相关性和数值变量的场景。雷达图适用于涉及寻找循环和数值变量的场景。条形图适用于涉及计数频率和分类变量的场景。

[0190] 由于某些类型的图可以更好地适用于特定类型的数据(例如,分类数据或数值数据)和分析,因此它们同样可以被不同地调整尺寸和更新。聚焦时间系列的线形图包含最后t时间的事件,并且可以随着时间的系统(CPU)推移而移动(更新)。换句话说,即使没有事件到达,该图仍然将从右到左更新并且移动任何先前绘制的事件。默认情况下,线形图可以被配置为包含最近300秒的事件。然而,用户可以改变该参数t(例如,300)以及将时间粒度从毫秒改变为分钟。如果时间窗口被定义,则时间窗口的尺寸可以用作线形图的X轴的默认尺寸(尺度)。

[0191] 雷达图类似于线形图,雷达图中还添加了周期间隔的配置。例如,配置可以指示以每个60秒的循环(周期)显示300秒。散点图和条形图不面向时间,并且因此在一些实施例中,仅当事件到达时才更新,而不一定随时间推移而更新。可以使用功率分析来如下确定X轴的尺寸:考虑散点图被用于辨认(spot)相关性,并且考虑通常如果在变量之间存在80%协方差则相关性被认为强,然后使用95%的置信度水平(即,存在5%的几率随机点表示重要模式),给出:

[0192] pwr.r.test(r＝.20,power0.95,sig.level.05,alternative＝‘greater’)
265.8005

[0193] 即,散点图应该包括至少265个点。如果将此放松到75%的相关性,具有10%的误差限度,则:

[0194] pwr.r.test(r＝.25,power＝0.90,sig.level＝.10,alternative＝‘greater′)
＝＝103.1175

[0195] 两种选项都是可能的。在一些实施例中,用户可以将尺寸定制为任何任意的数。

[0196] 条形图与散点图类似地起作用,并且在一些实施例中,条形图仅在新事件到达时才更新。类别的数量可以使用过程来决定。可以假定在10到20范围内的默认值。用户可以根据需要进一步定制。在一些实施例中,可以选择前n个类别。在一些实施例中,这可以被编码到查询中。

[0197] 表格输出也可以以类似于散点图来调整尺寸。当绘制具有多个变量的点时出现的一个问题是尺度的问题。具体而言,这在使用具有多个系列的线形图时是非常明显的。这可以在两步中完成:

[0198] 1.将数据中心化使得它更接近平均值;这带来异常值。

[0199] 2.通过将数据除以其标准偏差来规格化比率。

[0200] 公式为：

[0201] x’＝(x-mean)/standard-deviation(x′＝(x-平均值)/标准偏差)。

[0202] 由于数据的流化本质,平均值和标准偏差在一些实施例中是连续更新的。

[0203] 集群无监管学习

[0204] 集群将变量(特征)更靠近的事件分组在一起。这种分组允许用户识别由于某种未知关系而走到一起的事件。例如,在股票市场中,衍生品的集群一起上行或下行是常见的。如果有来自IBM的积极营收报告,那么很可能随后有来自例如Oracle、Microsoft和SAP的积极结果。

[0205] 集群的一个吸引力是它是无监管的;即,用户不需要识别响应变量或提供训练数据。这个框架与流化数据很好地相适应。在一些实施例中,使用以下算法：

[0206] 如果每个事件i包含指定为xij的j个变量(例如,价格、体量),并且如果目标是将这些事件集群到k个集群中使得每个k集群由其质心ckj定义(该质心ckj是对于作为该集群的一部分的所有事件的j个变量的平均值的向量),那么对于插入到处理窗口中的每个事件,该事件的k集群可以如下确定：

[0207] 1.如果它是第一个事件,则将它分配给最小的k集群(例如集群0)。

[0208] 2.对于每个(新)输入事件i,如下计算它到k个集群的质心的(平方)距离：

[0209] 2a.distik＝SUMj((xij-ckj)^2)

[0210] 2b.将事件i分配给具有最小distik的集群k

[0211] 3.为所选择的集群k重新计算质心：

[0212] 3a.对于所有j,ckj’＝(ckj+xij)/|ck|+1

[0213] 对于流化数据,处置离开处理窗口的事件存在额外的复杂性：

[0214] 1.对于来自集群k的被清除的每个(旧)事件i,重新计算其集群的质心：

[0215] 1a.对于所有j,ckj’＝(ckj-xij)/|ck|-1

[0216] 随着对于每个事件(新的和旧的)发生的质心改变,存在将现有事件(点)重新定位到在其中距离已经变为最小距离的新集群的潜在可能。因此,在一些实施例中,该过程为所有被分配的点重新计算距离,直到不再发生重新分配。但是,这可能是繁重的步骤。如果处理窗口小并且节奏快,则移除和添加新点将具有同样效果,并且缓慢地聚合到可以达到的最佳的局部最优。因为这种聚合未得到保证,所以可以提供当事件到达时启用/禁用重新计算的选项。

[0217] 另一个问题是缩放的问题;因为距离被计算为特征之间的欧几里得(Euclidian)距离,所以在一些实施例中特征被相等地缩放。否则,在距离计算期间,单个特征可能淹没(overwhelm)其它特征。

[0218] 集群算法可以被表示为以下形式的CQL汇总函数：

[0219] cluster(max-clusters:int,scale:Boolean):int

[0220] cluster(max-clusters:int,scale:Boolean,key-property:String):List<String,Integer>

[0221] 参数max-clusters定义集群的总数(即,k),并且一旦开始查询就不改变。后者签名使得能够重新计算集群分配,并且将事件键的列表返回到集群分配。

[0222] 就可视化而言，集群数据可以被叠加在先前定义的所支持的图中的任何图的上面。换句话说，如果用户选择散点图，那么可以包括将点与k个集群之一相关联的颜色和/或形状作为每个点的一部分。图19是示出根据一些实施例的叠加在散点图上的表示集群的形状1900的示例的图。形状包括属于该形状表示的集群的点。

[0223] HBASE

[0224] 如以上所讨论的，TF-IDF值的计算涉及为“反向文档频率”计算访问历史文档，这使得TF-IDF流处理场景是用于与例如HBase的索引层一起使用的良好候选。HBase适用于‘Big Data(大数据)’存储，其中不需要关系数据库管理系统(RDBMS)的功能。HBase是一种类型的‘NOSQL’数据库。HBase不支持结构化查询语言(SQL)作为访问数据的主要方法。相反，HBase提供JAVA应用编程接口(API)来检索数据。

[0225] HBase数据存储库中的每一行具有一个键。HBase数据存储库中的所有列属于特定的列族。每一列族包括一个或多个限定符。因此，为了从HBase数据存储库中检索该数据，使用了行键(row key)、列族(column family)和列限定符(column qualifier)的组合。在HBase数据存储库中，每个表具有行键，这与关系数据库中的每个表是如何具有行键的类似。

[0226] 图1是示出根据一些实施例的HBase数据存储库中的表100的示例的图。表100包括列102-列110。列102存储行键。列104存储姓名。列106存储性别。列108存储数据库课程的成绩。列110存储算法课程的成绩。表100的列限定符表示列102-列110的名称：姓名、性别、数据库和算法。

[0227] 表100涉及两个列族112和114。列族112包括列104和列106。在这个示例中，列族112被命名为“基本信息”。列族114包括列108和列110。在这个示例中，列族114被命名为“课程成绩”。

[0228] HBase列限定符的设想类似于NoSqlDB中的次键(minor key)的概念。例如，在NoSqlDB中，记录的主键(major key)可以是人的姓名。次键可以是将为该人存储的不同信息片段。例如，给定主键“/Bob/Smith”，对应的次键可以包括“出生日期”、“城市”和“州”。对于另一个示例，给定主键“/John/Snow”，对应的次键可以类似地包括“出生日期”、“城市”和“州”。

[0229] 包含在HBase数据存储库中的信息不使用任何查询语言来检索。HBase背后的目标是高效地存储大量数据，而不执行任何复杂的数据检索操作。如以上提到的，在HBase中，数据使用JAVA API来检索。以下代码片段给出了可以如何在HBase中存储和检索数据的思想：

[0230] HBaseConfiguration config＝new HBaseConfiguration()；

[0231] batchUpdate.put(″myColumnFamily:columnQualifier1″,

[0232] ″columnQualifier1value!″.getBytes())；

[0233] Cell cell table.get(″myRow″,″myColumnFamily:columnQualifier1″)；

[0234] String valueStr＝new String(eell.getValue())；

[0235] HBase可以用于存储各种应用的元数据信息。例如，公司可以将与各种销售相关联的客户信息存储在HBase数据库中。在这种情况下，HBase数据库可以使用使得能够编写使用HBase作为外部数据源的CQL查询的HBase盒带(cartridge)。

[0236] 根据一些实施例，使用CQL处理器，利用包含在HBase数据存储库中的上下文数据

来对源事件进行丰富。HBase数据存储库通过抽象形式被引用。

[0237] 根据一些实施例,事件处理网络(event processing network)组件被创建以表示HBase数据存储库。HBase数据存储库事件处理网络组件类似于事件处理网络<table>组件,并且在CQL处理器中被用作外部关系源。HBase数据存储库事件处理网络组件使用事件类型进行类型化。HBase数据库通过其自己的机制启动,并且是可访问的。HBase数据库不需要由诸如Oracle事件处理器(OEP)的事件处理器直接管理。

[0238] 根据一些实施例,HBase数据存储库事件处理网络组件被提供为数据盒带。HBase数据盒带提供具有以下属性的<store>事件处理网络组件:事件处理网络组件的id、存储库位置(以HBase数据库服务器的domain(域):端口的形式的位置)、事件类型(由CQL处理器看到的存储库的模式(schema))、表名称(HBase表的名称)。

[0239] 根据一些实施例,该事件处理网络组件具有相关的<column-mappings>组件,以便指定从CQL事件特性到HBase列族/限定符的映射。该组件在类似于JAVA数据库连接(JDBC)盒带配置的HBase盒带配置文件中声明。该组件具有以下属性:名称(要为其声明映射的<store>事件处理网络组件的id)、rowkey(HBase表的行键)、cql-attribute(在CQL查询中使用的CQL列名)、hbase-family(HBase列族)、以及hbase-qualifer(HBase列限定符)。根据一些实施例,在CQL列是java.util.map的情况下,用户仅指定'hbase-family'。根据一些实施例,在CQL列是基本数据类型的情况下,用户指定'hbase-family'和'hbase-qualifier'两者。

[0240] 根据一些实施例,<hbase:store>组件使用'table-source'要素被链接到CQL处理器,如以下示例中所示:

[0241] <hbase:store id="User"tablename="User"event-type="UserEvent"store-

[0242] location="localhost:5000"row-key="username">

[0243] </hbase:store>

[0244] <wlevs:processor id="P1">

[0245] <wlevs:table-source ref="User"/>

[0246] </wlevs:processor>

[0247] 根据一些实施例,用于该<hbase:store>组件的列映射在事件处理器(例如,OEP)的HBase配置文件中指定,如以下示例中所示:

[0248] <hbase:column-mappings>

[0249] <store>User</store>

[0250] <mapping cql-attribute="address"hbase-family="address"/>

[0251] <mapping cql-attribute="firstname"hbase-family="data"hbase-qualifier="firstname"/>

[0252] <mapping cql-attribute="lastname"hbase-family="data"hbase-qualifier="lastname"/>

[0253] <mapping cql-attribute="email"hbase-family="data"hbase-qualifier="email"/>

[0254] <mapping cql-attribute="role"hbase-family="data"hbase-qualifier="role"/>

[0255] </hbase:column-mappings>

[0256] 根据一些实施例,UserEvent(用户事件)类具有以下字段:

[0257] String userName;

[0258] java.util.Map address;

[0259] String first name;

[0260] String lastname;

[0261] String email;

[0262] String role;

[0263] 在以上示例中,CQL列“address(地址)”是映射,因为它将保持来自‘address’列族的所有列限定符。CQL列“firstname(名)”、“lastname(姓)”、“email(电子邮件)”和“role(角色)”保持基本数据类型。这些是来自“data(数据)”列族的具体列限定符。来自事件类型的‘userName(用户名称)’字段是行键,并且因此它不具有任何到HBase列族或限定符的映射。

[0264] 根据一些实施例,HBase模式本质上可以是动态的,并且可以在创建HBase表之后的任何时间点添加附加的列族和/或列限定符。因此,事件处理器(例如,OEP)允许用户将事件字段检索为包含所有动态添加的列限定符的映射。在这种情况下,用户将java.util.Map声明为JAVA事件类型中的事件字段之一。因此,以上的‘UserEvent’事件类型具有名称为“address”的java.util.Map字段。如果盒带不支持动态添加的列族,那么如果事件处理应用需要使用新添加的列族,则事件类型可以被修改。

[0265] 根据一些实施例,HBase数据库作为集群执行。在这种场景中,主节点的主机名在以上配置中提供。

[0266] 根据一些实施例,在HBase源的配置期间,从用户接收事件类型储存库中存在的事件类型的名称。当从用户接收“column-mappings”时,用户接口可以将该具体事件类型中的列(字段)名称提供作为cql-column。因此,可以在用户接口级别消除错误的输入。可用的HBase列族和其中的列限定符的详细信息可以提供给用户以从中选择。解析器验证在盒带中执行。

[0267] 客户销售示例

[0268] 以下示例识别大的销售和相关联的客户。销售数据在传入流中获得并且客户信息从HBase数据库中获得。

[0269] <?xml version＝"1.0"encoding＝"UTF-8"?>

[0270] <beans xmlns＝"http://www.springframework.org/schema/beans"

[0271] xmlns:xsi＝"http://www.w3.org/2001/XMLSchema-instance"

[0272] xmlns:osgi＝"http://www.springframework.org/schema/osgi"

[0273] xmlns:wlevs＝"http://www.bea.com/ns/wlevs/spring"

[0274] xmlns:hbase＝"http://www.oracle.com/ns/ocep/hbase"

[0275] xmlns:hadoop＝"http://www.oracle.com/ns/ocep/hadoop"

[0276] xsi:schemaLocation＝"

[0277] http://www.springframework.org/schema/beans

[0278] http://www.springframework.org/schema/beans/spring-beans.xsd

[0279] http://www.springframework.org/schema/osgi

[0280] http://www.springframework.org/schema/osgi/spring-osgi.xsd

[0281] http://www.bea.com/ns/wlevx/spring

[0282] http://www.bea.com/ns/wlevs/spring/spring-wlevs-v11_1_1_6.xsd">

[0283] <wlevs:event-type-repository>

[0284] <wlevs:event-type type-name＝"UserEvent">

[0285] <wlevs:class>

[0286] com.bea.wlevs.example.UserEvent

[0287] </wlevs:class

[0288] </wlevs:event-type>

[0289] <wlevs:event-type type-name＝"SalesEvent">

[0290] <wlevs:class>com.bea.wlevs.example.SalesEvent</wlevs:class>

[0291] </wlevs:event-type>

[0292] </wlevs:event-type-repository>

[0293] <!--Assemble event processing network(event processing network)-->

[0294] <wlevs:adapter id＝"A1"class＝"com.bea.wlevs.example.SalesAdapter">

[0295] <wlevs:listener ref＝"S1"/>

[0296] </wlevs:adapter>

[0297] <wlevs:channel id＝"S1"event-type＝"SalesEvent">

[0298] <wlevs:listener ref＝"P1"/>

[0299] </wlevs:channel>

[0300] <hbase:store id"User"event-type"UserEvent"

