

有限责任公司 11258

(51) Int.Cl.

G05B 15/02 (2006.01)

H04L 12/28 (2006.01)

H04L 67/12 (2022.01)

H04L 67/561 (2022.01)

(56) 对比文件

CN 104104538 A.2014.10.15

US 2012110373 A1, 2012.05.03

US 2009055126 A1, 2009.02.26

US 2014359552 A1, 2014.12.04

Mohammad Aazam等.Fog Computing and
rt Gateway Based Communication for
ud of Things.2014 International
ference on Future Internet of Thing
Cloud.2014,第464-470页.

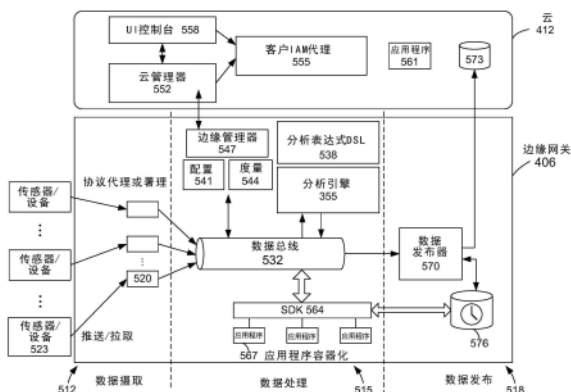
宙查员 李湘伟

权利要求书2页 说明书25页 附图6页

地时间序列数据库或云存储中。服务和组件可以被容器化以确保在任何网关环境中的平稳运行。

(57) 摘要

本文涉及用于改进对分布式网络中的传感器流数据的处理的方法。其特征包括：由在网关设备或嵌入式系统上托管的软件层中的传感器数据触发。软件层被连接到局域网。服务、应用程序和数据处理引擎的存储库被使得能够由软件层访问。通过由软件层提供的表达式语言将传感器数据与对特定状况的发生的语义描述进行匹配。通过持续执行表达式来自动发现模式事件。智能地跨越网关设备和网络上的由软件层管理的嵌入式系统对服务和应用程序进行编排以链接应用程序和分析表达式。基于资源可用性来优化应用程序和分析的布局。监视软件层的健康状况。将原始传感器数据或表达式的结果存储在本



1. 一种用于使用边缘计算改进数据的处理的方法,所述方法包括:

在耦合在本地网络的第一物理传感器和远程网络之间的边缘计算平台处,从所述第一物理传感器接收第一传感器数据流,其中,所述第一传感器数据流包括与正被监视的状况有关的数据;

在所述边缘计算平台的数据总线上产生与所述第一传感器数据流相对应的第一流数据;

在所述边缘计算平台的分析引擎处,实时处理来自所述数据总线的所述第一流数据,而无需先将所述第一传感器数据流传输到所述远程网络,其中,该处理包括:对所述第一流数据执行一个或多个表达语言的分析表达式、在包括对所述状况的指示的所述第一流数据中识别模式的存在、并生成关于所述状况的智能信息;并且

在所述边缘计算平台上执行应用程序,其中,所述应用程序基于关于所述状况的所述智能信息来确定是否执行以下项中的至少一项:将所述智能信息中的至少一部分发送到所述远程网络以进行附加处理或在所述本地网络中采取预定动作来影响正被监视的所述状况而无需等待所述附加处理,

其中,在所述边缘计算平台的分析引擎处的处理包括:使用所述分析引擎的表达语言来指定包括一个或多个分析表达式的第一虚拟传感器,其中,该虚拟传感器包括:从所述数据总线接收所述第一传感器数据流的第一输入、将第二传感器数据流输出在所述数据总线上的第一输出,并且所述第二传感器数据流包括根据第一函数从正被操作的所述第一传感器数据流得到的流数据;

执行所述表达语言的一个或多个分析表达式以在所述第二传感器数据流中检测模式的存在,其中,当在所述第二传感器数据流中检测到模式的存在时,出现对所述状况的指示;并且

当出现对所述状况的指示时,在所述数据总线上生成有关所述状况的智能信息。

2. 根据权利要求1所述的方法,包括:

将包括基于所述第一传感器数据流和第二传感器数据流汇集的数据的智能信息发送到所述远程网络;并且

在所述远程网络处,使用所汇集的数据执行预测性维护。

3. 根据权利要求1所述的方法,其中,所述边缘计算平台被嵌入在所述本地网络的网络设备中。

4. 根据权利要求1所述的方法,其中,所述边缘计算平台包括所述本地网络的网关。

5. 根据权利要求1所述的方法,包括:

将包括基于所述第一传感器数据流和第二传感器数据流汇集的数据的智能信息发送到所述远程网络;并且

在所述远程网络处,使用机器学习来处理所汇集的数据。

6. 根据权利要求1所述的方法,包括:

将包括基于所述第一传感器数据流和第二传感器数据流汇集的数据的智能信息发送到所述远程网络;并且

在所述远程网络处,对所汇集的数据执行远程监视。

7. 根据权利要求1所述的方法,包括:

将包括基于所述第一传感器数据流和第二传感器数据流汇集的数据的智能信息发送到所述远程网络;并且

在所述远程网络处,使用所汇集的数据来执行操作智能。

8. 根据权利要求1所述的方法,包括:

实时汇集所述第一传感器数据流和第二传感器数据流以获得所汇集的数据,并将所汇集的数据发送到所述远程网络以进行附加处理。

9. 根据权利要求1所述的方法,包括:

在所述边缘计算平台的数据摄取代理处,接收所述第一传感器数据流;并且

所述数据摄取代理从所述第一传感器数据流产生所摄取的第一流数据。

10. 根据权利要求9所述的方法,包括:

在所述边缘计算平台的数据丰富组件处,接收所摄取的第一流数据;并且

所述数据丰富组件实时地丰富所摄取的第一流数据并产生经丰富的所摄取的第一流数据,

其中,所述丰富包括以下各项中一项或多项:解码、利用元数据进行补充、以及标准化所摄取的流数据。

11. 根据权利要求10所述的方法,包括:

在所述分析引擎处,接收所述经丰富的所摄取的第一流数据,

其中,所述经丰富的所摄取的第一流数据包括由所述分析引擎处理的所述第一流数据。

12. 根据权利要求11所述的方法,其中,由所述分析引擎执行的所述一个或多个分析表达式指定一个或多个函数,每个函数包括以下各项中的至少一项:变换、模式检测、动态校准、信号处理、数学表达、数据压缩、数据分析、数据汇集、规则应用、警报或服务调用,以及在所述分析引擎处处理所述第一流数据包括将所述一个或多个函数应用于所述第一流数据,并生成所述智能信息。

13. 根据权利要求1所述的方法,其中,所述预定动作包括以下各项中的至少一项:生成警报或改变所述本地网络中的操作。

14. 根据权利要求1所述的方法,其中,所述边缘计算平台通过第一类型的第一网络连接耦合到所述本地网络,并且通过第二类型的第二网络连接耦合到所述远程网络,所述第二类型不同于所述第一类型,并且所述第二类型包括较低带宽的连接。

用于改进对分布式网络中的传感器流数据的处理的方法

[0001] 本申请是申请日为2016年8月29日、申请号为201680050011.8并且发明名称为“用于改进对分布式网络中的传感器流数据的处理的方法”的分案申请。

[0002] 相关申请的交叉引用

[0003] 本专利申请要求于2015年8月27日递交的美国专利申请62/210,981的权益,其通过引用与该本申请中引用的所有其他参考文献一起被结合于此。于2016年3月23日递交的美国专利申请62/312,106、62/312,187、62/312,223、和62/312,255通过引用被结合于此。

背景技术

[0004] 本发明涉及计算领域,并且更具体地,涉及用于处理由工业机器生成的大量数据的边缘计算。

[0005] 传统的企业软件应用程序托管依赖数据中心或“云”基础设施以开拓规模经济和系统效率。但是,这些数据中心可能与企业进行大部分业务操作的物理操作点(例如,工厂、仓库、零售店等)相距任意距离。工业物联网(IIoT)指设备或用例的集合,其依赖于利用以非常高的频率跟踪事件的传感器的物理操作仪器。

[0006] 许多领域中的工业机器都归属于这种物联网(IoT),这些领域涵盖了制造业、石油和天然气、采矿业、运输业、电力和水利、可再生能源、医疗保健、零售业、智能建筑、智能城市和互连车辆。虽然云计算已成功地实现,但是尚存在诸多缺陷:将所有数据发送至云存储装置是不实际的,因为连接不可能总存在、带宽不足、或者即使存在足够的带宽,成本也会过高。即使连接性、带宽和成本都不是问题,也不可能进行实时决策和预测性维护,这样可能会导致机器的严重损坏。