[0301] store-locations＝"localhost:5000"table-name＝"User">

[0302] </hbase:store/>

[0303] <wlevs:processor id＝"P1">

[0304] <wlevs:table-source ref＝"User"/>

[0305] </wlevs:processor>

[0306] <wlevs:channel id＝"S2"advertise＝"true"event-type＝"SalesEvent">

[0307] <wlevs:listener ref＝"bean"/>

[0308] <wlevs:source fef＝"P1"/>

[0309] </wlevs:channel>

[0310] <!--Create business object-->

[0311] <bean id＝"bean"class＝"com.bea.wlevs.example.OutputBean"/>

[0312] </beans>

[0313] 在HBase盒带配置文件中指定以下列映射：

[0314] <hbase:columnmappings>

[0315] <name>User</name>

[0316] <rowkey>userName</name>

[0317] <mapping cql-column＝"firstname"hbase-family＝"data"

[0318] hbase-qualifier＝”firstname”/>

[0319] <mapping cql-column＝”lastname”hbase-family＝”data”

[0320] hbase-qualifier＝”lastname”/>

[0321] <mapping cql-column＝”email”hbase-family＝”data”

[0322] hbase-qualifier”email”/>

[0323] <mapping cql-column”role”hbase-family”data”

[0324] hbase-qualifier＝”role”/>

[0325] <mapping cql-column＝”address”hbase-family＝”address”/>

[0326] </hbase:columnmappings>

[0327] 以上示例中的“User”HBase表具有以下模式：

[0328] 行键:username

[0329] 列族:data,address

[0330] ‘data’列族的列限定符:firstname,lastname,email,role

[0331] ‘address’列族的列限定符:country,state,city,street

[0332] 处理器运行将输入流与该表连接的以下CQL查询：

[0333] select user.firstname,user.lastname,user.email,user.role,user.address.get(“city”),price from S1[now],User as user

[0334] where S 1.username＝user.username and price>10000

[0335] 这里,“address”列族被声明为

[0336] “com.bea.wlevs.example.UserEvent”类中的“java.util.Map”字段。因此,使用“user.address.get(‘<column-qualifer-name>’)”以便从该列族检索具体列限定符的值。

[0337] 用于OPENTSBD监视系统的HBASE基础

[0338] 一些实施例可以涉及映射的映射:行键、列族、列限定符和多个版本。一行可以包含多个列族。然而,每个族被一起对待(例如,经压缩/未压缩)。列族可以包含多个列限定符。列限定符可以被动态添加或移除。每个单体(cell)具有多个版本,默认情况下检索最近的版本。与列族相关联的策略确定保留多少版本以及它们何时被清除。

[0339] 一些实施例是无模式的和无类型的。API可以包括get(获取)、put(放置)、delete(删除)、scan(扫描)和increment(递增)。使用经过滤的扫描来执行非键上的查询,经过滤的扫描支持丰富的过滤语言。

[0340] 用于OPENTSBD监视系统的模式

[0341] 一些实施例可以涉及如下的UID表：

[0342] ROW COLUMN+CELL

[0343] \x00\x00\x01 column＝name:metrics,value＝mysql.bytes_sent

[0344] \x00\x00\x02 column＝name:metrics,value＝mysql.bytes_received

[0345] mysql.bytes_received column＝id:metrics,value＝\x00\x00\x02

[0346] mysql.bytes_sent column＝id:metrics,value＝\x00\x00\x01

[0347] 一些实施例会涉及如下的度量表：

[0348] 行(键)：

[0349]

度量 UID (3 字节)	部分时间戳 (4 字节)	标签 1 名称 UID (3 字节)	标签 1 值 UID (3 字节)	...

[0350] 列+单体：

[0351]

较低时间戳 (12 位)	掩码 (4 位)

[0352] 对OPENTSBD监视系统的查询

[0353] 根据一些实施例，从CQL的角度，外部关系映射到HBase表源。在配置HBase表源时，可以从用户接收关于外部关系的哪个特性映射到HBase表中的哪个列族或哪个column family.column qualifier(列族.列限定符)的详细信息。

[0354] 根据一些实施例，可以从度量名称找到UID。CQL查询可以如下：

[0355] select metrics from S[now],UID_TABLE where UID_TABLE.rowkey＝S.uid

[0356] 这里，“metrics(度量)”是名为UID_TABLE的外部关系的特性，该特性映射到名为UID_TABLE的HBase表的“name”列族中的“metrics”列限定符。此外，“rowkey”是外部关系的映射到HBase表的行键的另一个特性。

[0357] 根据一些实施例，以cpu开头的所有度量名称可以使用诸如以下的查询来找到：

[0358] select metrics from S[now],UID_TABLE where rowkey like“^cpu”

[0359] 这里，指定了表示要匹配的正则表达式的串。该串可以是流S的特性，但并不需要是流S的特性。可以使用HBase API来支持这种基于正则表达式的查询。在HBase中，可以通过指定包含性的开头和排除性的结尾对行进行扫描。

[0360] 根据一些实施例，使用了类似的技术，诸如：

[0361] SELECT name:metrics FROM UID_TABLE,S WHERE rowkey>＝10 AND rowkey<

[0362] 一些实施例利用

[0363] htrp://hbase.apache.org/apidocs/org/apache/hadoop/hbase/filter/RegexStringComparator.html

[0364] 根据一些实施例，为外部关系指定了谓词支持能力。如果给定的谓词落入所支持的谓词能力列表中，则它在外部关系上执行。否则，所有数据都被带入到存储器中，并且然后CQL引擎应用该谓词。

[0365] 根据一些实施例，可以找到由主机过滤的服务延迟度量的量度(measures)。在这种情况下，CQL是：

[0366] select measures from S[now]as p,metrics_table where rowkey＝key-

[0367] encoding(p.servicelatency-UID,p.ELEMENT_TIME)and host＝‘myhost’

[0368] 在以上示例中，“rowkey”、“host(主机)”和“measures”是映射到HBase表源的外部关系的列，并且key-encoding(键编码)是用户定义的函数。

[0369] 根据一些实施例，可以找到其服务延迟高于1000(毫秒)的主机区域。列限定符可以被动态添加到现有行。例如，可以添加包含主机被部署的区域的新的列限定符“region

(区域)”。如果验证特性“host”的元数据不可用,则可以使用以下方法。在hbase:column-mappings中,用户可以指定:

[0370] <mapping cql-attnbute=”cl”hbase-family=”cfl”/>

[0371] 这里,“cfl”是列族名称,并且c1是类型j ava.util.Map。用户可以访问“cfl”中的限定符为c1.get(“qualifier-name”)。因此,CQL查询可以如下:

[0372] select info.get(“region”)from S[now]as p,metrics_table where rowkey=key-

[0373] encoding(p.serviceLatencyUID)and measures>p.threshold

[0374] 这里,“info”是外部关系的特性的名称,其中该外部关系映射到“region”限定符被动态添加到的列族。

[0375] 量度可以具有多个版本。根据一些实施例,如果某个应用时间戳比最近的版本旧,则获得先前的版本。根据一些实施例,最近的版本被使用。可能存在面向事务的用例,诸如:

[0376] SELECT product:price FROM PRODUCT_TABLE,TRANSACTION_STREAM[now]

[0377] AS S

[0378] WHERE row-key=S.productld

[0379] 换句话说,虽然价格可能已改变,但是在交易被发出时所看到的价格仍将考虑它的应用时间戳来兑现(honor)。

[0380] 硬件概述

[0381] 图16绘出了用于实现实施例之一的分布式系统1600的简化图。在所示的实施例中,分布式系统1600包括一个或多个客户端计算设备1602、1604、1606和1608,这些设备被配置为通过(一个或多个)网络1610执行和操作诸如web浏览器、专有客户端(例如OracleForms)等之类的客户端应用。服务器1612可以经由网络1610与远程客户端计算设备1602、1604、1606和1608在通信上耦合。

[0382] 在各种实施例中,服务器1612可以适于运行由系统的组件中的一个或多个组件提供的一个或多个服务或软件应用。在一些实施例中,这些服务可以作为基于web或云的服务或者在软件即服务(Software as a Service,SaaS)模型下提供给客户端计算设备1602、1604、1606和/或1608的用户。操作客户端计算设备1602、1604、1606和/或1608的用户可以继而利用一个或多个客户端应用与服务器1612交互,以利用由这些组件提供的服务。

[0383] 在图中所绘出的配置中,系统1600的软件组件1618、1620和1622被示为在服务器1612上实现。在其它实施例中,系统1600的组件中的一个或多个组件和/或由这些组件提供的服务也可以由客户端计算设备1602、1604、1606和/或1608中的一个或多个来实现。操作客户端计算设备的用户然后可以利用一个或多个客户端应用来使用由这些组件提供的服务。这些组件可以用硬件、固件、软件或其组合来实现。应当理解,各种不同的系统配置是可能的,这些配置可以与分布式系统1600不同。因此,在图中示出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,并不是要进行限制。

[0384] 客户端计算设备1602、1604、1606和/或1608可以是便携式手持设备(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),这些设备运行诸如Microsoft Windows Mobile® 的软件

和/或诸如iOS、Windows Phone、Android、BlackBerry 17、Palm OS等各种移动操作系统，并且启用了互联网、电子邮件、短消息服务(SMS)、Blackberry®或其它通信协议。客户端计算设备可以是通用个人计算机，例如包括运行各种版本的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各种可商用UNIX®或类UNIX操作系统中任意操作系统的工作站计算机，这些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。可替代地或附加地，客户端计算设备1602、1604、1606和1608可以是能够通过(一个或多个)网络1610通信的任何其它电子设备，诸如瘦客户端计算机、启用了互联网的游戏系统(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。

[0385] 虽然示例性分布式系统1600被示为具有四个客户端计算设备，但是任何数量的客户端计算设备都可以被支持。诸如具有传感器的设备等的其它设备可以与服务器1612交互。

[0386] 在分布式系统1600中的(一个或多个)网络1610可以是本领域技术人员熟悉的、可以支持使用各种可商用协议中的任何协议的数据通信的任何类型的网络，这些协议包括但不限于TCP/IP(传输控制协议/互联网协议)、SNA(系统网络体系结构)、IPX(互联网数据包交换)、AppleTalk等。仅仅作为示例，(一个或多个)网络1610可以是局域网(LAN)，诸如基于以太网、令牌环网等的局域网。(一个或多个)网络1610可以是广域网和互联网。它可以包括虚拟网络，包括但不限于虚拟专用网(VPN)、内联网、外联网、公共交换电话网(PSTN)、红外网络、无线网络(例如，在电气和电子协会(IEEE)802.11协议套件、Bluetooth®和/或任何其它无线协议中的任何协议下操作的网络)和/或这些网络和/或其它网络的任意组合。

[0387] 服务器1612可以包括一个或多个通用计算机、专用服务器计算机(包括例如PC(个人计算机)服务器、UNIX®服务器、中档服务器、大型计算机、机架式服务器等)、服务器群组、服务器集群或者任何其它适当的布置和/或组合。在各种实施例中，服务器1612可以适于运行上述公开中描述的一个或多个服务或软件应用。例如，服务器1612可以对应于用于执行上述根据本公开的实施例的处理的服务器。

[0388] 服务器1612可以运行操作系统，操作系统包括任何以上讨论的操作系统，以及任何可商用的服务器操作系统。服务器1612也可以运行各种附加的服务器应用和/或中间层应用中的任何应用，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(国际商业机器公司)等的那些可商用数据库服务器。

[0389] 在一些实现中，服务器1612可以包括分析和整合从客户端计算设备1602、1604、1606和1608的用户接收到的数据馈送和/或事件更新的一个或多个应用。作为示例，数据馈送和/或事件更新可以包括但不限于Twitter®(推特)馈送、Facebook®(脸书)更新或者从一个或多个第三方信息源接收到的实时更新和连续数据流，这些连续数据流可以包括

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1612也可以包括经由客户端计算设备1602、1604、1606和1608中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。

[0390] 分布式系统1600也可以包括一个或多个数据库1614和数据库1616。数据库1614和数据库1616可以驻留在各种位置中。作为示例,数据库1614和数据库1616中的一个或多个可以驻留在服务器1612本地(和/或驻留在服务器1612中)的非临时性存储介质上。可替代地,数据库1614和数据库1616可以远离服务器1612,并且经由基于网络的连接或专用的连接与服务器1612通信。在一组实施例中,数据库1614和数据库1616可以驻留在存储区域网络(SAN)中。类似地,用于执行属于服务器1612的功能的任何必要的文件可以根据需要本地地存储在服务器1612上和/或远程地存储。在一组实施例,数据库1614和数据库1616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。

[0391] 图17是根据本公开的实施例的系统环境1700的一个或多个组件的简化框图,通过该系统环境1700,由实施例系统的一个或多个组件提供的服务可以被提供为云服务。在所示的实施例中,系统环境1700包括可由用户使用以与提供云服务的云基础设施系统1702交互的一个或多个客户端计算设备1704、1706和1708。客户端计算设备可以被配置为操作可以被客户端计算设备的用户使用来与云基础设施系统1702交互以使用由云基础设施系统1702提供的服务的诸如web浏览器、专有客户端应用(例如,Oracle® Forms)或一些其它应用之类的客户端应用。

[0392] 应当理解,在该图中绘出的云基础设施系统1702可以具有除了所绘出的那些之外的其它组件。此外,在该图中示出的实施例只是可以结合本发明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1702可以具有比在该图中示出的组件更多或更少的组件、可以合并两个或更多个组件、或者可以具有不同的组件配置或布置。

[0393] 客户端计算设备1704、1706和1708可以是类似于上面针对1602、1604、1606和1608所描述的那些设备的设备。

[0394] 虽然示例性系统环境1700被示为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都可以被支持。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1702交互。

[0395] (一个或多个)网络1710可以便于客户端1704、1706和1708与云基础设施系统1702之间的通信和数据交换。每个网络可以是本领域技术人员熟悉的、可以支持利用任何各种可商用协议中的任何协议的数据通信的任何类型的网络,这些协议包括上面针对(一个或多个)网络1610所描述的的那些协议。

[0396] 云基础设施系统1702可以包括一个或多个计算机和/或服务器,这些计算机和/或服务器可以包括上面针对服务器1612所描述的那些计算机和/或服务器。

[0397] 在某些实施例中,由云基础设施系统提供的服务可以包括可以让云基础设施系统的用户按需使用的许多服务,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、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地缩放，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，来自云服务提供商系统的、经由诸如互联网的通信网络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内部服务器和系统不同。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该应用。

[0398] 在一些示例中，在计算机网络云基础设施中的服务可以包括对由云供应商提供给用户的或者以本领域中已知的其它方式提供的存储空间、托管的数据库、托管的web服务器、软件应用或其它服务的受保护的计算机网络访问。例如，服务可以包括通过互联网对云上远程存储空间的受密码保护的访问。作为另一个示例，服务可以包括用于被联网的开发人员私人使用的基于web服务的托管的关系数据库和脚本语言中间件引擎。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。

[0399] 在某些实施例中，云基础设施系统1702可以包括以自助服务、基于订阅、弹性可缩放、可靠、高可用性并且安全的方式交付给客户的一套应用、中间件和数据库服务供应物(offerings)。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。

[0400] 在某些实施例中，云基础设施系统1702可以适于自动供应、管理和跟踪客户对由云基础设施系统1702提供的服务的订阅。云基础设施系统1702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，在公共云模型中云基础设施系统1702由销售云服务的组织拥有(例如，被Oracle拥有)并且使得服务对一般公众和不同行业的企业可用。作为另一个示例，服务可以在私有云模型下提供，在私有云模型中云基础设施系统1702只为单个组织运营并且可以为该组织内的一个或多个实体提供服务。云服务也可以在社区云模型下提供，在社区云模型中云基础设施系统1702和由云基础设施系统1702提供的服务被相关社区中的若干个组织共享。云服务也可以在混合云模型下提供，混合云模型是两种或更多种不同模型的组合。

[0401] 在一些实施例中，由云基础设施系统1702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1702提供的一个或多个服务。云基础设施系统1702然后执行处理，以提供客户的订阅订单中的服务。

[0402] 在一些实施例中，由云基础设施系统1702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在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一套按需应用的能力。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。

[0403] 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在PaaS类别下的云服务。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、以及能够构建利用由平台提供的共享服务的新应用的服务。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。客户可以获得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数据云服务(DBCS)以及其它服务。

[0404] 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 Fusion Middleware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,该共享服务部署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提供数据库即服务。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,以及Java云服务可以为客户提供部署Java应用的平台。

[0405] 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。基础设施服务便于底层计算资源(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。

[0406] 在某些实施例中,云基础设施系统1702还可以包括基础设施资源1730,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1730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)的预先集成和优化组合。

[0407] 在一些实施例中,云基础设施系统1702中的资源可以由多个用户共享并且按需动态地重新分配。此外,资源可以分配给在不同时区中的用户。例如,云基础设施系统1730可以使第一时区内的第一用户集合能够针对指定的小时数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配给位于不同时区中的另一用户集合,从而最大化资源的利用。

[0408] 在某些实施例中,可以提供由云基础设施系统1702的不同组件或模块以及由云基础设施系统1702提供的服务共享的若干内部共享服务1732。这些内部共享服务可以包括但不限于安全性和身份服务、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、高可用性的备份和恢复服务、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。

[0409] 在某些实施例中,云基础设施系统1702可以提供对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1702接收到的客户的订阅的能力等。

[0410] 在一个实施例中,如在图中所绘出的,云管理功能可以由诸如订单管理模块1720、订单编排模块1722、订单供应模块1724、订单管理和监视模块1726以及身份管理模块1728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/或服务器或者利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、服务器群组、服务器集群或任何其它适当的布置和/或组合。

[0411] 在示例性操作1734中,使用客户端设备(诸如客户端设备1704、1706或1708)的客户可以通过请求由云基础设施系统1702提供的一个或多个服务并且对由云基础设施系统

1702提供的一个或多个服务的订阅下订单来与云基础设施系统1702交互。在某些实施例中，客户可以访问云用户接口(UI)、云UI 1712、云UI 1714和/或云UI1716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1702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1702提供的一个或多个服务的信息。

[0412] 在客户下订单之后，订单信息经由云UI 1712、1714和/或1716被接收。

[0413] 在操作1736处，订单被存储在订单数据库1718中。订单数据库1718可以是由云基础设施系统1718操作和结合其它系统元件操作的若干个数据库之一。

[0414] 在操作1738处，订单信息被转发到订单管理模块1720。在一些情况下，订单管理模块1720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，并且在通过验证时接纳(book)订单。

[0415] 在操作1740处，关于订单的信息被传送到订单编排模块1722。订单编排模块1722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1722可以编排资源的供应，以利用订单供应模块1724的服务支持订阅的服务。

[0416] 在某些实施例中，订单编排模块1722使得能够管理与每个订单相关联的业务流程，并且应用业务逻辑来确定订单是否应当继续供应。在操作1742处，当接收到新订阅的订单时，订单编排模块1722向订单供应模块1724发送分配资源和配置履行该订阅订单所需的那些资源的请求。订单供应模块1724使得能够为由客户订购的服务分配资源。订单供应模块1724在由云基础设施系统1700提供的云服务和用来供应用于提供所请求的服务的资源的物理实现层之间提供抽象层。订单编排模块1722可以因此与实现细节隔离，这些实现细节诸如服务和资源实际上是否被实时供应或者预先被供应并且仅在请求时才进行分配/指定。

[0417] 在操作1744处，一旦供应了服务和资源，就可以通过云基础设施系统1702的订单供应模块1724向客户端设备1704、1706和/或1708上的客户发送所提供的服务的通知。

[0418] 在操作1746处，可以由订单管理和监视模块1726来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1726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、所传送的数据量、用户的数量、以及系统运行时间和系统停机时间的量。

[0419] 在某些实施例中，云基础设施系统1700可以包括身份管理模块1728。身份管理模块1728可以被配置为提供身份服务，诸如云基础设施系统1700中的访问管理和授权服务。在一些实施例中，身份管理模块1728可以控制关于希望利用由云基础设施系统1702提供的服务的客户的信息。这种信息可以包括认证这种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)执行哪些动作的信息。身份管理模块1728也可以包括对关于每个客户的描述性信息以及可以如何访问和修改该描述性信息和由谁来访问和修改该描述性信息的管理。

[0420] 图18示出了其中可以实现本发明的各种实施例的示例计算机系统1800。系统1800可以用来实现上述计算机系统中的任何一个。如图所示，计算机系统1800包括经由总线子系统1802与若干外围子系统通信的处理单元1804。这些外围子系统可以包括处理加速单元1806、I/O子系统1808、存储子系统1818和通信子系统1824。存储子系统1818包括有形的计算机可读存储介质1822和系统存储器1810。

[0421] 总线子系统1802提供了用于让计算机系统1800的各个组件和子系统按意图彼此通信的机制。虽然总线子系统1802被示意性地示为单个总线，但是总线子系统的可替代实施例可以利用多个总线。总线子系统1802可以是若干种类型的总线结构中的任何一种，包括利用各种总线体系架构中的任何架构的存储器总线或存储器控制器、外围总线、以及局部总线。例如，这些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)局部总线和外围组件互连(PCI)总线，其中PCI总线可以实现为按IEEE P1386.1标准制造的夹层(Mezzanine)总线。

[0422] 可以实现为一个或多个集成电路(例如，常规的微处理器或微控制器)的处理单元1804控制计算机系统1800的操作。在处理单元1804中可以包括一个或多个处理器。这些处理器可以包括单核或多核处理器。在某些实施例中，处理单元1804可以实现为一个或多个独立的处理单元1832和/或1834，其中在每个处理单元中包括单个或多核处理器。在其它实施例中，处理单元1804也可以实现为通过将两个双核处理器集成到单个芯片中而形成的四核处理单元。

[0423] 在各种实施例，处理单元1804可以响应于程序代码执行各种程序并且可以维护多个并发执行的程序或进程。在任何给定时间，要执行的程序代码中的一些或全部可以驻留在(一个或多个)处理器1804中和/或存储子系统1818中。通过适当的编程，(一个或多个)处理器1804可以提供上述各种功能。计算机系统1800可以附加地包括处理加速单元1806，处理加速单元1806可以包括数字信号处理器(DSP)、专用处理器等。

[0424] I/O子系统1808可以包括用户接口输入设备和用户接口输出设备。用户接口输入设备可以包括键盘、诸如鼠标或轨迹球的指点设备、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。用户接口输入设备可以包括例如诸如Microsoft Kinect®运动传感器的运动感测和/或姿势识别设备，Microsoft Kinect®运动传感器使得用户能够通过利用姿势和口语命令的自然用户接口控制诸如Microsoft Xbox® 360游戏控制器的输入设备并与其交互。用户接口输入设备也可以包括眼睛姿势识别设备，诸如检测用户的眼睛活动(例如，当拍摄图片和/或进行菜单选择时的“眨眼”)并将眼睛姿势变换为到输入设备(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。

[0425] 用户接口输入设备也可以包括但不限于三维(3D)鼠标、操纵杆或指点杆、游戏板和绘图平板、以及音频/视频设备，诸如扬声器、数码相机、数码摄像机、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，诸如计算机断层扫描、磁共振成像、正电子发射断层扫描、医疗超声设备。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。

[0426] 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视觉显示器等。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1800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。

[0427] 计算机系统1800可以包括存储子系统1818，存储子系统1818包括被示为当前位于系统存储器1810内的软件元件。系统存储器1810可以存储可加载在处理单元1804上并且可在处理单元1804上执行的程序指令，以及在这些程序执行期间生成的数据。

[0428] 取决于计算机系统1800的配置和类型，系统存储器1810可以是易失性的(诸如随机存取存储器(RAM))和/或非易失性的(诸如只读存储器(ROM)、闪存存储器，等等)。RAM通常包含可被处理单元1804立即访问和/或目前正被处理单元1804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1810可以包括多种不同类型的存储器，诸如静态随机存取存储器(SRAM)或动态随机存取存储器(DRAM)。在一些实现中，包含有助于诸如在启动期间在计算机系统1800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。作为示例，而不是限制，系统存储器1810还示出了应用程序1812、程序数据1814以及操作系统1816，其中应用程序1812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。作为示例，操作系统1816可以包括各种版本的Microsoft **Windows®** 、Apple **Macintosh®** 和/或Linux操作系统、各种可商用 **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®** 10OS和 **Palm®** OS操作系统的移动操作系统。

[0429] 存储子系统1818也可以提供有形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。当被处理器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1818中。这些软件模块或指令可以被处理单元1804执行。存储子系统1818也可以提供储存库，用于存储根据本发明被使用的数据。

[0430] 存储子系统1800也可以包括计算机可读存储介质读取器1820，计算机可读存储介质读取器1820可以进一步连接到计算机可读存储介质1822。计算机可读存储介质1822可以与系统存储器1810一起以及可选地与系统存储器1810组合来全面地表示用于临时和/或更持久地包含、存储、传输和检索计算机可读信息的远程、本地、固定和/或可移除存储设备加存储介质。

[0431] 包含代码或代码的部分的计算机可读存储介质1822也可以包括本领域已知或使用的任何适当的介质，包括存储介质和通信介质，诸如但不限于以用于信息的存储和/或传输的任何方法或技术实现的易失性和非易失性、可移除和不可移除介质。这可以包括有形的计算机可读存储介质，诸如RAM、ROM、电可擦除可编程ROM(EEPROM)、闪存存储器或其它存储器技术、CD-ROM、数字多功能盘(DVD)或其它光学存储装置、磁带盒、磁带、磁盘存储装置或其它磁存储设备、或其它有形的计算机可读介质。这也可以包括非有形的计算机可读介质，诸如数据信号、数据传输，或者可以用来传输期望的信息并且可以被计算机系统1800访问的任何其它介质。

[0432] 作为示例，计算机可读存储介质1822可以包括从不可移除的非易失性磁介质读取或写到其的硬盘驱动器、从可移除的非易失性磁盘读取或写到其的磁盘驱动器、以及从可移除的非易失性光盘(诸如CD ROM、DVD和Blu-Ray®盘或其它光学介质)读取或写到其的光盘驱动器。计算机可读存储介质1822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、数字音频带等。计算机可读存储介质1822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(诸如固态RAM、动态RAM、静态RAM，基于DRAM的SSD，磁阻RAM(MRAM)SSD)，以及使用基于DRAM的SSD和基于闪存存储器的SSD的组合的混合SSD。盘驱动器及其相关联的计算机可读介质可以为计算机系统1800提供计算机可读指令、数据结构、程序模块及其它数据的非易失性存储装置。

[0433] 通信子系统1824提供到其它计算机系统和网络的接口。通信子系统1824用作用于从其它系统接收数据和从计算机系统1800向其它系统传输数据的接口。例如，通信子系统1824可以使计算机系统1800能够经由互联网连接到一个或多个设备。在一些实施例中，通信子系统1824可以包括用于访问无线语音和/或数据网络的射频(RF)收发器组件(例如，利用蜂窝电话技术，诸如3G、4G或EDGE(用于全球演进的增强型数据速率)的先进数据网络技术，Wi-Fi(IEEE 1602.11标准族)，或其它移动通信技术，或其任意组合)、全球定位系统(GPS)接收器组件和/或其它组件。在一些实施例中，作为无线接口的附加或替代，通信子系统1824可以提供有线网络连接(例如，以太网)。

[0434] 在一些实施例中，通信子系统1824也可以代表可以使用计算机系统1800的一个或多个用户接收结构化和/或非结构化的数据馈送1826、事件流1828、事件更新1830等形式的输入通信。

[0435] 作为示例，通信子系统1824可以被配置为实时地从社交网络和/或其它通信服务的用户接收数据馈送1826，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。

[0436] 此外，通信子系统1824也可以被配置为接收连续数据流形式的数据，连续数据流可以包括实时事件的事件流1828和/或事件更新1830，连续数据流形式的数据本质上可能是连续的或无界的而没有明确结束。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。通信子系统1824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1826、事件流1828、事件更新1830等，其中这一个或多个数据库可以与耦合到计算机系统1800的一个或多个流化数据源计算机通信。

[0437] 计算机系统1800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)、PC、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。

[0438] 本实施例的计算机系统1800及其组件可以被配置为基于如先前的实施例中所描述的本发明的原理来执行任何适当的操作。例如，包括在这种计算机系统中的通信子系统可以接收事件流。包括在计算机系统中的处理器或处理单元可以创建表示数据存储库的数

据存储库事件的表组件，并且使用表组件作为外部关系源来实现事件处理网络的引擎，以从数据存储库中选择对应于事件流的特定事件的特定数据，并将该特定数据添加到该特定事件。