[0007] 因此,需要包括改进的边缘分析的改进的计算系统、架构、和技术,以便处理由工业机器生成的大量数据。

发明内容

[0008] 一种用于实现边缘处智能的方法。其特征包括:由在网关设备或嵌入式系统上托管的软件层中的传感器数据触发。软件层被连接到局域网。服务、应用程序和数据处理引擎的存储库被使得能够由软件层访问。通过由软件层提供的表达式语言将传感器数据与对特定状况的发生的语义描述进行匹配。通过持续执行表达式来自动发现模式事件。智能地跨越网关设备和网络上的由软件层管理的嵌入式系统对服务和应用程序进行编排以链接应用程序和分析表达式。基于资源可用性来优化应用程序和分析的布局。监视软件层的健康状况。将原始传感器数据或表达式的结果存储在本地时间序列数据库或云存储中。服务和组件可以被容器化以确保在任何网关环境中的平稳运行。

[0009] 边缘智能在物联网(IoT)数据源处被启用。系统为实时边缘分析和应用程序提供了对IoT设备传感器数据的丰富的接入(流模式或批处理模式,或两者兼而有之)。该系统包括高效且高表达性的计算机语言,其用于通过在低内存占用机器中操作的高性能分析引擎来执行分析函数和表达式。该系统允许将汇集数据发布到云以进行进一步的机器学习。该

系统包括用于开发边缘应用程序的软件开发包。基于云的管理控制台允许对边缘部署、配置、应用程序和分析表达式进行管理。

[0010] 一种边缘基础设施和平台的具体实现来自于FogHorn Systems, Inc. (FogHorn公司)。FogHorn公司的网站www.foghorn-systems.com、出版物(包括白皮书、用户指南、教程、视频、以及其他出版物)以及其他有关FogHorn技术和产品的出版物均通过引用被结合于此。

[0011] FogHorn提供了一种用以实现针对工业和商业物联网 (IoT) 数据的边缘智能的平台。数以百亿计的工业和商业物联网设备生成的数据量足以压垮整个互联网。FogHorn平台处理、分析并响应源于网络边缘的物联网数据。FogHorn的“智能边缘”软件平台实现了前所未有的自动化程度、作业效率、成本节约等等。

[0012] 工业物联网 (IIoT) 由互联的工业和商业设备组成, 例如传感器、机械装置和计算机。IIoT的目标是在分布式企业中实现更优的设备控制、数据管理、机器自动化和作业效率。公司可以在边缘处应用雾计算, 以利用实时分析和自动响应捕获IIoT绿地机会, 同时还利用云计算进行整个系统的管理及优化。如果增加额外的计算资源是不可行的, 则FogHorn边缘计算平台还被设计为在现有的可编程逻辑控制器 (PLC) (例如, 棕地机会) 中运行。棕地是指为了解决信息技术 (IT) 问题领域来实现新系统, 但同时也考虑已建立的系统。新的软件架构同时将现有和正在运行的软件纳入了考虑范围。

[0013] 边缘智能平台是一种基于雾计算概念的基于软件的解决方案, 这种解决方案将数据处理和分析扩展到更靠近IIoT设备所在的边缘。保持靠近边缘设备而不是将所有数据发送到远处的集中式云, 最大限度地减少了延迟, 从而实现最优性能、更快的响应时间、以及更有效的维护和操作策略。这种方式还显著降低了整体带宽需求和管理广泛分布式网络的成本。

[0014] 对边缘处的IIoT操作的关注降低了整体带宽需求, 并能够实现立即对时间敏感的状况进行自动响应。随着工业界数十亿台新的IIoT设备的增加, 每天都会产生大量PB级数据。将所有这些数据发送到云不仅成本高昂, 而且还伴随着更大的安全隐患。在边缘处进行操作可确保更快的响应时间、更低的风险、以及更低的总成本。

[0015] 在实施方式中, 边缘计算平台系统包括: 若干传感器数据流以及物理地布置在传感器和通信网络之间的软件层, 该软件层被配置为基于从传感器接收的原始数据来执行计算。从数据处理层和通信网络的角度来看, 软件层可以包括可编程虚拟传感器。

[0016] 该软件可包括数据处理层, 该数据处理层包括用于进行流处理的复杂事件处理引擎和表达式语言。数据处理层可以包括数据丰富层和软件开发包, 用以与应用程序进行交互和进行应用程序的开发。该软件可以包括数据发布层, 用以发布传感器数据流或从本地时间序列数据库或所选的云存储供应商中的至少一者中的表达式导出的数据或元数据。

[0017] 系统可以包括应用程序库, 其中, 软件层被配置为存储、索取和部署来自应用程序库的应用程序。应用程序被配置为在硬件层上处理传感器数据。应用程序库被配置为基于硬件层上可用的资源来定制应用程序容器。

[0018] 在实施方式中, 传感器系统包括: 若干传感器; 以及物理地布置在传感器和通信网络之间的硬件层, 该硬件层被配置为基于从传感器接收的原始数据执行计算。从通信网络的角度来看, 硬件层可以被配置为可编程为虚拟传感器。虚拟传感器可包括由硬件层中的

编程指定的应用程序接口 (API)。

[0019] 系统可以包括应用程序库,其中,硬件层被配置为从应用程序库接收应用程序。应用程序被配置为处理硬件层上的传感器数据。应用程序库被配置为基于硬件层上可用的资源来定制应用程序容器。

[0020] 在实施方式中,一种方法包括:在边缘平台系统的数据摄取代理处通过第一网络连接类型接收传感器数据流数据;将所摄取的流数据从数据摄取代理传送到系统的数据总线;在被连接到数据总线(例如,订阅来自数据总线的数据)的分析引擎处处理所摄取的流数据,其中,该处理包括执行表达式语言中提供的分析表达式以从所摄取的流数据生成智能信息;并将智能信息发布在数据总线上。该方法可以包括:通过数据丰富组件实时丰富所摄取的数据,并使所摄取的数据能够在数据总线上可用。丰富可以包括但不限于:数据解码、元数据修饰、数据标准化等。

[0021] 该方法可包括:在数据发布者组件处从数据总线接收智能信息;以及通过数据发布者组件将智能信息存储在时间序列数据库中。此外,该方法可以包括通过数据发布者组件经由第二网络连接类型传送至少部分智能信息并将其存储在云存储装置中,其中,第二网络连接类型不同于第一网络连接类型,且第二网络连接类型与第一网络连接类型相比具有对传感器数据流数据的更低带宽的连接。

[0022] 例如,第一网络连接类型可以使用工业IoT协议,例如MQTT、OPC统一架构、或Modbus协议、以及自定义协议。第二网络连接类型可以使用互联网协议,例如TCP/IP、超文本传输协议(HTTP)、WebSockets、WebHDFS或Apache Kafka、或这些的任意组合。第二网络连接类型也可使用上述针对第一网络连接类型的协议。

[0023] 数据摄取代理可以通过第一网络连接类型经由推送协议来访问传感器数据流数据。数据摄取代理也可以通过第一网络连接类型经由拉取协议来访问传感器数据流数据。

[0024] 该方法可包括:提供允许用户开发用于边缘平台系统的应用程序的软件开发包;以及通过软件开发包,开发可访问并处理可从数据总线上获取的智能信息的第一容器化应用程序。该方法可包括:在数据发布者组件处从数据总线接收智能信息;通过数据发布者组件将智能信息存储在时间序列数据库中;以及通过软件开发包,开发可访问和处理存储在时间序列数据库中的智能信息的第二容器化应用程序。

[0025] 该方法可以包括:提供边缘平台系统的应用程序商店,其中,第一容器化应用程序和第二容器化应用程序可通过应用程序商店供其他用户进行访问。

[0026] 第一容器化应用程序可在边缘平台系统内运行,其中,第一容器化应用程序具有到数据总线的直接连接。第一容器化应用程序(无需更改)也可在云环境中运行,其中,第一容器化应用程序不具有到数据总线的直接连接,而是通过第二网络连接类型来访问智能信息。

[0027] 该方法可以包括:使用表达式语言来创建第一虚拟传感器,该第一虚拟传感器具有连接到第一物理传感器的第一输入和作为该第一输入的第一函数的第一输出。该第一函数是表达式语言中指定的。第一虚拟传感器从第一物理传感器接收第一流数据。在第一输出处,第一虚拟传感器根据第一函数对第一流数据进行操作而输出第二流数据。智能信息包括该第二流数据。

[0028] 该方法可以包括:使用表达式语言来创建第一虚拟传感器,该第一虚拟传感器具

有连接到第一物理传感器的第一输入、连接到第二物理传感器的第二输入以及作为第一输入和第二输入的第一函数的第一输出。该第一函数是表达式语言中指定的。第一虚拟传感器从第一物理传感器接收第一流数据以及从第二物理传感器接收第二流数据。在第一输出处,第一虚拟传感器根据第一函数对第一流数据以及第二流数据进行操作而输出第三流数据。智能信息包括该第三流数据。

[0029] 在实施方式中,一种方法包括:在边缘平台系统的数据摄取代理处通过第一网络连接类型接收传感器数据流数据;将所摄取的流数据从数据摄取代理传送到系统的数据总线;在连接到数据总线的分析引擎处处理所摄取的流数据,其中,处理包括执行在表达式语言中提供的分析表达式以从所摄取的流数据中生成智能信息;以及提供第一虚拟传感器,该第一虚拟传感器具有连接到第一物理传感器的输入和作为输入的第一函数的输出,其中,该第一函数是表达式语言中指定的,第一虚拟传感器从第一物理传感器接受第一流数据,并且在输出处,第一虚拟传感器根据第一函数对第一流数据进行操作而输出第二流数据,并且智能信息包括该第二流数据。