[0439] 优选地，事件处理网络的引擎被配置为至少部分地基于映射来识别特定数据。优选地，映射可以被配置为在表组件的创建之后被更新。映射可以包括在表组件中作为列族。数据存储库可以包括数据存储位置的集群。数据存储库中的至少一个可以包括非关系型数据库或者引擎包括连续查询语言引擎。

[0440] 对于本领域技术人员显而易见的是，对于上述单元的特定操作过程，可以对共享相同概念的相关方法/系统实施例中的对应步骤/组件进行参考，并且该参考也被视为相关单元的公开。因此，为了描述的简洁，将不再重复或详细描述特定操作过程中的一些。

[0441] 由于计算机和网络不断变化的本质，在图中绘出的计算机系统1800的描述旨在仅仅作为具体示例。具有比图中所绘出的系统更多或更少组件的许多其它配置是可能的。例如，定制的硬件也可以被使用和/或特定的元件可以用硬件(包括FPGA、ASIC等)、固件、软件(包括applets)或其组合来实现。另外，可以采用到诸如网络输入/输出设备的其它计算设备的连接。基于本文所提供的公开和教导，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。

[0442] 在前述说明书中，参考本发明的具体实施例描述了本发明的各方面，但是本领域技术人员将认识到本发明不限于此。上述发明的各种特征和方面可以被独自使用或联合使用。此外，在不脱离本说明书的较广泛的精神和范围的情况下，实施例可以在超过本文所描述的任何数量的环境和应用中加以利用。因此，说明书和附图被视为说明性的而不是限制性的。

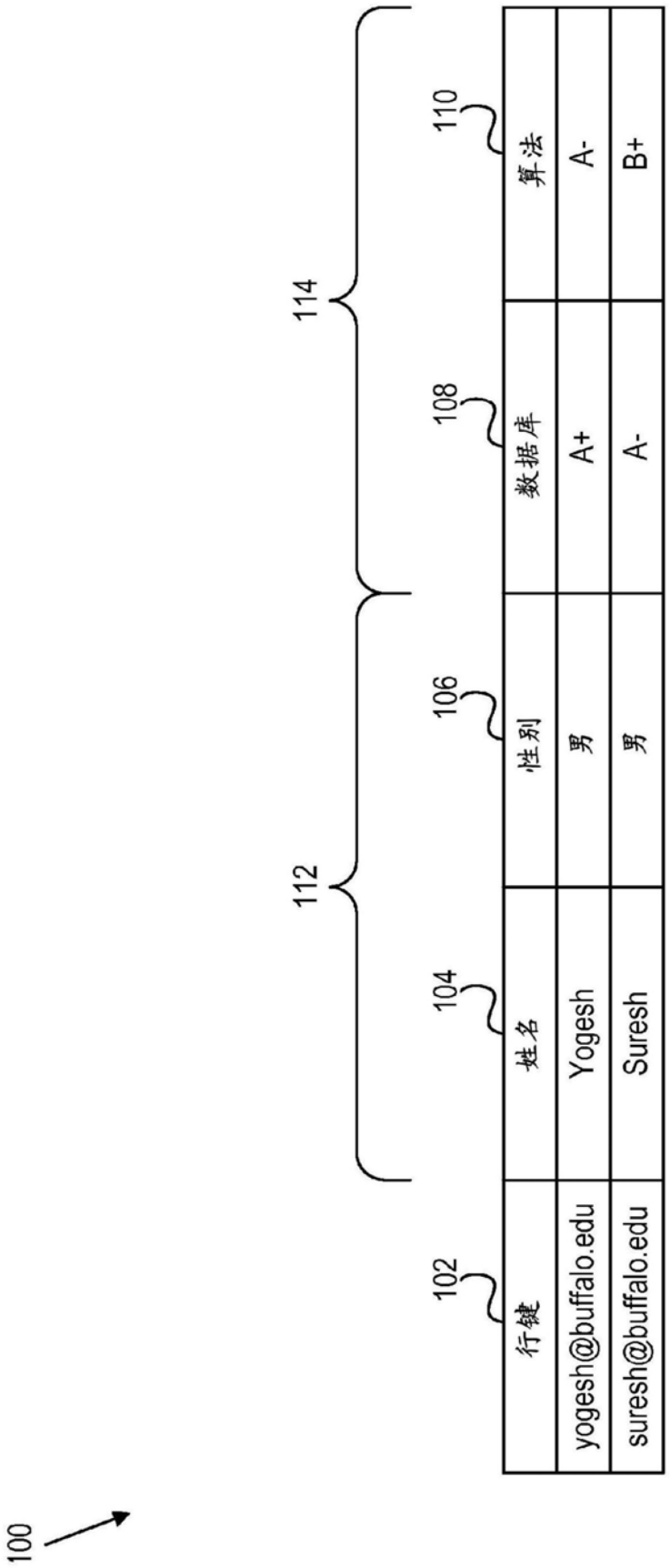

100
102
104
106
108
110
112
114
行键
姓名
性别
数据库
算法
yogesh@buffalo.edu
Yogesh
男
A+
A-
suresh@buffalo.edu
Suresh
男
A-
B+

图1

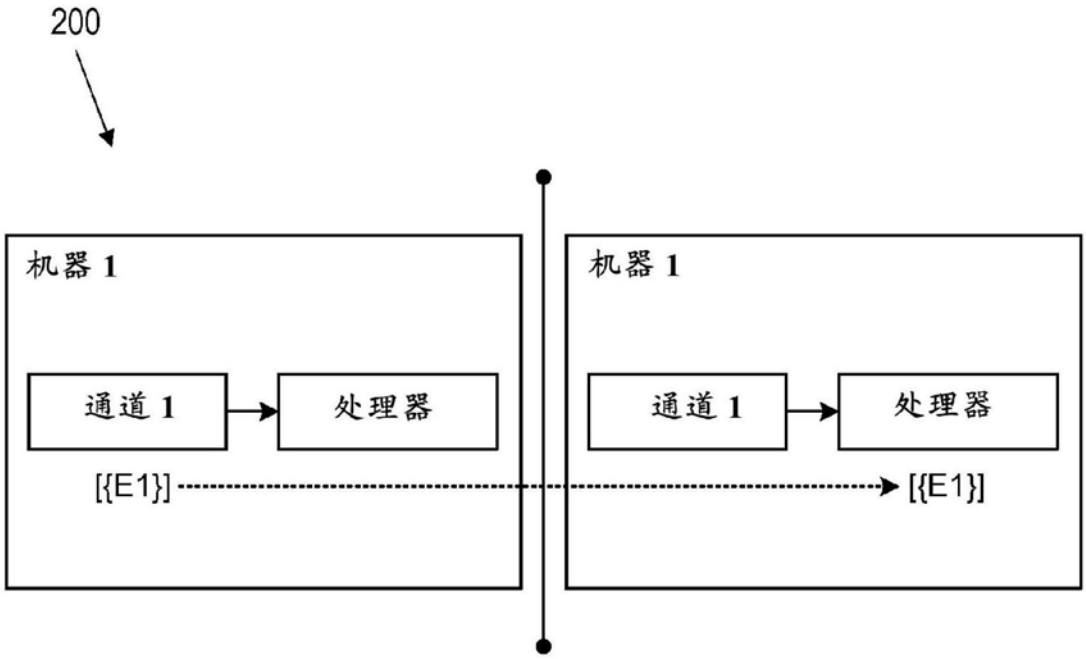


图2

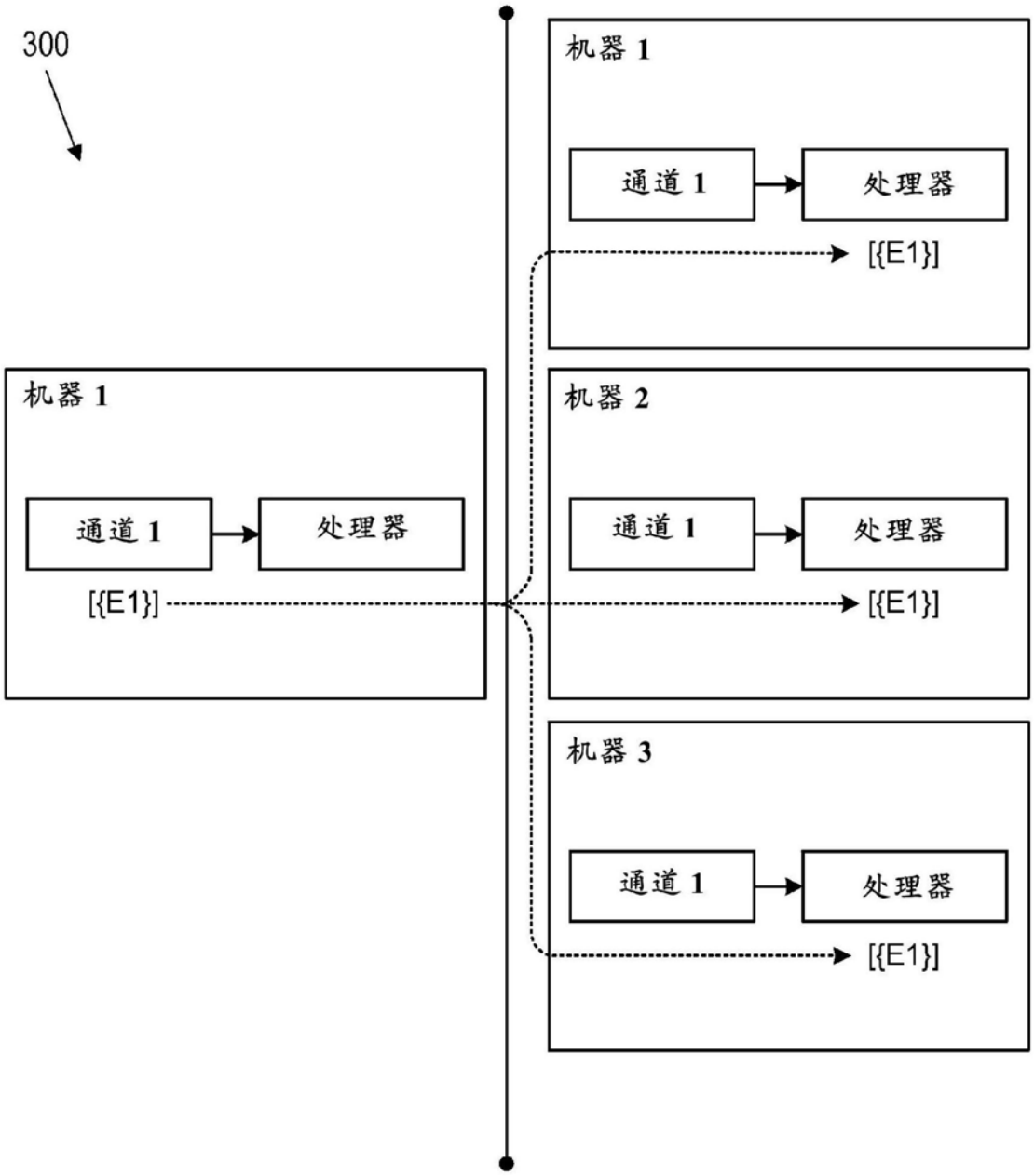


图3

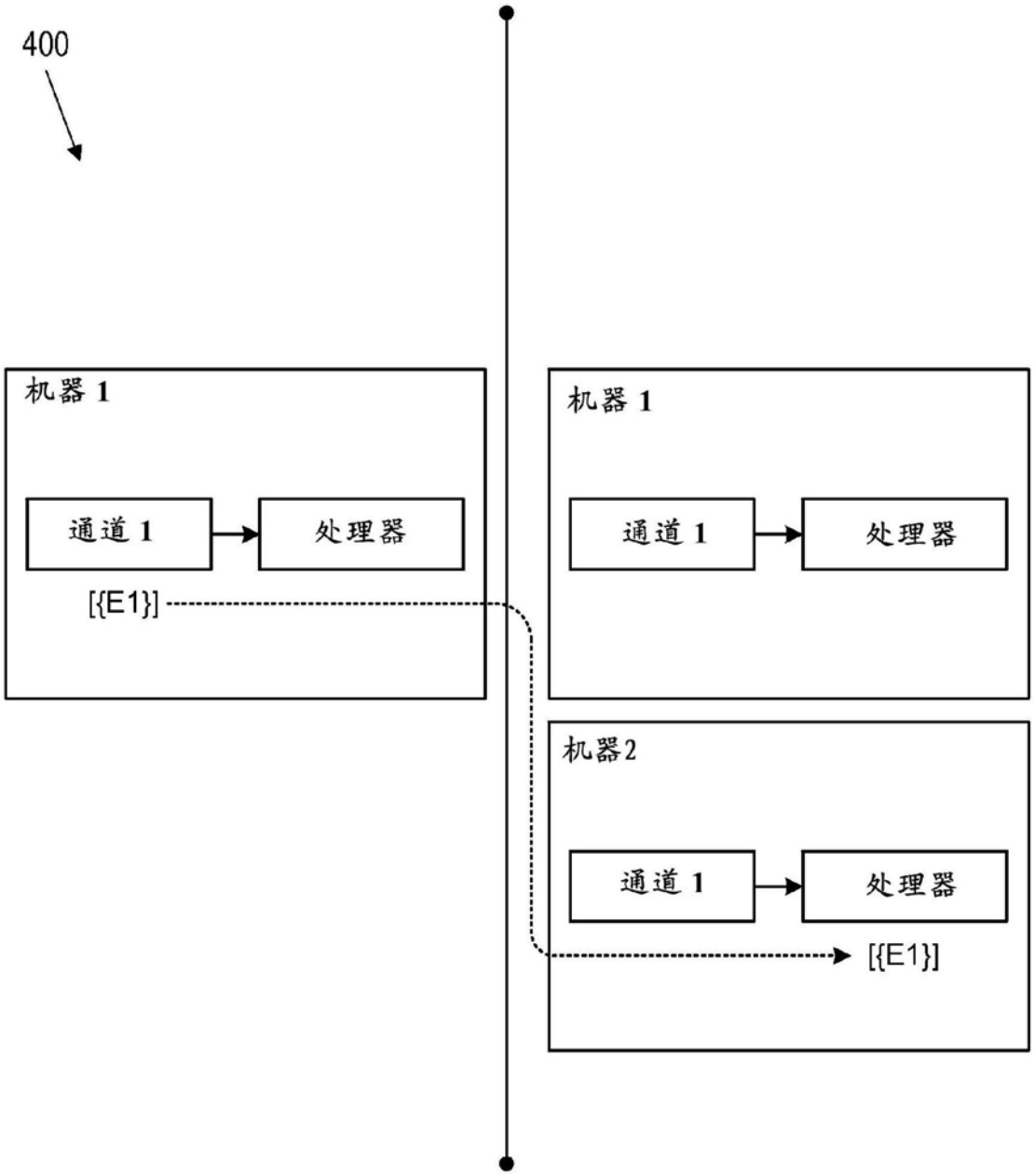


图4

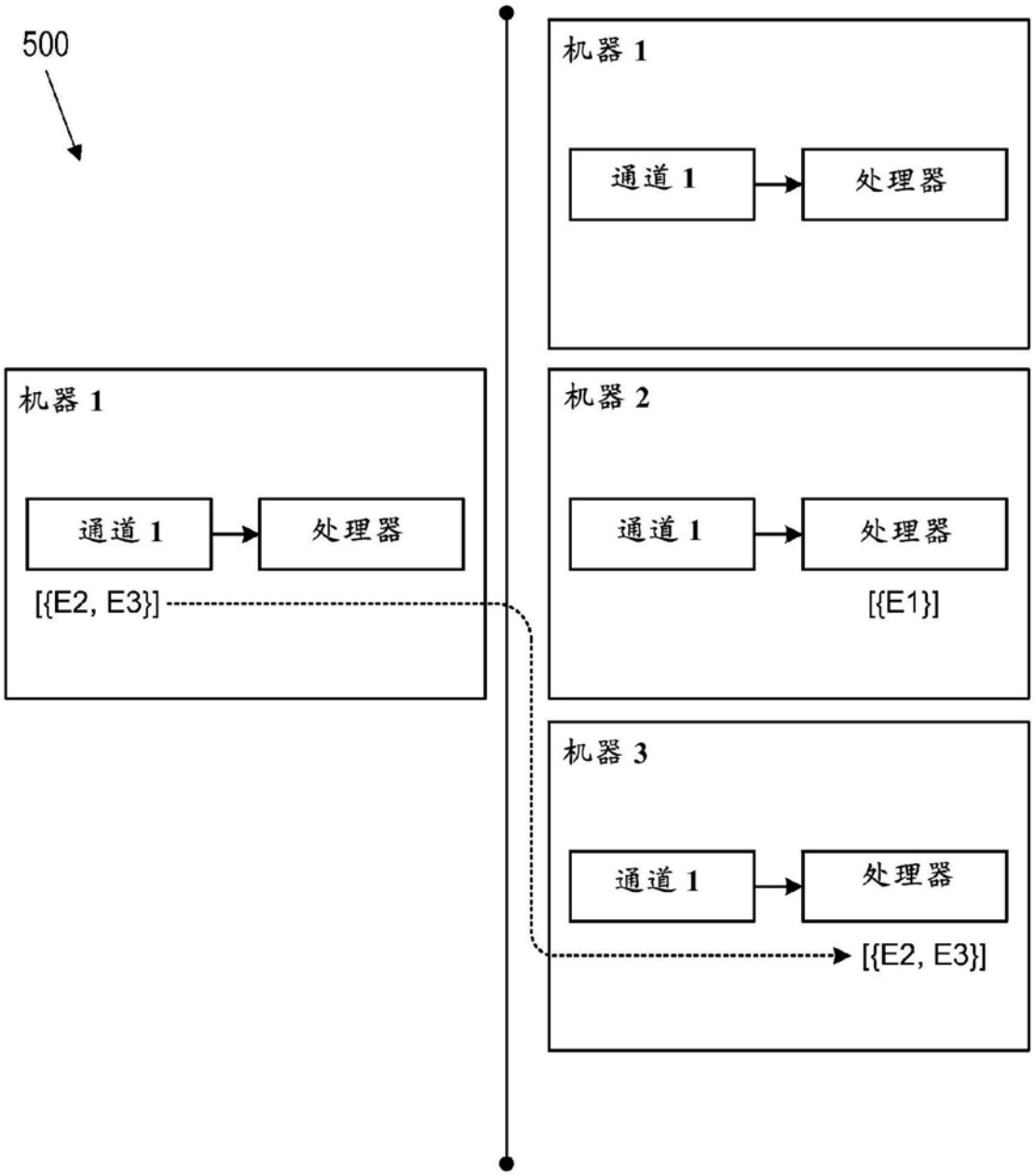


图5

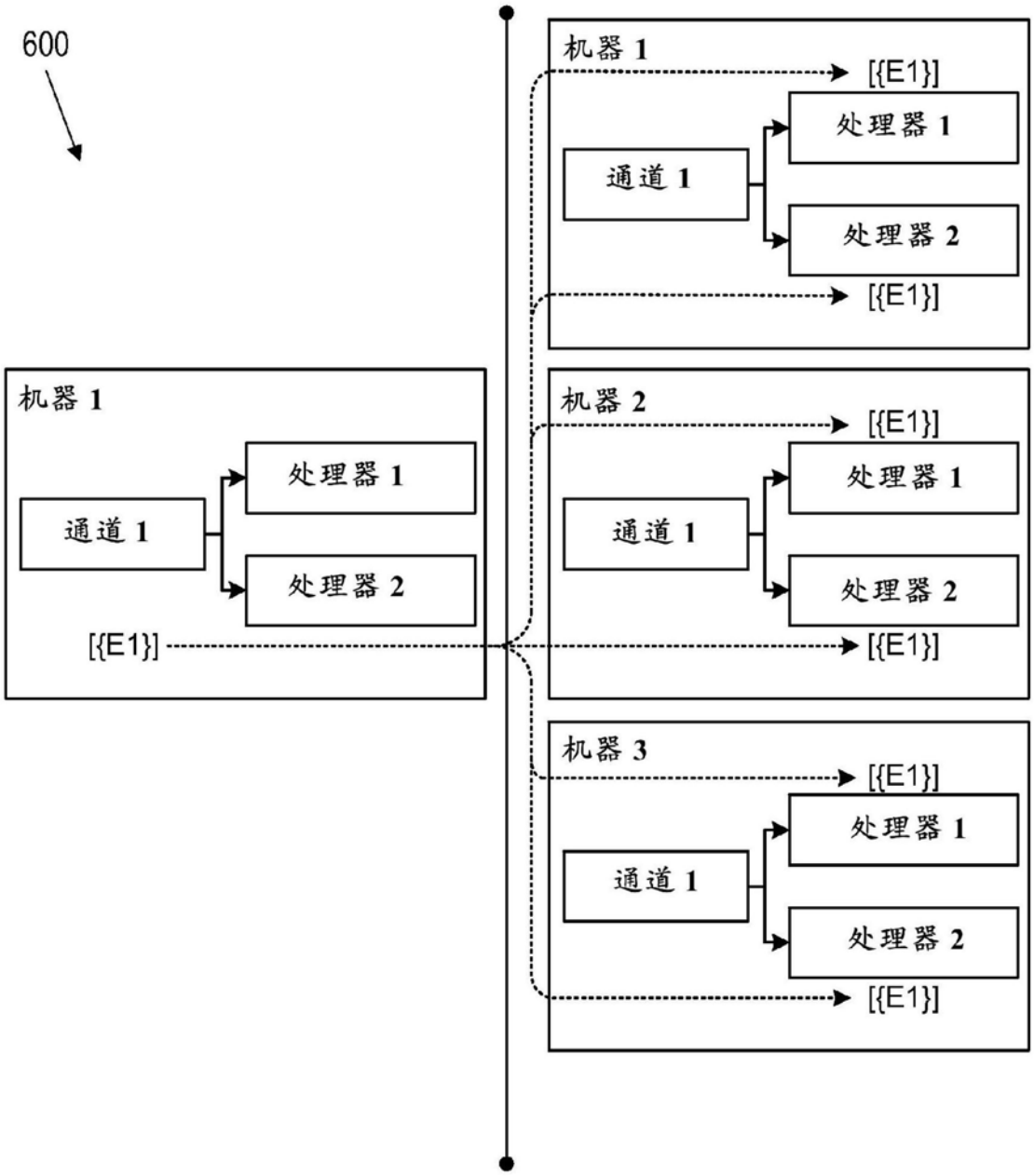

600
机器 1
[{E1}]
处理器 1
通道 1
处理器 2
[{E1}]
机器 1
通道 1
处理器 1
处理器 2
[{E1}]
[{E1}]
机器 2
[{E1}]
通道 1
处理器 1
处理器 2
[{E1}]
机器 3
[{E1}]
通道 1
处理器 1
处理器 2
[{E1}]

图6

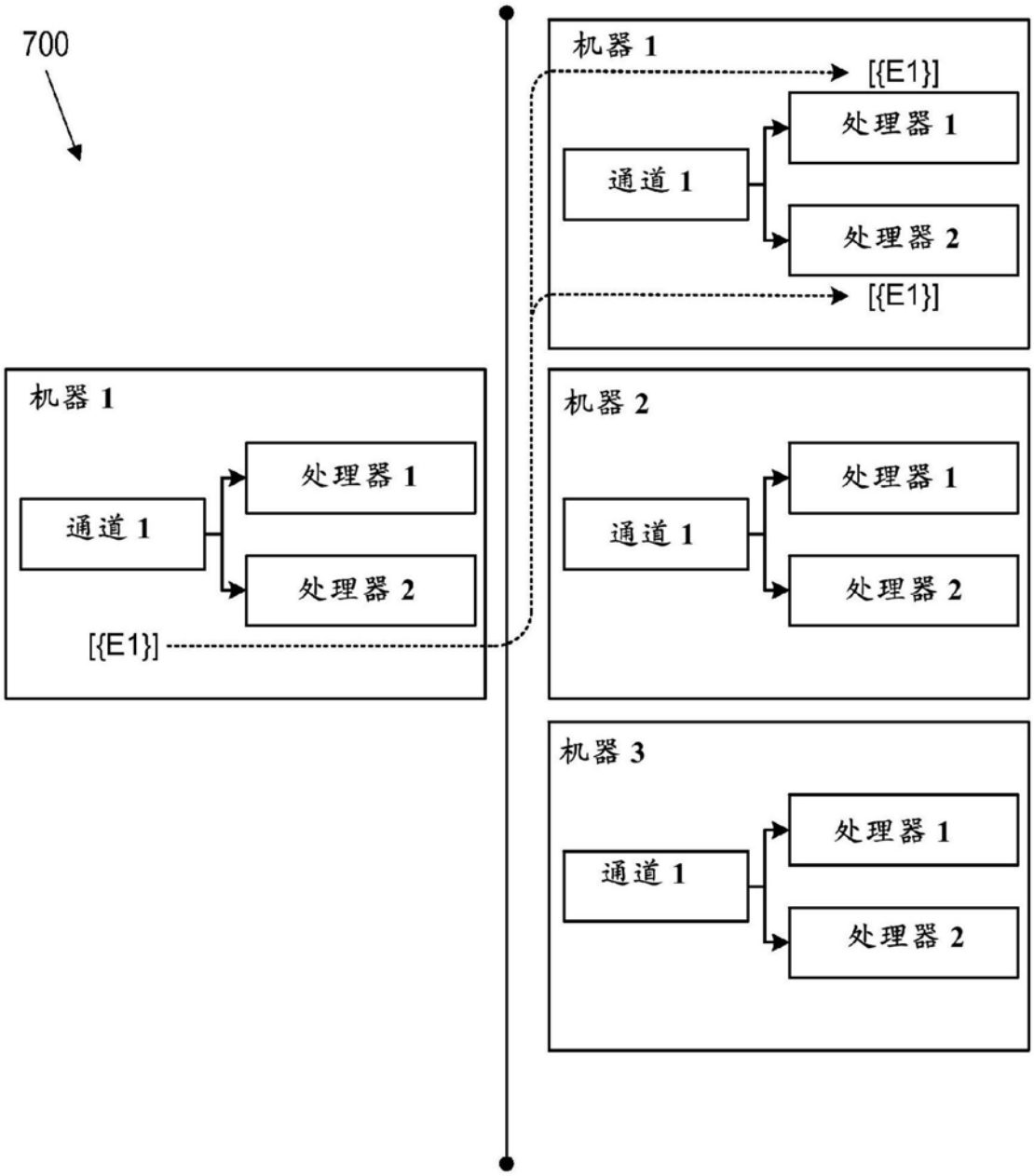

700
机器 1
[{E1}]
处理器 1
通道 1
处理器 2
[{E1}]
机器 1
通道 1
处理器 1
处理器 2
[{E1}]
机器 2
通道 1
处理器 1
处理器 2
机器 3
通道 1
处理器 1
处理器 2

图7

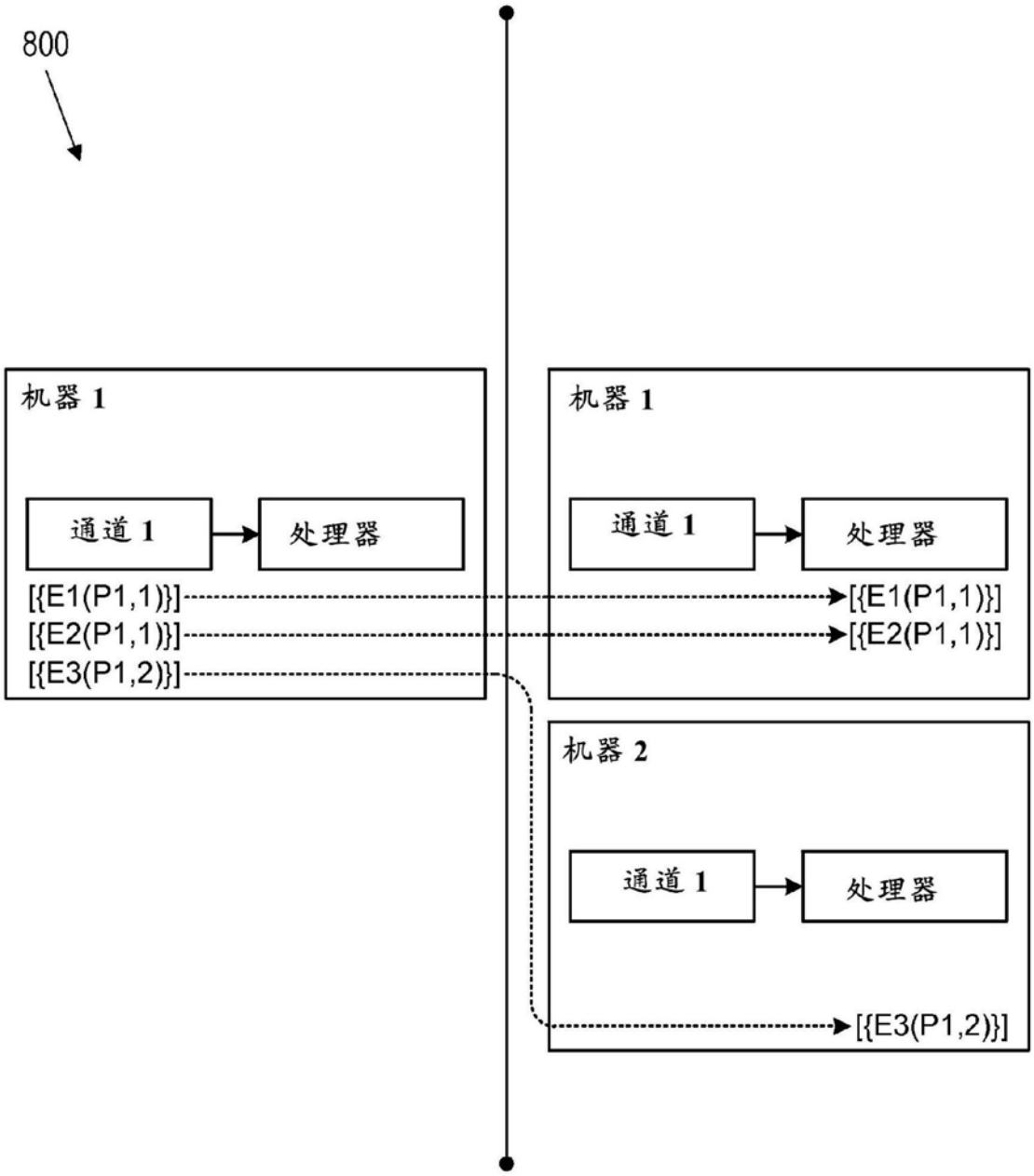

800
机器 1
通道 1
处理器
[{E1(P1,1)}]
[{E2(P1,1)}]
[{E3(P1,2)}]
机器 1
通道 1
处理器
[{E1(P1,1)}]
[{E2(P1,1)}]
机器 2
通道 1
处理器
[{E3(P1,2)}]