[0030] 该方法还包括:提供第二虚拟传感器,该第二虚拟传感器具有连接到第二物理传感器的第一输入、连接到第一虚拟传感器的输出的第二输入、以及作为该第一输入和第二输入的第二函数的输出,其中,该第二函数是表达式语言中指定的,第二虚拟传感器从第二物理传感器接收第三流数据并从第一虚拟传感器接收第二流数据,并且在第一输出处,第二虚拟传感器根据第二函数对第二流数据和第三流数据进行操作而输出第四流数据,并且智能信息包括该第四流数据;将智能信息发布在数据总线上;在数据发布者组件处接收来自数据总线的智能信息;以及通过数据发布者组件将智能信息存储在时间序列数据库中。

[0031] 在实施方式中,一种方法包括:在边缘平台的表达式语言中指定第一虚拟传感器,其中,第一虚拟传感器包括作为输入的第一函数的输出,其中,第一输入被连接到第一物理传感器的第一流,并且第一虚拟传感器输出第二流;以及允许在表达式语言中指定第二虚拟传感器,其中,第二虚拟传感器包括作为第一输入和第二输入的第二函数的输出,其中,第一输入被连接到第二物理传感器的第二流,且第二输入被连接到来自第一虚拟传感器的第二流。

[0032] 在考虑了下文的详细说明和附图后,本发明的其他目的、特征、和优点将变得明显,其中,相同的附图标记在整个附图中表示相似的特征。

附图说明

[0033] 图1示出了客户端-服务器系统和网络的框图。

[0034] 图2示出了客户端或服务器的更详细的示图。

[0035] 图3示出了计算机系统的系统框图。

[0036] 图4是位于传感器流与云之间的边缘计算平台的框图。

[0037] 图5示出了包括边缘分析的边缘计算平台的更详细的框图。

[0038] 图6示出了边缘基础设施与云基础设施之间的操作流程。

[0039] 图7示出了使用物理传感器以经由传感器表达式语言引擎创建一些虚拟传感器的示例。

具体实施方式

[0040] 图1是包括本发明的实施例的分布式计算机网络100的简化框图。计算机网络100包括若干客户端系统113、116、和119以及经由多个通信链路128被耦合到通信网络124的服务器系统122。通信网络124提供允许分布式网络100的各个组件彼此进行通信和信息交换的机制。

[0041] 通信网络124本身可以由许多互连的计算机系统和通信链路组成。通信链路128可以是硬连线链路、光链路、卫星或其他无线通信链路、波传播链路、或用于信息的通信的任意其他机制。通信链路128可以是DSL、电缆、以太网或其他硬连线链路、无源或有源光链路、3G、3.5G、4G和其他移动、卫星或其他无线通信链路、波传播链路或用于信息的通信的任意其他机制。

[0042] 各种通信协议可以用于促进图1中所示的各种系统之间的通信。这些通信协议可以包括VLAN、MPLS、TCP/IP、网络隧道、HTTP协议、无线应用协议(WAP)、供应商特定协议、自定义协议等。虽然在一个实施例中,通信网络124是互联网,但是在其他实施例中,通信网络124可以是包括局域网(LAN)、广域网(WAN)、无线网络、内联网、专用网络、公共网络、交换网络、以及这些的组合等的任何合适的通信网络。

[0043] 图1中的分布式计算机网络100仅仅是对包含本发明的实施例的说明,并不限制如权利要求中所述的本发明的范围。本领域的普通技术人员可认识到其他变化、修改和替代方案。例如,不止一个服务器系统122可以连接到通信网络124。作为另一示例,多个客户端系统113、116、和119可经由访问提供商(未示出)或者经由另一些服务器系统被耦合至通信网络124。

[0044] 客户端系统113、116和119通常从提供信息的服务器系统请求信息。出于这个原因,服务器系统通常具有比客户端系统更多的计算和存储容量。但是,特定的计算机系统既可充当客户端又可充当服务器,这取决于计算机系统是请求还是提供信息。另外,虽然已使用了客户端-服务器环境来描述本发明的各个方面,但显而易见的是,本发明也可以体现在独立的计算机系统中。

[0045] 服务器122负责从客户端系统113、116、和119接收信息请求、执行满足请求所需的处理、以及将与请求对应的结果转发回请求客户端系统。满足请求所需的处理可以由服务器系统122执行,或者可替代地委托给连接到通信网络124的其他服务器来执行。

[0046] 客户端系统113、116、和119使用户能够访问和查询由服务器系统122存储的信息。在特定实施例中,客户端系统可以作为独立应用程序运行,例如,桌面应用程序或移动智能电话或平板设备应用程序。在另一实施例中,在客户端系统上运行的“网页浏览器”应用程序使用户能够选择、访问、索取或查询由服务器系统122存储的信息。网页浏览器的示例包括由微软公司(Microsoft Corporation)提供的Internet Explorer(IE)浏览器程序、由Mozilla提供的火狐(Firefox)浏览器、由谷歌(Google)提供的Chrome浏览器、由苹果公司(Apple)提供的Safari浏览器等。

[0047] 在客户端-服务器环境中,一些资源(例如,文件、音乐、视频、或数据)被存储在客户端,而另一些资源被存储在网络中的其他地方(例如,服务器)或从这些地方被传送,并且可以通过网络(例如,互联网)被访问。因此,用户的数据可以存储在网络或“云”中。例如,用户可以在客户端设备上处理远程存储在云(例如,服务器)上的文档。客户端设备上的数据

可以与云同步。

[0048] 图2示出了本发明的示例性客户端或服务器系统。在实施例中,用户通过计算机工作站系统与系统接合,如图2所示。图2示出了计算机系统201,包括监视器203、屏幕205、外壳207(也可以被称为系统单元、机柜、或机箱)、键盘或其他人力输入设备209、以及鼠标或其他定点设备211。鼠标211可具有一个或多个按钮,例如鼠标按钮213。

[0049] 应当理解的是,本发明不限于具有特定的规格(例如,台式计算机规格)的任何计算设备,而是可以包括所有类型的各类规格的计算设备。用户可以与包括智能电话、个人计算机、膝上型设备、电子平板设备、全球定位系统(GPS)接收器、便携式媒体播放器、个人数字助理(PDA)、其他网络接入设备的任何计算设备和能够接收或传输数据的其他处理设备接合。

[0050] 例如,在具体实施方式中,客户端设备可以是智能手机或平板设备,例如,苹果iPhone(例如,苹果iPhone 6)、苹果iPad(例如,苹果iPad或苹果iPad mini)、苹果iPod(例如,苹果iPod Touch)、三星Galaxy产品(例如,Galaxy S系列产品或Galaxy Note系列产品)、谷歌Nexus设备(例如,Google Nexus 6、Google Nexus 7或Google Nexus 9)以及微软设备(例如,微软Surface平板设备)。通常,智能电话包括可通过触摸屏显示器访问的电话部分(以及相关关联的无线电)和计算机部分。

[0051] 存在用于存储电话部分(例如,联系人和电话号码)和计算机部分(例如,包括浏览器、图片、游戏、视频、和音乐的应用程序)的数据的非易失性存储器。智能电话通常包括用于拍摄照片和视频的相机(例如,前置摄像头或后置摄像头,或兼有这两者)。例如,智能电话或平板设备可用于拍摄可被流传送至一个或多个其他设备的实时视频。

[0052] 外壳207容纳常见的计算机组件,其中一些未被示出,例如处理器、存储器、大容量存储设备217等。大容量存储设备217可以包括大容量磁盘驱动器、软盘、磁盘、光盘、磁光盘、固定盘片、硬盘、CD-ROM、可记录CD、DVD、可记录DVD(例如,DVD-R、DVD+R、DVD-RW、DVD+RW、HD-DVD、或蓝光光盘)、闪存和其他非易失性固态存储(例如,USB闪存驱动器或固态驱动器(SSD))、电池备份的易失性存储器、磁带存储装置、读卡器、和其他类似介质,以及这些的组合。

[0053] 本发明的计算机实现的或计算机可执行的版本或计算机程序产品可使用计算机可读介质存储来体现、可被存储于其上或可与其相关联。计算机可读介质可以包括参与向一个或多个处理器提供指令以供运行的任何介质。此类介质可以采取许多形式,包括但不限于:非易失性介质、易失性介质、和传输介质。非易失性介质包括例如闪存、或光盘、或磁盘。易失性介质包括静态或动态存储器,例如高速缓冲存储器或RAM。传输介质包括同轴电缆、铜线、光纤线路和布置在总线中的导线。传输介质还可以采取电磁波、射频波、声波或光波的形式,例如那些在无线电波和红外数据通信期间产生的波。

[0054] 例如,本发明的软件的二进制的、机器可执行版本可被存储或驻留在RAM或高速缓冲存储器中,或者被存储或驻留在大容量存储设备217上。本发明的软件的源代码也可被存储或驻留在大容量存储设备217(例如,硬盘、磁盘、磁带、或CD-ROM)上。作为又一示例,本发明的代码可以通过导线、无线电波或通过诸如互联网之类的网络来发送。