图8

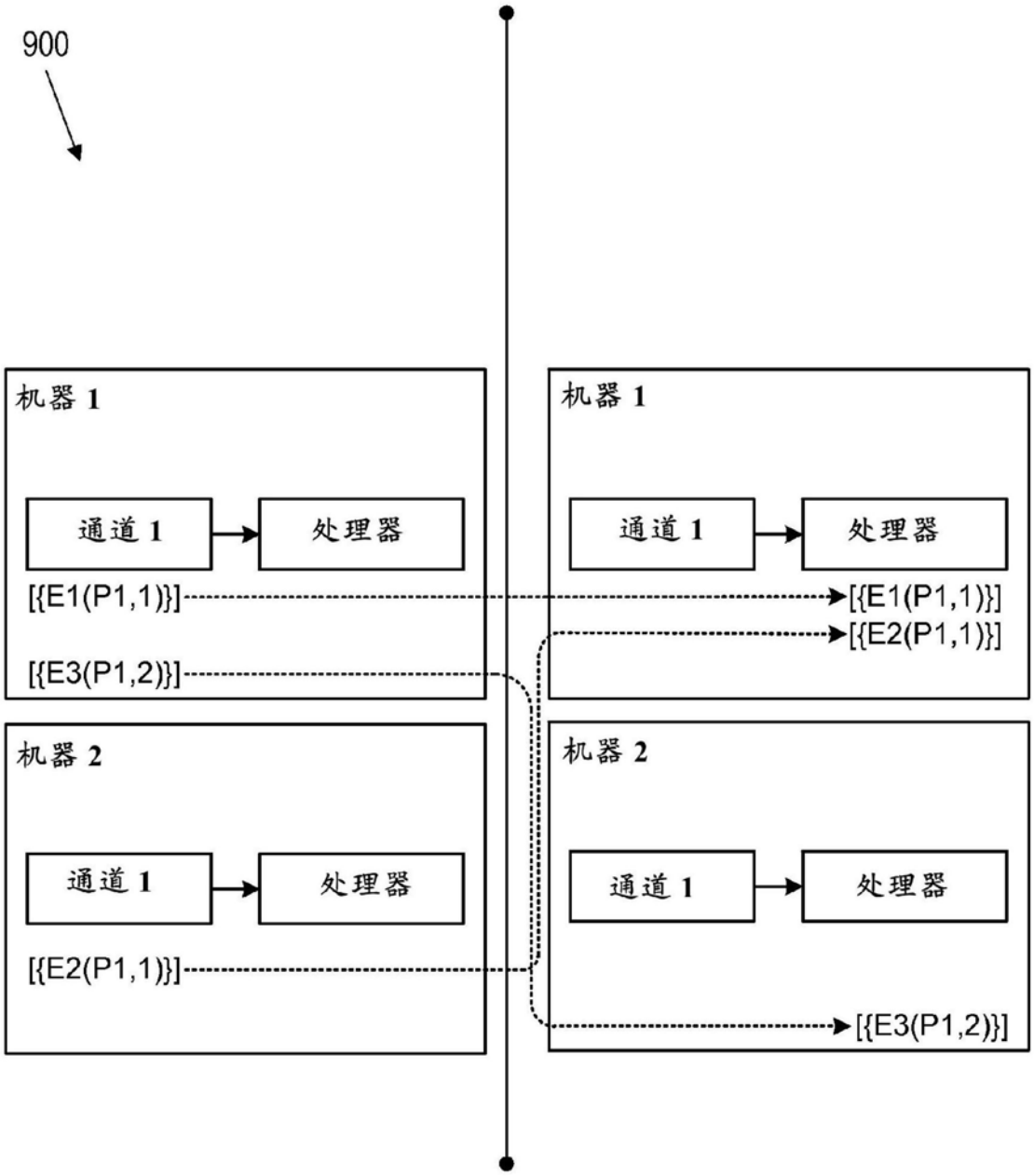

900
机器 1
通道 1
处理器
[{E1(P1,1)}]
[{E3(P1,2)}]
机器 1
通道 1
处理器
[{E1(P1,1)}]
[{E2(P1,1)}]
机器 2
通道 1
处理器
[{E2(P1,1)}]
机器 2
通道 1
处理器
[{E3(P1,2)}]