[0055] 图3示出了用于执行本发明的软件的计算机系统201的系统框图。如图2中所示,计算机系统201包括监视器203、键盘209、和大容量存储设备217。计算机系统201还包括诸如

中央处理器302、系统存储器304、输入/输出(I/O)控制器306、显示适配器308、串行或通用串行总线(USB)端口312、网络接口318、和扬声器320之类的子系统。本发明还可以与具有更多或更少子系统的计算机系统一起使用。例如,计算机系统可以包括不止一个处理器302(即,多处理器系统)或者系统可以包括高速缓冲存储器。

[0056] 诸如322的箭头表示计算机系统201的系统总线架构。但是,这些箭头是对用于链接子系统的任何互连方案的说明。例如,扬声器320可以通过端口连接到其他子系统或者具有到中央处理器302的内部直接连接。处理器可以包括多个处理器或多核处理器,其可允许信息的并行处理。图2中所示的计算机系统201只是适用于与本发明一起使用的计算机系统的示例。适用于与本发明一起使用的子系统的其他配置对于本领域的普通技术人员而言将是显而易见的。

[0057] 计算机软件产品可以用任何适当的编程语言来编写,例如C、C++、C#、Pascal、Fortran、Perl、Matlab(来自MathWorks, www.mathworks.com)、SAS、SPSS、JavaScript、AJAX、Java、Python、Erlang、和Ruby on Rails。计算机软件产品可以是具有数据输入和数据显示模块的独立应用程序。可选地,计算机软件产品可以是可被实例化为分布式对象的类。计算机软件产品也可以是组件软件,例如Java Beans(来自甲骨文公司(Oracle Corporation))或Enterprise Java Beans(来自甲骨文公司的EJB)。

[0058] 系统的操作系统可以是Microsoft **Windows®**系列系统(例如,Windows 95、98、Me、Windows NT、Windows 2000、Windows XP、Windows XP x64版本、Windows Vista、Windows 7、Windows 8、Windows 10、Windows CE、Windows Mobile、Windows RT)、塞班操作系统(Symbian OS)、Tizen、Linux、HP-UX、UNIX、Sun OS、Solaris、Mac OS X、苹果iOS、Android、Alpha OS、AIX、IRIX32、或IRIX64中的一者。也可以使用其他操作系统。Microsoft **Windows®**是微软公司的商标。

[0059] 此外,计算机可以连接到网络并且可以使用该网络与其他计算机接合。网络可以是内联网、因特网或互联网等等。网络可以是(例如使用铜线的)有线网络、电话网络、分组网络、(例如使用光纤的)光网络、或无线网络、或者这些的任意组合。例如,可以使用采用诸如Wi-Fi(IEEE标准802.11、802.11a、802.11b、802.11e、802.11g、802.11i、802.11n、802.11ac、和802.11ad,仅列举几个示例)、近场通信(NFC)、射频识别(RFID)、移动或蜂窝无线(例如2G、3G、4G、3GPP LTE、WiMAX、LTE、LTE升级版、Flash-OFDM、HIPERMAN、iBurst、EDGE Evolution、UMTS、UMTS-TDD、1xRDD、和EV-DO)之类的协议的无线网络在计算机和本发明的系统的组件(或步骤)之间传递数据和其他信息。例如,来自计算机的信号可以至少部分地、无线地传输到组件或其他计算机。

[0060] 在实施例中,利用在计算机工作站系统上运行的网络浏览器,用户通过诸如互联网之类的网络访问万维网(WWW)上的系统。网页浏览器用于下载各种格式的网页或其他内容,包括HTML、XML、文本、PDF、和postscript,并可用于将信息上传到系统的其他部分。网络浏览器可以使用统一资源标识符(URL)来识别网络上的资源并在网络上传输文件时使用超文本传输协议(HTTP)。

[0061] 在其他实施方式中,用户通过本地应用程序和非本地应用程序中的任一者或这两者来访问系统。本地应用程序本地安装在特定的计算系统上,并且特定于该计算系统的操作系统或一个或多个硬件设备或这些的组合。这些应用程序(有时也被称为“应用”)可以经

由直接互联网升级修补机制或通过应用程序商店(例如,苹果iTunes和应用商店、Google Play商店、Windows Phone商店、和黑莓App World商店)(例如周期性地)进行更新。

[0062] 系统可以运行在与平台无关的非本地应用程序上。例如,客户端可以使用与一个或多个服务器连接的网络而通过该一个或多个服务器的网页应用程序访问系统,并将该网页应用程序加载到网页浏览器中。例如,网页应用程序可以由网络浏览器通过互联网从应用程序服务器下载。非本地应用程序也可以从其他来源获得,例如盘片。

[0063] 图4示出了通常运行在位于传感器409和云412之间的边缘网关或等同设备上的边缘计算平台406的框图。边缘计算平台促使得到对于管理和优化工业机器以及其他工业物联网而言重要的边缘智能。边缘网关的组件包括以下各项:摄取421、丰富425、复杂事件处理(CEP)引擎429、应用程序432、通过表达式语言进行的分析435以及传输438。云可以包括边缘配置和编排443以及云和边缘分析及应用可移植性446。

[0064] 如上所述,一种边缘计算平台的具体实现来自于FogHorn。FogHorn是迅速崛起的“边缘智能”领域中的领头雁。FogHorn的突破性的解决方案通过将高性能处理、分析和异质应用托管在更接近控制系统和物理传感器的位置,可实现针对闭环设备优化的边缘智能。这为工业客户在制造业、石油和天然气、电力和水利、运输业、采矿业、可再生能源、智慧城市等领域带来了大量数据并实现了实时现场处理。FogHorn技术受到了全球领先的工业互联网创新者和在云计算、高性能边缘网关、以及物联网系统集成方面的主要参与者的青睐。

[0065] Foghorn提供:在流和批处理模式下针对边缘应用程序的丰富的物联网设备和传感器数据接入。用于执行分析功能的高效且高表达性的DSL。可以在小型机器上运行的强大的小型化分析引擎。用于将所汇集的数据发送到云以进行进一步机器学习的发布功能。用于开发边缘应用的SDK(多语种)。用于管理配置、应用程序和分析表达式的边缘部署的管理控制台。

[0066] FogHorn提供了一种高效且具有高扩展性的边缘分析平台,这种边缘分析平台可实现对来自工业机器的传感器数据的实时现场流处理。FogHorn软件堆栈是在边缘和云上运行的服务的组合。

[0067] “边缘”解决方案可以支持将传感器数据摄取到本地存储库中,可以选择将未被处理的数据发布到云环境以进行离线分析。但是,许多工业环境和设备缺少互联网连接,导致数据无法被使用。但即使在存在互联网连接的情况下,生成的大量数据也很容易超过可用带宽,或者发送到云的成本太高。此外,当数据被上传到云时、在数据中心处理时、以及当结果被传送回边缘时,采取任何行动都可能为时已晚。

[0068] FogHorn解决方案通过提供高度小型化的复杂事件处理(CEP)引擎(也被称为分析引擎)和用于表达对大量传入传感器数据流应用的规则的强大且高表达性的领域特定语言(DSL)来解决这个问题。然后,可以立即使用这些表达式的输出来防止代价高昂的机器故障或停机时间,并实时提高工业操作和流程的效率和安全性。

[0069] FogHorn平台包括:在低占用空间环境以及高吞吐量或网关环境中运行的能力。可以处理传入的流传感器数据的具有高扩展性的高性能CEP引擎。在边缘上具有丰富的数据访问的异质应用的开发和部署。跨越云和边缘的应用程序移动性。先进的机器学习(ML)和云与边缘之间的模型传递。FogHorn开箱即可支持主要的工业数据摄取协议(例如,OPC-UA、Modbus、MQTT、DDS等)以及其他数据传输协议。此外,用户可以轻松地将自定义协议适配器

插入FogHorn的数据摄取层。

[0070] FogHorn边缘服务在IIoT设备所在网络的边缘进行操作。边缘软件堆栈负责将数据从传感器和工业设备摄取到高速数据总线上,然后对流数据执行用户定义的分析表达式以获取剖析(insight)并对设备进行优化。这些分析表达式由FogHorn的具有高扩展性、低占用空间的复杂事件处理(CEP)引擎来执行。

[0071] FogHorn边缘服务还包括用于基于时间的传感器数据查询的本地时间序列数据库和用于开发可以在流和批处理模式下使用数据的应用程序的多语种SDK。可选地,这些数据也可被发布到客户选择的云存储目的地。

[0072] FogHorn平台还包括在云或自行管理(on-premise)环境中运行以远程配置和管理边缘的服务。FogHorn的云服务包括管理UI,用于开发和部署分析表达式、使用被称为Docker(www.docker.com)的应用程序将应用程序部署到边缘、以及用于管理服务与客户身份访问管理和持久性解决方案的集成。该平台还将能够将云中开发的机器学习模型转换为可在边缘执行的传感器表达式。