图9

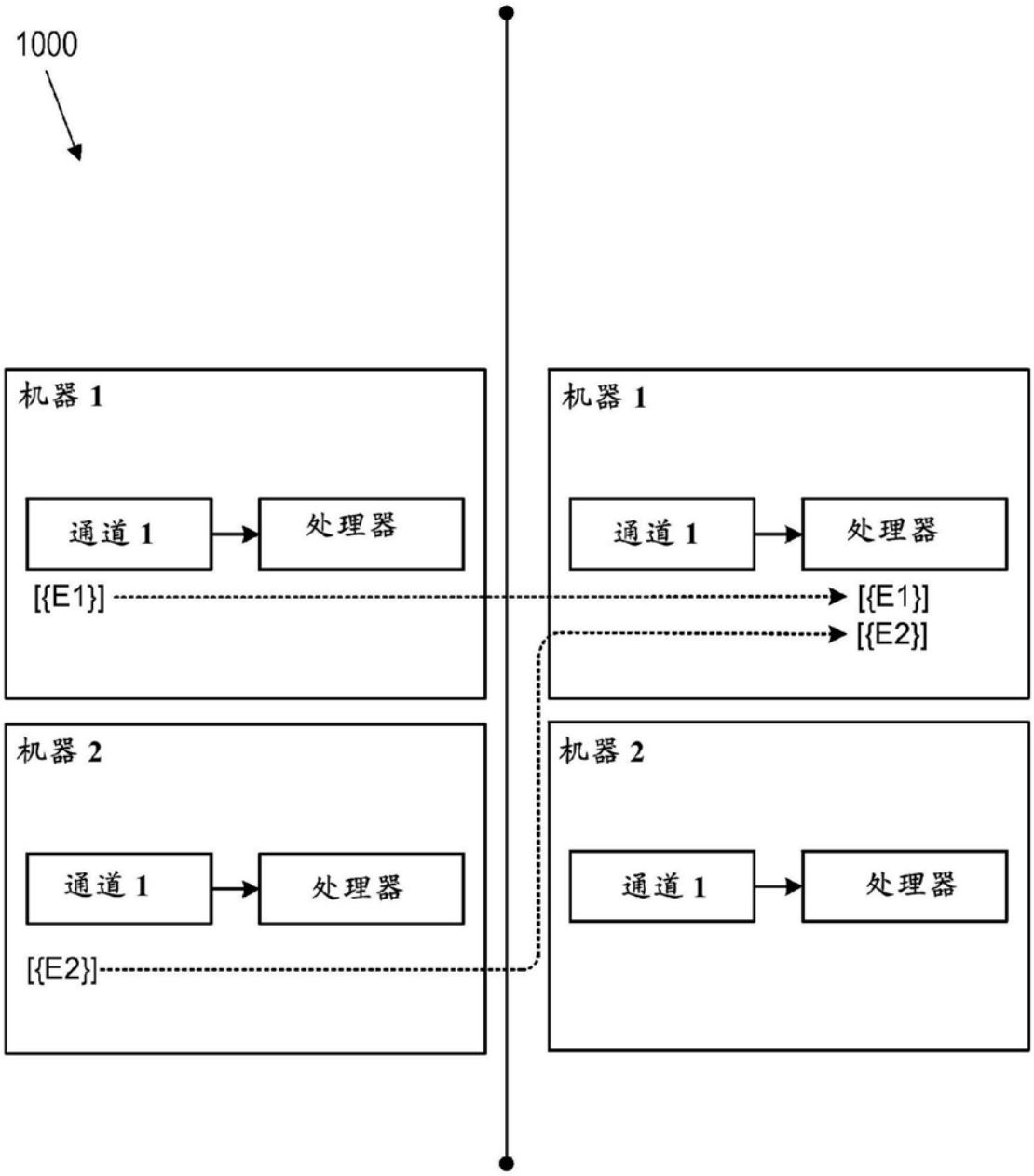


图10

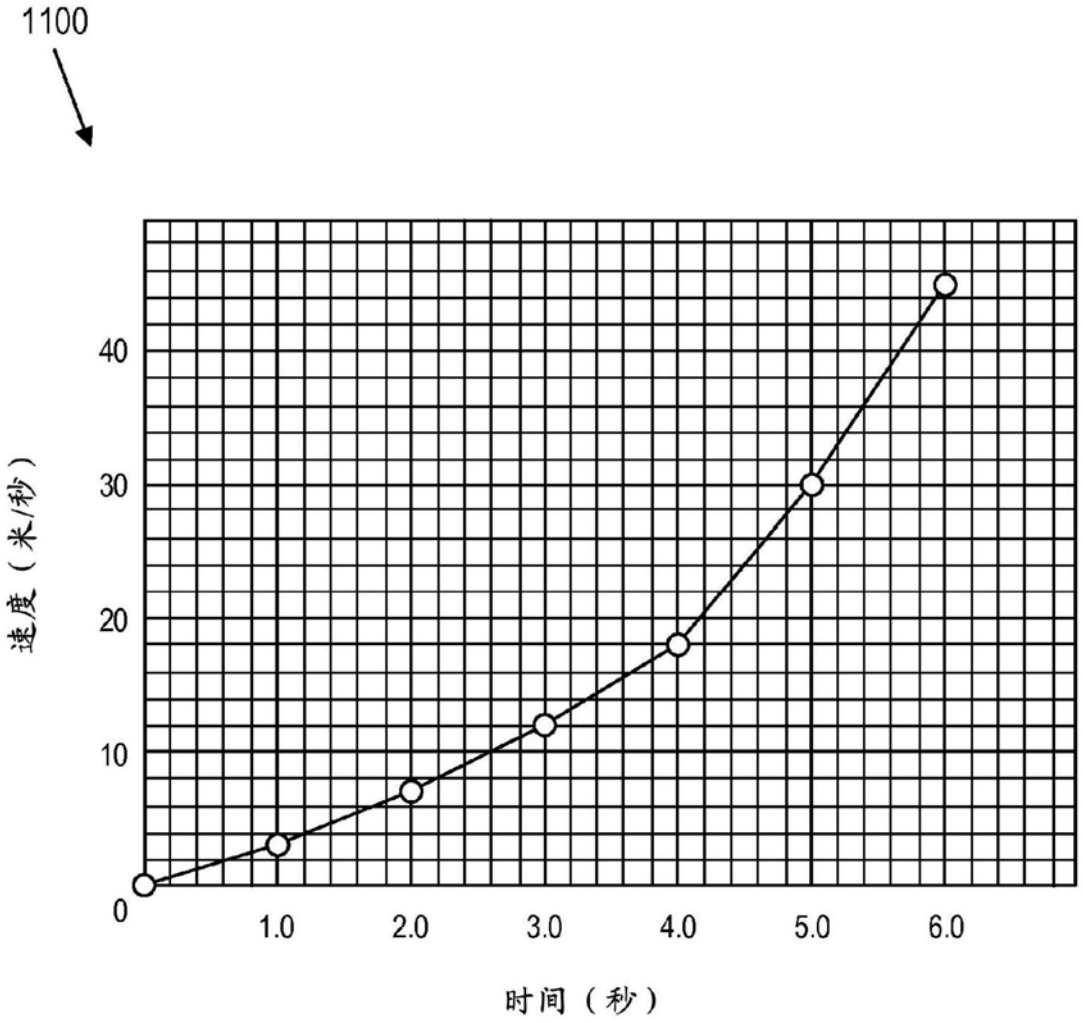


图11

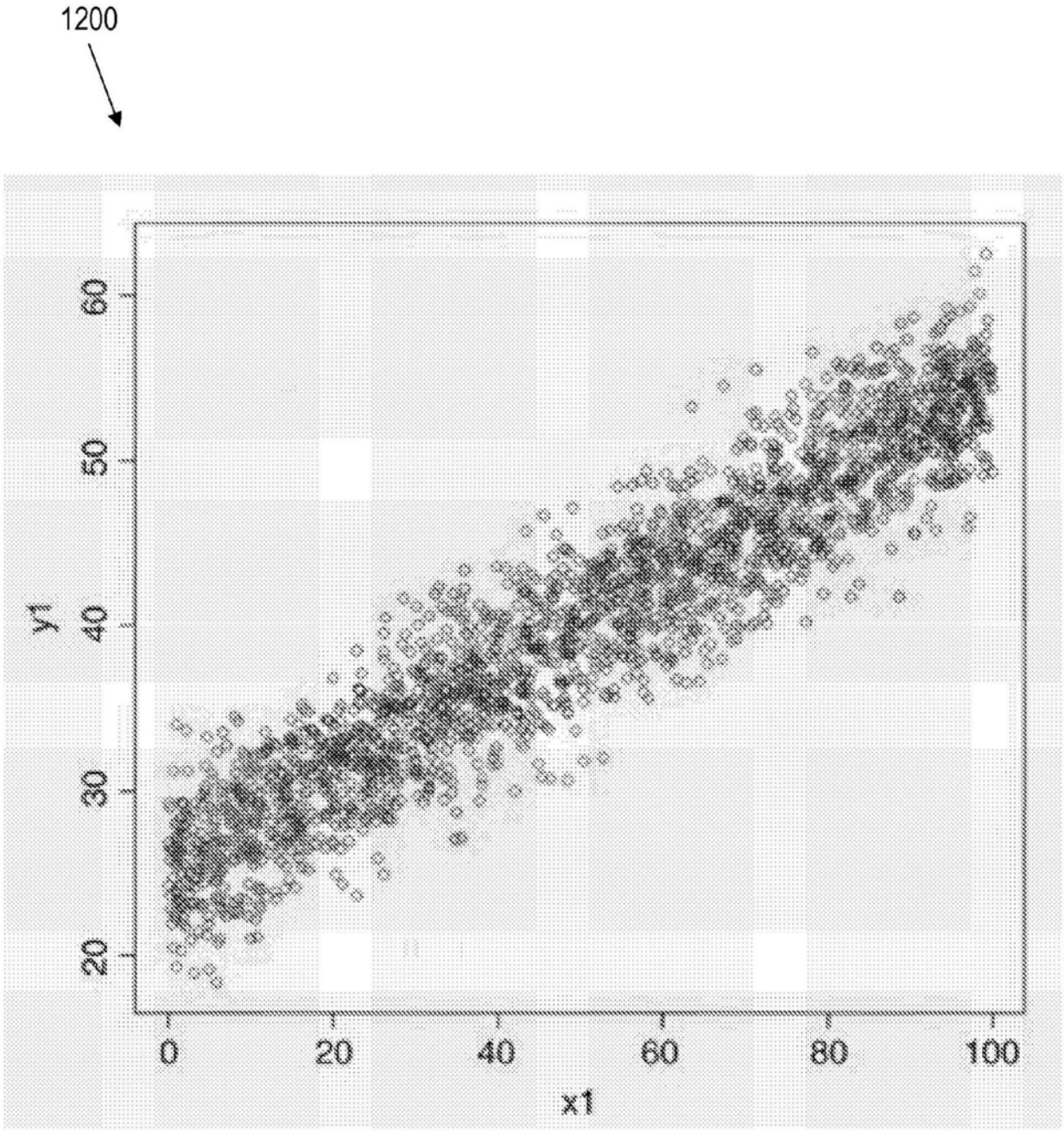


图12

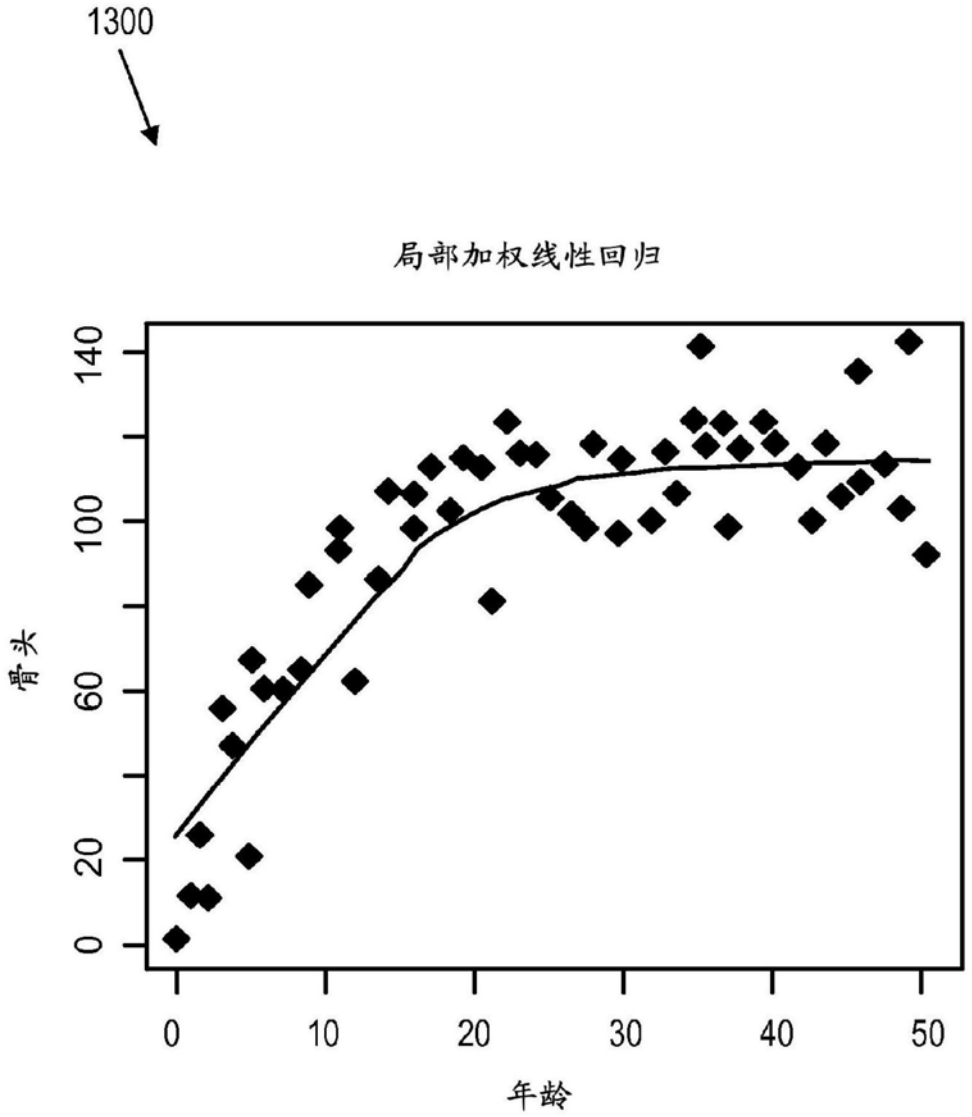


图13

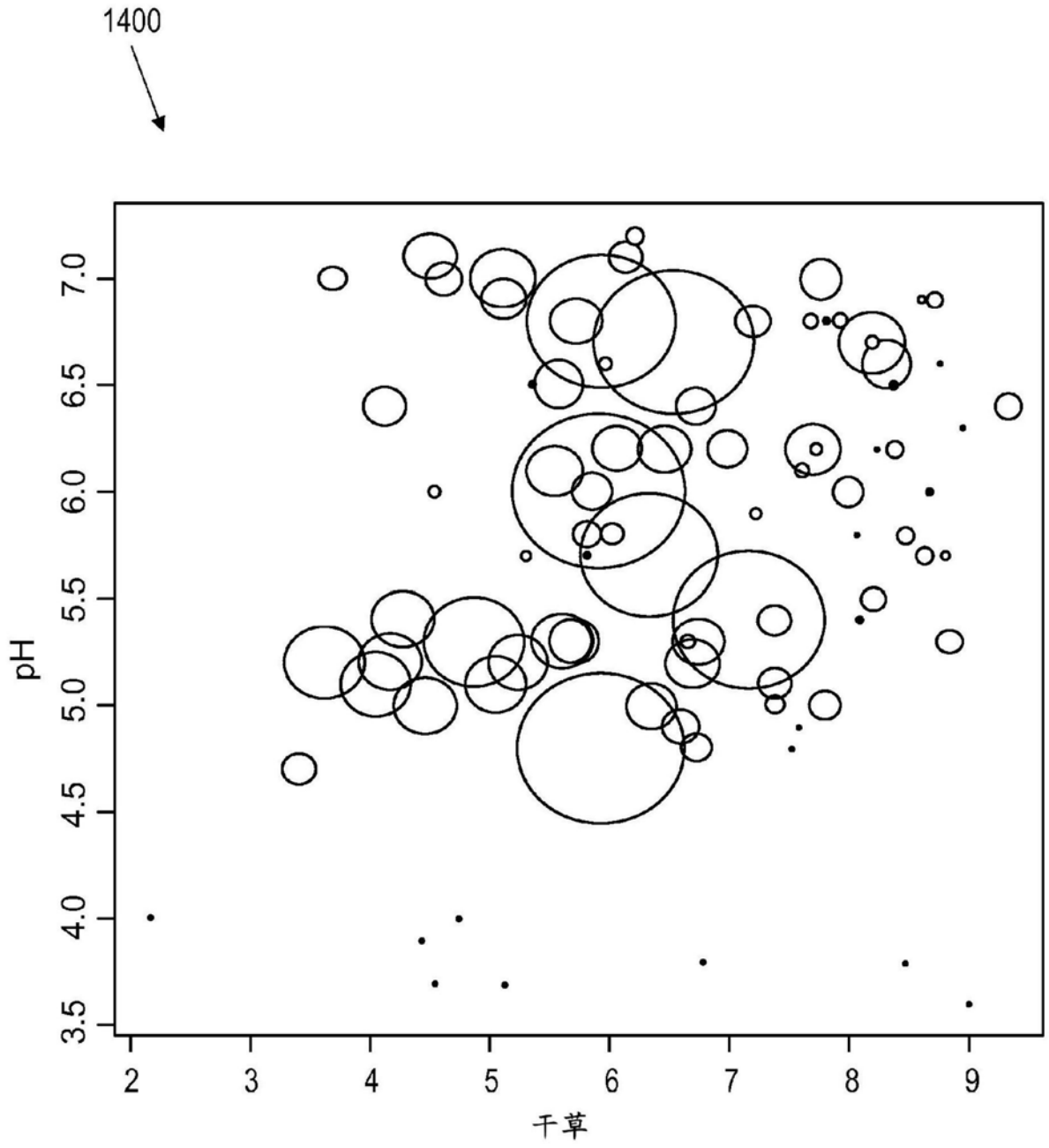


图14

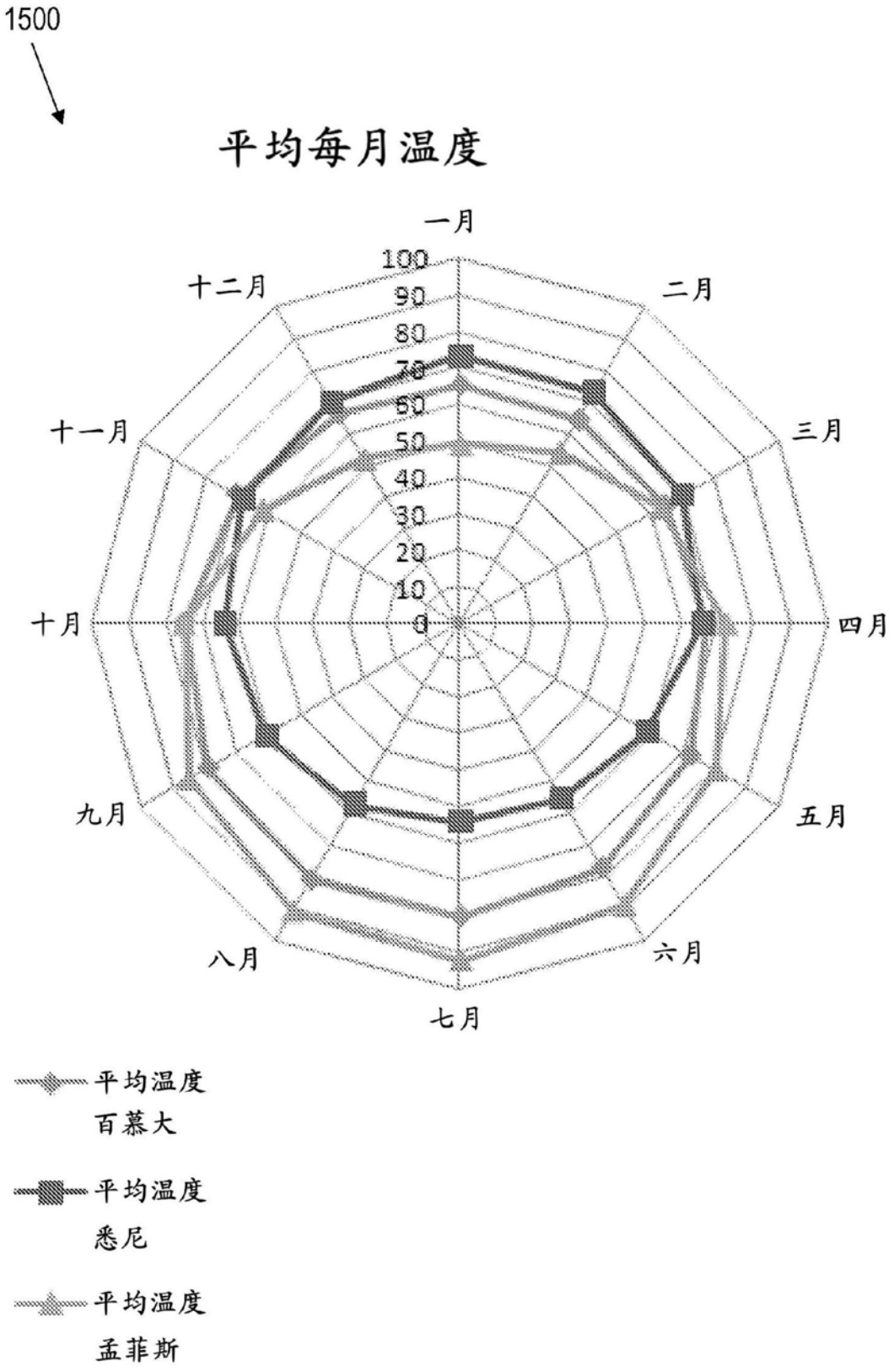

1500
平均每月温度
一月
二月
三月
四月
五月
六月
七月
八月
九月
十月
十一月
十二月
100
90
80
70
60
50
40
30
20
10
0
平均温度
百慕大
平均温度
悉尼
平均温度
孟菲斯