[0073] FogHorn通过将边缘智能计算平台直接嵌入到小型边缘设备中为工业物联网带来突破性的发展。该软件的开销非常低,因此可以将其嵌入到各种边缘设备和高度受限的环境中。

[0074] 在网关版和微型版中,FogHorn软件可实现尽可能靠近遍布工业世界的控制系统和物理传感器基础架构的高性能边缘处理、优化分析和异质应用程序。保持靠近边缘设备而不是将所有数据发送到远处的集中的云端,最大限度地减少了延迟,从而实现最优的性能、更快的响应时间、以及更有效的维护和操作策略。这样还大大地降低了整体带宽需求和管理广泛分布式网络的成本。

[0075] 关于FogHorn网关版本。FogHorn网关版本是一款适用于各行各业的工业物联网用例的综合型雾计算套装软件。该版本专为具有多台工业机器或设备的大规模环境而设计,实现了用户可配置的传感器数据摄取和分析表达式,并支持高级应用程序开发和部署。

[0076] 关于FogHorn微型版。FogHorn微型版将雾计算的强大功能引入小型边缘网关和其他IoT机器。在微型版中提供了与网关版中包含的CEP分析引擎和高表达性DSL相同的CEP分析引擎和高表达性DSL。此版本非常适合于实现嵌入式系统或任何内存受限设备中的高级边缘分析。

[0077] 作为示例,应用程序应用实时数据监视和分析、预测性维护调度、以及自动流重定向来防止由于气蚀现象的发生而对泵造成代价很大的损坏。另一示例是风能管理系统,其使用FogHorn边缘智能软件来最大限度地发电、延长设备使用寿命、以及将历史分析应用于进行准确的能源预测。

[0078] 图5示出了边缘计算平台的更详细的框图。该平台具有三个逻辑层或区:数据摄取512、数据处理515、和数据发布518。数据摄取组件包括连接到生成数据的传感器或设备523的代理520。这些代理通过各个协议服务器经由一个或多个协议从传感器收集或摄取数据。代理可以是针对协议(例如,MQTT、OPC UA、Modbus、和DDS等)的客户端或署理(broker)。由传感器提供或输出的数据通常是二进制数据流。这些数据从传感器到代理的传输或传递可以通过推送或拉取方法进行。

[0079] 推送描述了由发送方(例如,传感器)发起针对给定事务的请求的通信类型。拉取

(或获得)描述了由接收方(例如,代理)发起针对信息传输的请求的通信类型。另一种通信技术是轮询,其中,接收方或代理定期查询或检查传感器是否有数据要发送。

[0080] MQTT(早前被称为MQ Telemetry Transport)是用于TCP/IP协议之上的ISO标准的基于发布-订阅的“轻量级”消息传递协议。其他协议包括高级消息队列协议、IETF约束应用协议、XMPP、和Web应用消息传递协议(WAMP)。

[0081] OPC统一架构(OPC UA)是由OPC基金会开发的针对互操作性的工业M2M通信协议。它是开放平台通信(OPC)的下一代。

[0082] Modbus是一种串行通信协议,这种串行通信协议最初是由Modicon(现为施耐德电气)于1979年发布,供结合其可编程逻辑控制器(PLC)来使用。Modbus由于其简单且鲁棒的特性已经成为针对所有意图和目的的标准通信协议。它现在是连接工业电子设备的常用手段。

[0083] 数据处理515包括连接到数据摄取层的代理520的数据总线532。数据总线是针对所有连接组件之间的数据和控制消息的中央骨干。组件订阅流经数据总线的数据和控制消息。分析引擎535是一个十分重要的组件。分析引擎基于以表达式语言538开发的分析表达式来执行传感器数据的分析。连接到数据总线的其他组件包括配置服务541、度量服务544、和边缘管理器547。数据总线还包括“解码器服务”,通过将原始二进制数据解码为可消耗数据格式(例如,JSON)并使用额外的必要且有用的元数据进行修饰,从而丰富来自传感器的传入数据。此外,丰富可以包括但不限于数据解码、元数据修饰、数据标准化等。

[0084] JSON(有时也被称为JavaScript Object Notation(JS对象标记))是一种开放标准格式,其使用人类可读的文本来发送由属性-值对构成的数据对象。JSON是用于异步浏览器或服务器通信(AJAX)或这两者的通用数据格式。JSON的替代方案是AJAX使用的XML。

[0085] 边缘管理器连接到云412,并且具体地,连接到云管理器552。云管理器被连接到同样位于云中的用于客户身份和访问管理(IAM)的代理服务器555和用户界面控制台558。还有可经由云被访问的应用程序561。身份和访问管理是使正确的个体能够在正确的时间并出于正确的原因访问正确的资源的安全和商业法则。

[0086] 在数据处理515内,软件开发包(SDK)564组件也连接到数据总线,从而允许可以部署在边缘网关上的应用程序567的创建。该软件开发包还连接到本地时间序列数据库以获取数据。应用程序可以使用容器技术(例如,Docker)被进行容器化。

[0087] Docker容器将软件包裹在完整的文件系统中,其中包含了运行软件所需的所有东西:代码、运行时间、系统工具、和系统库—可安装在服务器上的任何东西。这确保了软件始终相同地运行,而无论运行的环境如何。

[0088] 数据发布518包括被连接到云中的存储位置573的数据发布者570。此外,软件开发包564的应用程序567可以访问时间序列数据库576中的数据。时间序列数据库(TSDB)是一种针对对时间序列数据、按时间(例如,日期时间或日期时间范围)索引的数字数组进行的处理而被优化的软件系统。时间序列数据库通常是滚动式或循环式的缓冲区或队列,其中,当新的信息被添加到数据库时,最旧的信息会被删除。数据发布者570还连接到数据总线并订阅需要存储在本地时间序列数据库或云存储中的数据。

[0089] 图6示出边缘602与云基础设施之间的操作流程。一些特定的边缘基础设施在以上已进行了描述。数据从传感器606被收集。这些传感器可以用于工业、零售业、健康护理、或

医疗设备、或电力或通信应用或这些的任意组合。

[0090] 边缘基础架构包括软件平台609,其具有数据处理612、本地时间序列数据库615、云池618、分析复杂事件处理引擎(CEP) 621、分析实时流的领域特定语言(DSL) 624(例如,Foghorn的Vel语言)以及实时汇集与访问627。该平台可以包括虚拟传感器630,其在下面被更详细地描述。虚拟传感器提供丰富的实时数据访问。

[0091] 该平台可以通过一个或多个应用程序633被接入,例如应用或应用程序1、2和3,这些应用程序可以使用软件开发包或SDK来开发。这些应用可以是异质的(例如,用多种不同的语言开发的),它们利用复杂事件处理引擎621并执行机器学习。可以使用应用商店637来分发应用,该应用商店可以由边缘平台开发者或边缘平台的客户(可以被称为合作伙伴)提供。通过应用商店,用户可以下载应用程序并与他人分享。这些应用可以执行分析和应用程序639,包括机器学习、远程监控、预测性维护、或操作智能、或这些的任意组合。

[0092] 对于应用而言,存在边缘和云之间的动态应用移动性。例如,使用FogHorn软件开发包开发的应用程序可以部署在边缘或部署在云中,从而实现边缘和云之间的应用移动性。这些应用可以用作边缘的一部分或作为云的一部分。在实施方式中,由于应用被容器化,使得这一特征成为可能,因此应用可以独立于执行它们的平台操作。对于分析表达式而言同样如此。

[0093] 存在允许实现集成管制与管理640的数据应用,包括对在云中或在私人数据中心644处的数据的监视或存储。

[0094] 边缘应用基础设施可以向一些最棘手和最偏远的工业环境提供实时反馈和自动化控制系统。下面介绍一些特定的应用:

[0095] 制造业:从创建半导体到大型工业机器的组装,边缘智能平台通过实时监控和诊断、机器学习、和操作优化提高了制造产率和效率。边缘智能的即时性实现了制造过程中的自动反馈环路以及预测性维护,从而最大限度地延长设备和装配线的运行时间和使用寿命。

[0096] 石油和天然气:石油和天然气开采是一种高风险的技术驱动型操作,这种高风险的技术驱动型操作依靠实时现场智能提供主动监控和保护以防止设备故障和环境破坏。由于这些作业在非常偏远的地方进行,缺乏对集中式数据中心的可靠高速访问,因此边缘智能可提供高级分析的现场配送,并可实现所需的实时响应以确保最大产出与安全。

[0097] 采矿业:由于采矿业面对的是偏远地区的极端的环境条件,几乎不能接入互联网。因此,采矿作业越来越依赖边缘智能进行实时现场监控和诊断、报警管理、和预测性维护,以最大限度地提高安全性、作业效率并最大限度降低成本和缩短停机时间。

[0098] 运输业:作为工业互联网崛起的一部分,火车和轨道、巴士、飞机、和船舶都配备了会产生PB级数据量的新一代仪器和传感器,这便需要额外的智能进行分析和实时响应。边缘智能可以在本地处理这些数据,以实现实时资产监控和管理,从而最大限度降低作业风险和缩短停机时间。边缘智能还可用于监控和控制引擎空转时间、提前检测磨损状况、检测跟踪问题、检测潜在入侵者、减少排放、节省燃料、以及实现利润的最大化。

[0099] 电力和水利:发电厂的意外故障可能会对下游电网造成巨大破坏。同样地,如果配水设备和泵在没有警告的情况下失效,也会对下游电网造成巨大破坏。为了避免这样的情况的发生,边缘智能可以实现预测性维护和实时响应的主动性优势。边缘智能还实现了在

接近源(而非云)处对传感器数据进行摄取与分析,以便减少延迟和带宽成本。

[0100] 可再生能源:新太阳能、风能和水能是非常有前景的清洁能源。然而,不断变化的天气条件使预测并向电网输送可靠的电力供应面临重大挑战。边缘智能实现实时调整来最大限度地提高发电量并实现高级分析用于精确的能量预测和传输。

[0101] 医疗保健:在医疗保健行业中,新的诊断设备、患者监护工具、和操作技术提供了前所未有的患者护理水平,同时还提供了大量高敏感患者数据。通过在源头处理和分析更多数据,医疗设备可以优化供应链运作,并以更低的成本提高患者服务和隐私。

[0102] 零售业:为了与网上购物竞争,零售商必须降低成本,同时改进客户体验并提供网上商店无法提供的服务水平。边缘智能可通过提供实时全渠道个性化和供应链优化来丰富用户体验。边缘智能还支持面部识别等新技术来提供更高级的个性化和安全性。

[0103] 智能建筑:智能建筑技术的许多优点包括更低的能耗、更高的安全性、更高的居民舒适性和安全性、以及更好的对建筑资产和服务的利用。智能建筑并不会将大量建筑数据发送到云端进行分析,而是使用边缘智能来实现更灵敏的自动化,同时降低带宽成本和延迟。

[0104] 智能城市:将来自不同市政系统(例如,街道照明、交通信息、停车、公共安全和其他)的数据汇集以实现交互式管理和社区访问是智能城市举措的共同愿景。然而,生成的大量数据需要过多的带宽和针对基于云的系统进行的处理。边缘智能提供了一种更有效的解决方案,其中,数据处理和分析可被分配到传感器和数据源所在的边缘。

[0105] 互联车辆:互联车辆技术通过将车辆操作和控制扩展到驾驶员以外的外部网络和系统,为运输系统增添了全新的层面。边缘智能和雾计算将实现分布式的路边服务,如交通管制、车辆速度管理、收费、停车辅助等。

[0106] 针对IoT的工业AppStore

[0107] 物联网(IoT)需要将软件和分析应用于来自物理作业设施(例如,工厂、仓库、零售商店等)的传感器数据。该系统包括应用分发设施或应用商店(AppStore)以及针对这样的AppStore市场的方法,该方法解决了在被称作针对IoT的工业AppStore的软件平台装置中的工业IoT用例对独特可靠性、安全性、部署、和数据管理的需求。

[0108] 1.应用程序清单:工业物联网应用程序具有深入的部署上下文,其可以作为元数据被捕获并与应用程序一并进行存储。上下文包括边缘硬件规格、部署位置、关于兼容系统的信息、用于安全和隐私的数据访问清单、用于对给定部署中不可用的数据字段进行建模的模拟器、以及版本管理。边缘硬件包括物理传感器和外部通信信道之间的物理层。边缘计算平台可用于定义虚拟或软件可编程传感器。边缘计算平台可以用软件、硬件、固件、嵌入式硬件、独立硬件、专用硬件、或这些的任意组合来实现。

[0109] 2.清单匹配:基于来自边缘平台的给定请求,AppStore被配置为基于上面列出的参数将正确清单与部署场景匹配。

[0110] 3.部署操作:工业AppStore平台还执行特定于工业边缘部署场景的操作任务,包括数据一致性、应用程序状态切换、和安全证书管理。这些是应用程序从云或数据中心位置移动到作为应用程序容器对象的边缘的迁移过程的关键阶段。

[0111] 可编程软件定义的传感器或虚拟传感器

[0112] 物理传感器是电子换能器,其测量所在环境的一些特性作为模拟或数字测量结

果。模拟测量结果通常使用模数转换器被转换为数字量值。传感器数据可以根据需要(轮询式地)进行测量,也可以作为数据流以统一的速率被提供。典型的传感器规格是范围、准确度、分辨率、漂移、稳定性、以及其他属性。大多数测量系统和应用程序利用或通传传感器数据来直接进行处理、传输、或存储。

[0113] 系统具有“可编程软件定义的传感器”,也被称为虚拟传感器,其是使用分析表达式语言创建的基于软件的传感器。在实施方式中,分析表达式语言是FogHorn的分析表达式语言。这种表达式语言被称为Ve1,并在其他专利申请中有更详细的描述。Ve1语言的执行效率很高,可以在受限的低占用空间环境中以低执行延迟支持实时流分析。例如,系统的延迟可能约为10毫秒或更低。

[0114] 在实施方式中,可编程软件定义的传感器利用被称为“传感器表达式语言”或SXL的描述性应用程序接口(API)来创建。SXL语言的一个特定实施方式是来自FogHorn的Ve1。SXL传感器是通过此结构创建的SXL传感器,并提供从由多个源(包括物理传感器和SXL传感器)生成的处理数据导出的测量结果。在该申请中,SXL和Ve1可以互换使用。

[0115] SXL(例如,Ve1)传感器可以从这三个源中的任一者或其组合中导出:

[0116] 1. 单个传感器数据。

[0117] 1.1. 从单个物理传感器导出的虚拟或SXL传感器可以使用动态校准、信号处理、数学表达式、数据压缩、或数据分析或任意组合来转换传入的传感器数据。

[0118] 2. 多个物理传感器数据。

[0119] 2.1. 作为(使用上述方法)从多个异质物理传感器的转换而得到的虚拟或SXL传感器。

[0120] 3. 物理传感器数据和虚拟传感器数据的组合,可用于SXL传感器(例如,Ve1)设备的实现。

[0121] SXL(例如,Ve1)传感器是领域特定的并且是在考虑到特定应用的情况下创建的。SXL编程接口的特定实施方式使应用程序能够通过转换(例如,数学表达式)和汇集来定义数据分析。SXL(例如,Ve1)包括一组通常基于编程语言的数学运算符。SXL传感器通过执行SXL结构或程序在运行时对数据进行操作。

[0122] SXL传感器的创建。SXL传感器被设计为用于实时提供数据的软件装置。这需要在嵌入式计算硬件上实时执行使用SXL开发的应用程序,以便以应用所需的速率生成SXL传感器数据。该系统包括高效的执行引擎来完成这一任务。

[0123] SXL传感器的优点包括:

[0124] 1. 可编程性:SXL使得SXL传感器可编程为合成数据,以便匹配在数据质量、频率和信息等方面的特定应用需求。SXL传感器可以作为无线软件升级广泛分布,以便插入源自物理传感器和其他(例如,之前存在的)SXL传感器的数据中。因此,应用程序开发人员可以创建数字基础设施,这有助于独立于物理基础设施布局高效运行业务逻辑。

[0125] 2. 可维护性或透明度:SXL传感器在应用程序和物理传感器之间创建了数字抽象层,该数字抽象层将开发人员与物理基础设施由于物理传感器的升级和业务产生的变化隔离开来。

[0126] 3. 效率:SXL-传感器通过将来自物理传感器的原始数据转换成包含在其中的信息的精确表示来创建信息管理的效率。这种效率转化为对应用程序中的下游的IT资源(例如,

计算、网络、和存储)的有效利用。

[0127] 4.实时数据: SXL传感器提供从真实世界或物理传感器数据流计算出的实时传感器数据。这使得数据在最短的延迟时间内可供应用程序使用。

[0128] 实施方式: 系统基于SXL接口构建了一种可扩展的实时SXL传感器的实现。SXL包括由Java语言支持的运算符, 并很好地集成了物理传感器及其协议。

[0129] 该系统提供了一种创新的方法, 用于精确表达将对物理传感器数据执行的操作。这种描述式表达将数字抽象的定义与物理传感器上的实现分开。

[0130] 表A提供了物理传感器和使用SXL创建的SXL传感器的示例。

[0131] 表A

[0132]

```
Physical_sensors (物理传感器):
name (名称): "pump (泵)"
measures (测量):
-
name: "flow (流量)"
serde (序列化和反序列化): "CSV"
protocol (协议): "MQTT"
field_map (字段映射):
p1: "flow"
-
name: "outlet_pressure (出气压力)"
serde: "CSV"
protocol: "MQTT"
field_map:
p1: "outlet_pressure"
-
name: "temperature (温度)"
serde: "CSV"
```

[0133]

```

protocol: "MQTT"
field_map:
p1: "temperature"
-
name: "inlet_pressure (进气压力)"
serde: "CSV"
protocol: "MQTT"
field_map:
p1: "inlet_pressure"

virtual_sensors (虚拟传感器):
-
name: "pressure_differential (压力差)"
expression (表达式):
    "pressure_differential=inlet_pressure-outlet_pressure"
-
name: "temperature_kelvin (温度_开尔文)"
expression:
    "temperature_kelvin= (temperature +459.67) × (5.0/9.0) "
-
name: "vapor_pressure (蒸汽压力)"
expression:
    "vapor_pressure=exp (20.386 - (5132.0/temperature_kelvin) ) "