图15

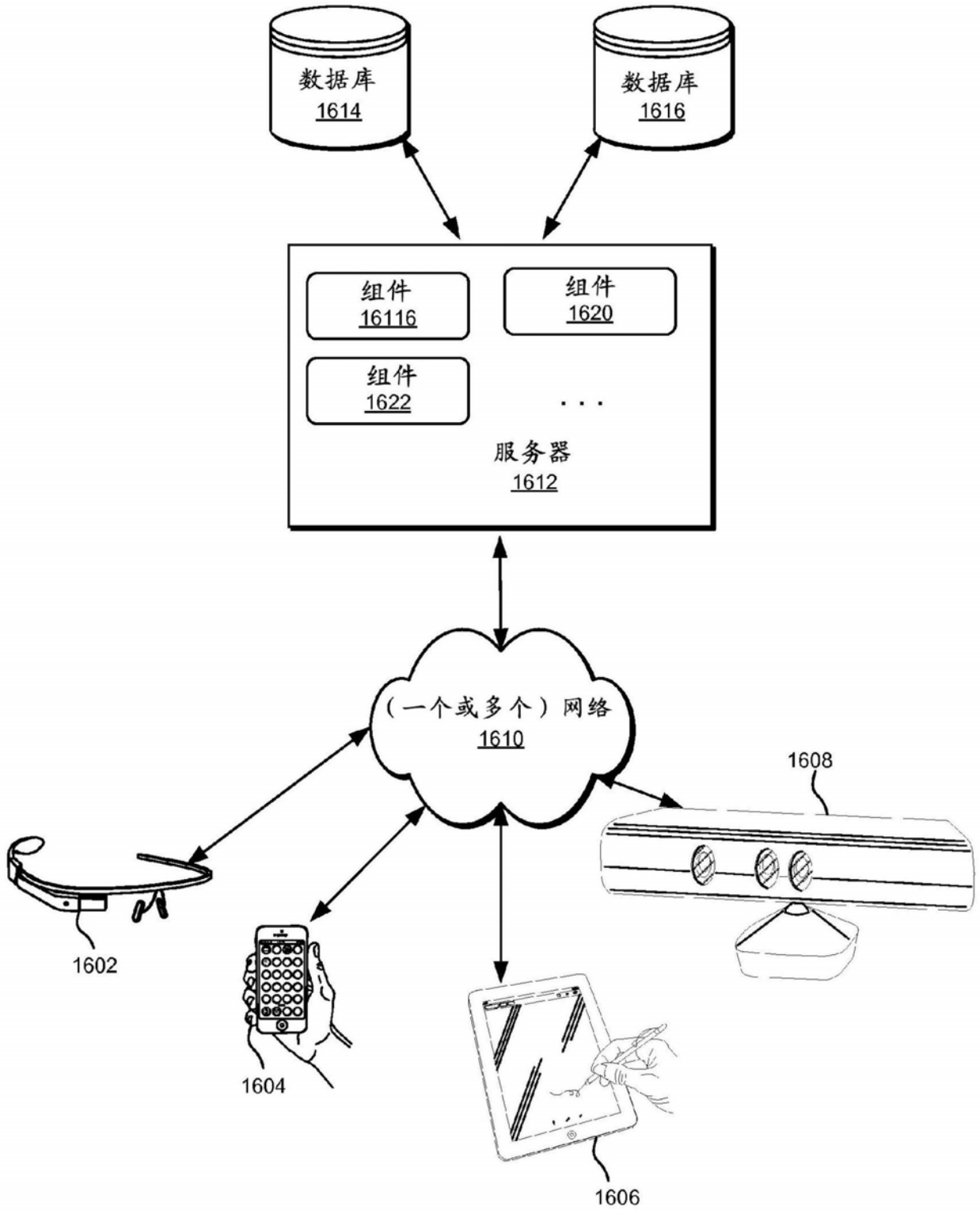

数据库
1614
数据库
1616
组件
16116
组件
1620
组件
1622
. . .
服务器
1612
（一个或多个）网络
1610
1608
1602
1604
1606

图16

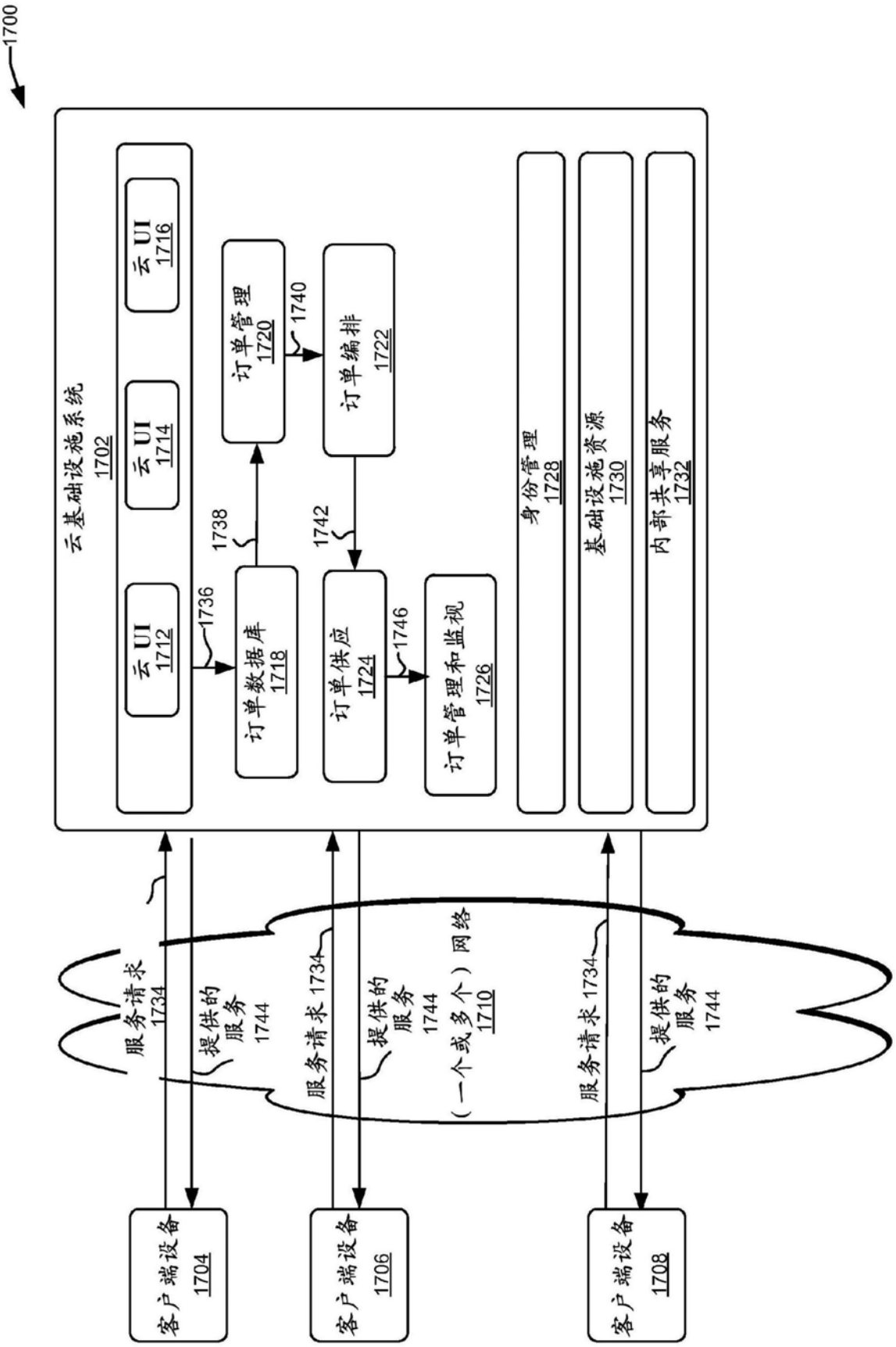

1700
云基础设施系统
1702
云 UI
1712
云 UI
1714
云 UI
1716
订单数据库
1718
订单管理
1720
订单编排
1722
订单供应
1724
订单管理和监视
1726
身份管理
1728
基础设施资源
1730
内部共享服务
1732
1736
1738
1740
1742
1746
服务请求
1734
提供的
服务
1744
服务请求 1734
提供的
服务
1744
（一个或多个）网络
1710
服务请求 1734
提供的
服务
1744
客户端设备
1704
客户端设备
1706
客户端设备
1708

图17

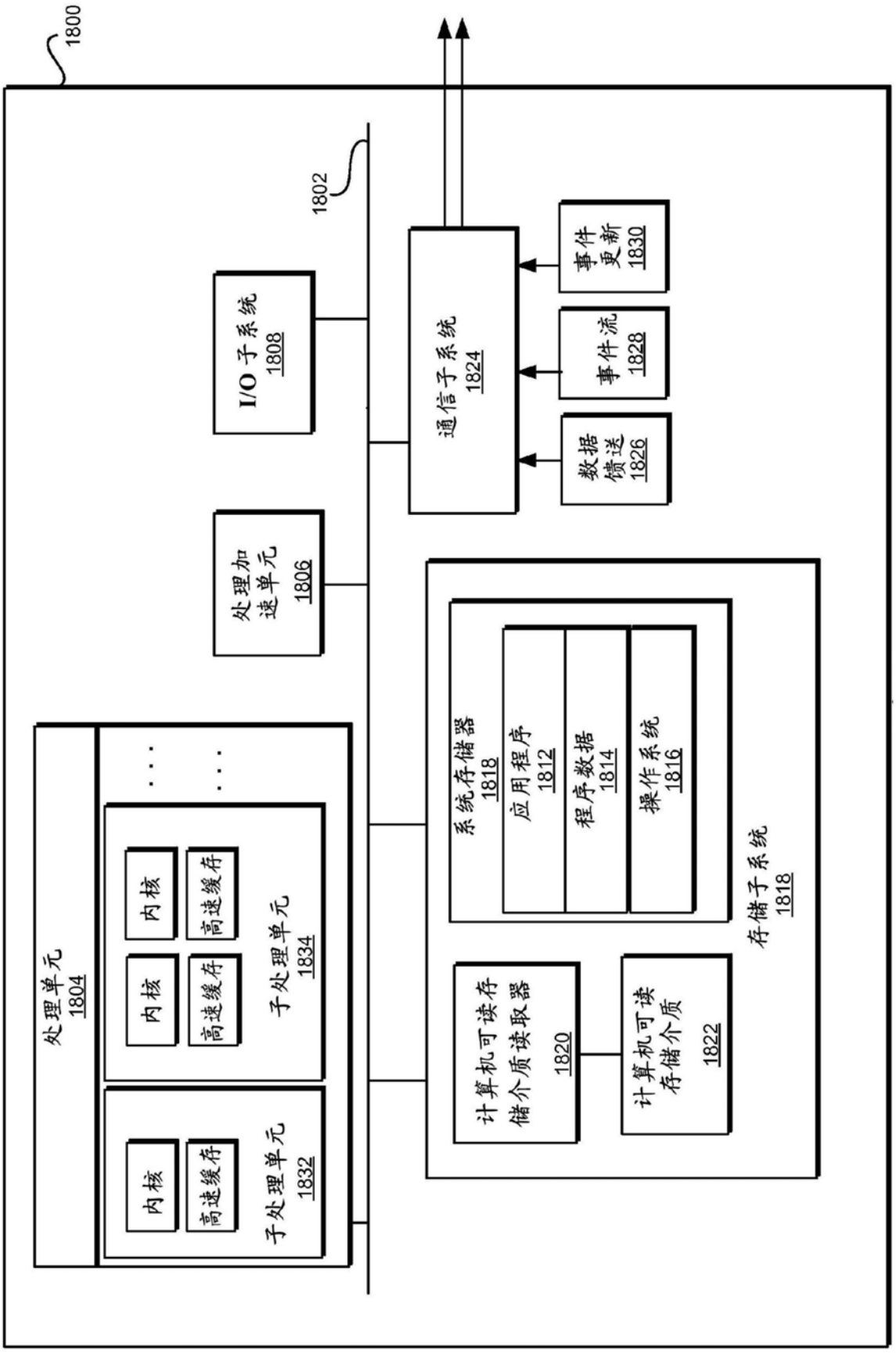

1800
1802
处理单元
1804
子处理单元
1832
内核
高速缓存
子处理单元
1834
内核
高速缓存
内核
高速缓存
I/O 子系统
1808
处理加速单元
1806
通信子系统
1824
事件更新
1830
事件流
1828
数据馈送
1826
存储子系统
1818
系统存储器
1818
应用程序
1812
程序数据
1814
操作系统
1816
计算机可读存储介质读取器
1820
计算机可读存储介质
1822

图18

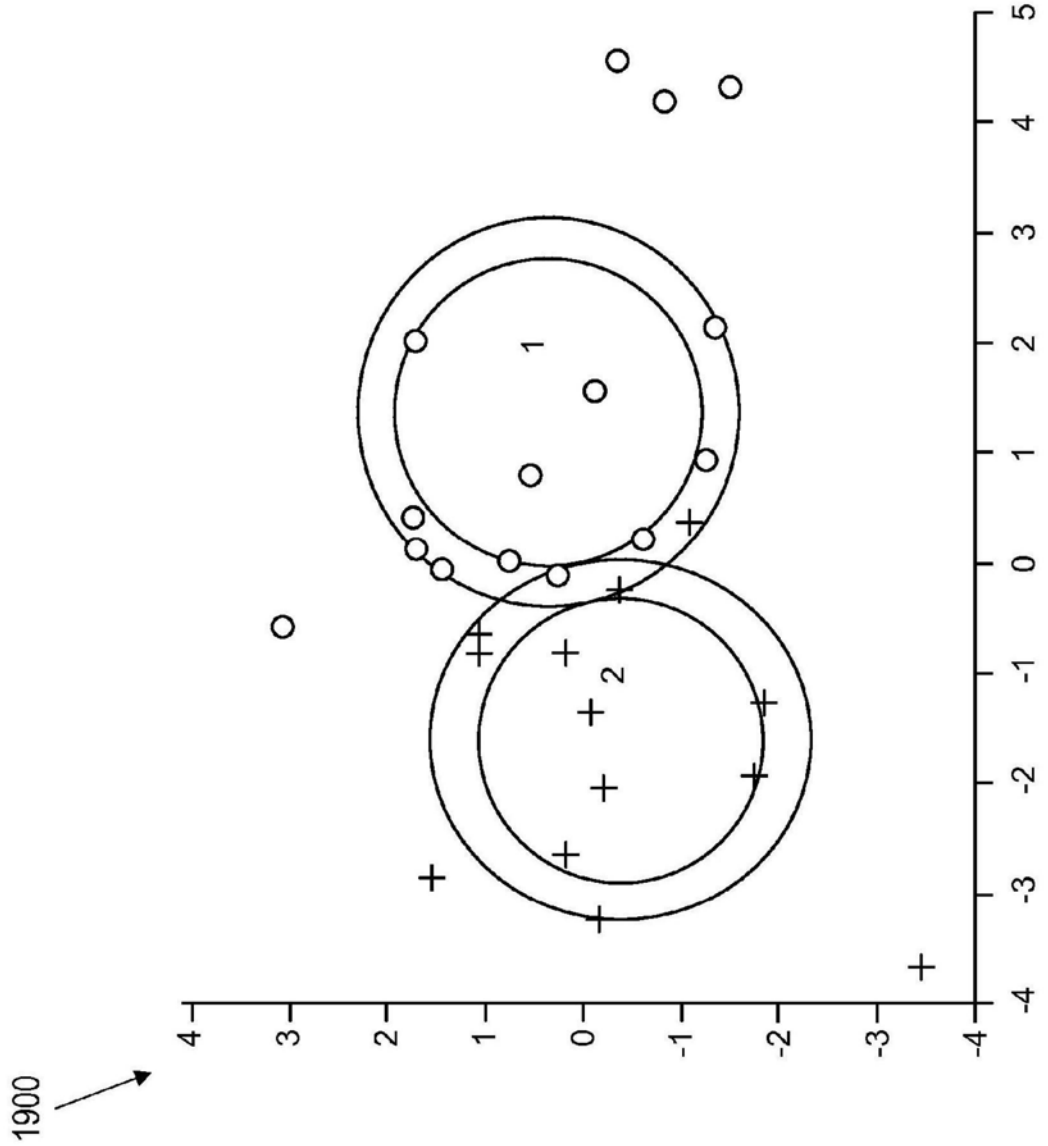


图19