```

[0134] 在上面的示例中,使用四个物理传感器来创建一组SXL传感器。SXL传感器数据可以被发送到本地存储装置或应用,或发送到其他远程服务(例如,云)以进行进一步分析。

[0135] 作为具体示例,在表A中有一个名为“pressure_differential”的虚拟传感器。其中,变量pressure_differential作为输出。虚拟传感器的输入是inlet_pressure和outlet_pressure。该虚拟传感器基于函数inlet_pressure-outlet_pressure而输出pressure_differential。

[0136] 表A具有另一个名为“temperature_kelvin”的虚拟传感器。temperature_kelvin是函数 $(\text{temperature} + 459.67) \times (5.0/9.0)$ 的输出,其中temperature是传感器的输入。

[0137] 表A具有另一个名为“vapor_pressure”的虚拟传感器。输vapor_pressure是函数 $\exp(20.386 - (5132.0/\text{temperature_kelvin}))$ 的输出,其中,temperature_kelvin是传感器的输入。变量temperature_kelvin是来自temperature_kelvin传感器的输出。

[0138] 表B提供了使用FogHorn的Vel表达式语言来定义虚拟传感器的示例。

[0139] 表B

[0140]

temperature is stream (“ temperature ” , real) ; 温度是流
(“temperature” , 实数)

inlet_pressure is stream (“ inlet_pressure ” , real) ; 进气压力是流
(“inlet_pressure” , 实数)

outlet_pressure is stream (“ outlet_pressure ” , real) ; 出气压力是流
(“outlet_pressure” , 实数)

local_pressure_differential is local_stream (“local_pressure_differential” ,
real) ; 本地压力差是本地流 (“local_pressure_differential” , 实数)

pressure_differential is stream (“pressure_differential” , real) ; 压力差是
流 (“pressure_differential” , 实数)

cavitation_alert is stream (“ cavitation_alert ” , int) ; 气蚀警报是流
(“cavitation_alert” , 整数)

local_temperature_kelvin is local_stream (“ local_temperature_kelvin ” ,
real) ; 本地温度_开尔文是本地流 (“local_temperature_kelvin” , 实数)

[0141]

temperature_kelvin is stream (“temperature_kelvin” , real) ; 温度_开尔文是流 (“temperature_kelvin” , 实数)

vapour_pressure is stream (“vapour_pressure” , real) ; 蒸汽压力是流 (“vapour_pressure” , 实数)

local_pressure_differential= (a-b) , 从 inlet_pressure 中选择 a , 并从 outlet_pressure 中选择 b;

pressure_differential = p , 从 local_pressure_differential 中选择 p;

cavitation_alert=1 , 如果 local_pressure_differential> 25 , 否则为 0;

local_temperature_kelvin = ((temp + 459.67) × 5/9) , 从 temperature 中选择 temp;

temperature_kelvin = t , 从 local_temperature_kelvin 中选择 t;

vapour_pressure= (20.386- (5132.0/tk)) , 从 local_temperature_kelvin 中选择 tk;

[0142] 在上面的示例中,使用三个物理传感器数据流来创建一组虚拟传感器,其稍后用于检测泵气蚀场景。虚拟传感器既可以是本地流,也可以作为第一类数据流发布在数据总线上。

[0143] 图7示出了用于从输入创建虚拟传感器的传感器表达式语言引擎707。传感器表达式语言引擎从物理传感器或其他虚拟传感器获取输入。输入的一些示例包括进气压力711、出气压力714、温度717、和流量720。可以使用任何数量的输入或输入的组合作为虚拟传感器的输入。基于输入,传感器表达式语言引擎可以生成具有输出的虚拟传感器,例如压力差731、温度734(可以以开尔文为单位)和蒸汽压力737。可以有任意数目的虚拟传感器和输出。如上所述,输出可以是关于虚拟传感器的输入的数学函数。

[0144] 尽管图7示出了表示虚拟传感器的框(例如,731、734和737)。虚拟传感器可以具有多个输出。例如,虚拟传感器731和734可以组合成具有两个输出的单个虚拟传感器。虚拟传感器731、734和737可以组合成具有三个输出的单个虚拟传感器。

[0145] 边缘和云之间的应用程序移动性

[0146] 传统的企业软件应用托管依赖于数据中心或“云”基础设施来开拓规模经济和系

统效率。但是,这些数据中心可能与企业进行大部分作业操作的物理操作点(例如,工厂、仓库、零售店、以及其他设施)相隔任意距离。物联网(IoT)指用例的集合,其依赖于利用以非常高的频率跟踪事件的传感器的物理操作仪器。

[0147] 该系统包括一种用于软件应用在后端数据中心或云与前线边缘基础设施层之间的无缝互操作性和移动性的方法和装置:

[0148] 1.该装置是一种软件平台,其可以以不同形式跨越云、中等规模计算服务器和小型化计算服务器部署,它们具有可由应用程序用于在进行物理作业时完成数据管理、数据分析、以及管理任务的同一组应用程序接口(API)。

[0149] 1.1.数据管理包括从边缘层处的多个网络接口摄取数据流、数据丰富化、存储、抵御硬件和软件故障的韧性以及数据的一致性。

[0150] 1.2.数据分析包括复杂事件处理(CEP)引擎、用于分析或流处理或用于分析和处理的表达式语言、汇集、规则和在边缘处选择机器学习工作流程。

[0151] 1.3.管理功能包括在边缘层处的资源调配、配置和设置以及应用程序生命周期任务管理。

[0152] 2.该系统还包括用于通过互联网在云和边缘基础设施层之间迁移这些应用程序而又能同时保持所管理的应用程序和数据的状态的一致性的方法。

[0153] 3.该方法利用应用容器技术来打包跨越不同平台的一致迁移或执行所需的软件库。

[0154] 4.该方法还可以具有推荐者系统,该系统考虑资源可用性、应用服务质量(QoS)需求和优先级来调度这种在云和基础设施的边缘层之间的迁移。

[0155] 5.该方法还允许应用程序进行合作,以便同时在基础设施的云层和边缘层上执行,并利用该装置(例如,边缘平台)来管理应用程序间的通信。

[0156] 在边缘开发的应用程序可以在云中运行,反之亦然。对于机器学习应用也同样如此。

[0157] 远程管理的工作流程

[0158] 作为边缘计算软件的一部分的部分上述系统可以通过由几个微服务支持的管理控制台来远程管理。可以通过此远程管理控制台配置、部署、管理和监控多个不同的边缘安装。

[0159] 一种用于实现边缘处智能的方法。其特征包括:由在网关设备或嵌入式系统上托管的软件层中的传感器数据触发。软件层被连接到局域网。服务、应用程序和数据处理引擎的存储库被使得能够由软件层访问。通过由软件层提供的表达式语言将传感器数据与对特定状况的发生的语义描述进行匹配。通过持续执行表达式来自动发现模式事件。智能地跨越网关设备和网络上的由软件层管理的嵌入式系统对服务和应用程序进行编排以链接应用程序和分析表达式。基于资源可用性来优化应用程序和分析的布局。监视软件层的健康状况。将原始传感器数据或表达式的结果存储在本地时间序列数据库或云存储中。服务和组件可以被容器化以确保在任何网关环境中的平稳运行。

[0160] 在实施方式中,一种用于服务调用的方法,其中,服务调用由在网关设备上托管并被连接到广域网的软件装置中的传感器数据触发,其中,服务、应用程序和数据处理引擎的存储库被使得能够由该装置访问;将所述传感器数据与可由该装置提供的数据的语义描述

进行匹配;发现针对匹配的数据语义类型的模式所设计的所有应用程序;智能地跨越网关设备和分布在由软件装置管理的广域网上的服务器对服务进行编排以链接所有匹配的应用程序和分析;基于网关设备处的资源可用性来优化应用程序和分析的布局。

[0161] 1.技术领域:本发明涉及物联网领域,特别是用于从传感器导出实时数据驱动的决策的分布式软件。

[0162] 2.讨论:由于传感器和网络(统称为物联网)的成本的降低,传感器数据集呈爆炸式地增长。数据的抽象概括了跨越各个垂直行业的物联网,包括制造业、能源及公用事业、城市基础设施、医疗保健、零售业、农业与资源、家庭自动化、以及消费者可穿戴设备。但是,软件开发、测试、部署、和管理的成本的急剧增加继而增加了出于在流程效率、业务盈利能力、和收益货币化方面创造价值的目的而摄取、汇合、管理、和处理这些传感器数据的经济障碍。物联网软件的这些经济障碍可被分为如下四类:

[0163] 1.异质性:物理世界中的数据源本质上是异质的,并且异质数据源的堆积会增加软件开发和测试的成本负担。

[0164] 2.最后一英里缺口:数据中心和物理传感器之间存在用于可靠地部署具有高质量服务的软件应用程序的缺口。

[0165] 3.安全性:传感器是用更少定义的数据管控和安全策略刻画新的信息路径的新数据来源。

[0166] 4.存储仓(silo):跨越应用程序存储仓进行的重复数据采集、处理、和管理会使硬件和开发资源的使用率低。

[0167] 该系统提供了一种被设计用于解决这些靠近传感器最后一英里处的难题的方法和软件装置。本发明装置中的智能可以发现传感器数据的语义;定制对服务和分析的编排,并链接软件应用程序的部署。

[0168] 本发明的一些实施例包括一种用于使软件应用程序能够访问来自遍布广阔区域的各种传感器的数据并且基于数据和分析提供实时响应和消息的方法和装置。该装置被设计为部署在靠近传感器和由企业或云服务供应商管理的数据中心中的服务器基础设施的嵌入式网关设备上。应用程序开发者向该装置表达他们的软件所使用的数据语义,并将他们的应用程序存储库暴露给该装置。网关设备可以通过其硬件系统中的网络接口访问物理传感器。

[0169] 所要求保护的方法解决了物联网软件的异质性、最后一英里缺口、安全性、和数据存储仓难题,如下所示:

[0170] 1.1.传感器数据通过网关设备中的网络接口到达软件装置。软件装置检查指示所支持的已知数据协议的模式的报头。

[0171] 1.2.一旦发现所支持的协议的报头,该装置便在编排的第一阶段加载相关的协议署理服务。

[0172] 2.软件装置通过协议署理发现数据流中的数据语义的维度,与应用程序开发者所表达的语义中的模式进行匹配,并且创建所匹配的元素的评价列表。所有匹配扫描都是通过令牌交换协议进行保护的,以便确保开发人员能够访问正在匹配的数据。该装置可以加载具有最高匹配分值的数据语义,或者通过人机界面推荐同一数据语义以进行人工确认。服务编排添加了开发者数据语义。

[0173] 3. 软件装置随后将来自不同物理源的传感器数据混合以匹配应用程序的数据语义定义。

[0174] 4. 应用程序的分析服务依赖性由传感器附近的网关设备上的软件装置以及同一装置管理下的数据中心中的其他服务器进行分析和配设。

[0175] 5. 具有适当语义的数据流通过数据处理流水线被软件装置输送以根据应用需要来转换数据。

[0176] 6. 该装置还管理通过安全链路在广域网上进行的数据传输,以实现在地理上分散的分析处理流水线之间的一致性。

[0177] 7. 应用程序可以通过在软件装置手册中向开发人员公开的编程API (具有基于令牌交换的安全性) 安全地访问经转换的数据。根据服务质量需求和该实例处的资源可用性,应用程序被链接到适当场所(网关设备、数据中心服务器)的服务组合。

[0178] 上面的阶段1-阶段7描述了一种由软件装置通过利用来自传感器数据的动态环境和存储库中的应用程序来智能地编排服务链的方法。因此,该方法和软件装置解决了上述物联网在软件工程方面的难题。

[0179] 异质传感器不可知式软件和数据管理

[0180] 物联网 (IoT) 通过配备高灵敏度传感器的物理作业设施为业务运营带来价值,其中,以高频率跟踪操作中的事件并通过软件和服务将传感器数据转化为可操作的分析剖析。但是,日常作业的物理设施(例如,工厂、仓库、零售店和其他设施)在传感器硬件、网络拓扑结构、数据协议和数据格式方面异质性明显。这种异质性会造成软件开发、测试、和部署方面高昂的费用开销;代码模块需要针对硬件联网数据协议数据格式的每种组合进行再开发。该系统提供了一种用于在这种异质设置环境中管理软件和数据的方法和软件装置。

[0181] 软件装置可选地被设计为托管在包括诸如搭配在传感器网络中的中型服务器(例如,双核处理器和4千兆字节的存储器)至小型化服务器(例如,具有少于1千兆字节的存储器的单核处理器核心)的计算基础设施上。下面描述的方法通过传感器和在软件装置上托管的应用程序之间的数字接口层来抽象化传感器网络中的异质性:

[0182] 1. 署理(例如,图5中的署理520):当传感器被连接时,连接会话发生,随后相应的客户端将发布并订阅主题名称。这是出于读取大量数据的目的而被优化的多协议署理。署理已经建立了安全机制来保护对安全证书的访问。署理具有包含特定用户的登录名和密码的配置文件。署理还建立了有关访问主题的策略。该系统是一种足以将这些特征应用于诸如MQTT、CoAP、AMQP、或DNP3之类的内置和种子协议署理以及自定义协议的通用系统。协议署理的插件架构使针对自定义协议的署理的开发非常容易。署理本身具有高性能,并使用零拷贝缓冲来实现快速处理。署理层是一种可避免任何单点故障的轻量级高可用性的实现方式。它具有基于动态计算领域特定语言(DSL)的有向无环图(DAG),从而使署理层具有灵活性和直观性。消息可能不会按时间戳顺序到达,因此很难知道何时关闭用于计算的时间窗口。署理可以通过跟踪应用程序时钟的低位电平(low watermark)来实现克服该问题的模型。

[0183] 2. 流处理图:系统使用有向无环图(DAG)(其为“流处理图”)来描述与流相关联的处理单元之间的生产者-消费者关系。图中存在源节点、宿节点和处理节点,其中,有向边表示各节点之间的信息流。源节点对应于输入数据流的源。这些节点只有流出的边,并且它们

彼此之间没有任何边。宿节点对应于最终经处理的信息的接收者。这些节点只有流向它们的边,并且它们彼此之间没有任何边。处理节点代表处理单元。处理单元可能需要同时来自多个数据流的输入并产生一个或多个有价值的输出流。这样的图可以按照全部有向边指向下的方式来绘制。该系统被设计为信息来自于顶部并通过中间的处理单元最终导向底部的输出流。

[0184] 3. 队列系统:如利特尔法则所定义的,不同层级之间的延迟和吞吐量之间的关系是非常复杂的。我们将每个系统看作一个整体,每个系统包含许多子系统,它们之间会存在复杂的依赖关系。每个子系统通过吞吐量或延迟与系统的其余部分绑定。在设计系统时,将这些交互纳入考虑范围并创建不会阻碍更大系统的系统是非常有用的。如果传感器以高于接收器(其中一者)可以处理消息的速度发送消息,则异步通信中会出现关于流控制的问题。

[0185] 所开发的系统已经完成了许多处理在高容量异步系统中的背压问题的实验,并随之创建了一种强大的反应式流处理和队列机制。启动这些自主发起的反应式流背后的主要驱动力是在无需无限制缓冲区的情况下跨越内存中的异步边界无损地传输数据。所使用的是延续性传递风格(例如,参与者模式)。当下游对流的需求为零时,参与者不会从它们自己的上游获取更多的流,而且,由于参与者是由消息驱动的,因此它们只有在下游对流有更多需求时才会被调度。为此,没有线程被阻塞,也没有调用堆栈被保存,但是参与者会记住它在流中的位置。这种设计促使实现一种考虑消息处理中的非确定性的新方式。

[0186] 4. 边缘混合器:每个传感器发布自己的主题。测量结果进行布告的时间间隔与传感器有关。最后,需要标准化以混合传感器。传感器测量结果被发布到基于时间的信道。这些测量结果以循环的方式被发布到这些队列中。部分或全部传感器测量结果可以在时间维度上缝合。系统的传感器混合器执行以下传感器时间序列标准化中的一者或多者:

[0187] 4.1. 标准化到区间内。

[0188] 4.2. 标准化为总和1。

[0189] 4.3. 标准化为欧几里德范数1。

[0190] 4.4. 标准化为零均值。

[0191] 4.5. 标准化为零均值和单位标准差。

[0192] 5. 传感器表达式语言:应用程序通过被称为传感器表达式语言的语言来定义边缘混合器中的流水线工作流程。该语言对数据管理和分析的应用程序端的抽象进行了定义,从而控制了用于抽象传感器异质性的端到端方法。传感器表达式语言的具体实施方式被称为Vel(R),并且在2016年3月23日递交的美国专利申请62/312,106、62/312,187、62/312,223、和62,312,255中进行了描述,其内容通过引用被结合于此。Vel是FogHorn Systems, Inc.的商标。

[0193] 基于边缘和设备类型的智能容器创建和管理

[0194] 容器技术无需为每个租户复制整个OS内核(并且因此与超级管理技术不同)便可以忽略的开销将作为被管理的操作系统(OS)的计算机服务器资源(例如,存储器、CPU、和存储装置)虚拟化。容器是作为现行Linux开源操作系统的一部分而开发的,并且随着诸如Docker和CoreOS之类的高级管理框架的普及在涉及软件开发和数据中心操作(“DevOps”)方面有着巨大的吸引力。对用于在非常接近物联网(IoT)使用案例(包括工厂、仓库、零售

店、和其他设施等物理位置)中的物理传感器网络的近线计算基础设施上处理传感器数据而执行的软件和分析的需求日益增加。这些计算节点包括例如从中型服务器(例如,双核处理器和4千兆字节的存储器)至小型化服务器(例如,具有少于1千兆字节的存储器的单核处理器核心)的服务器,这些服务器被连接到互联网并可在作业时访问部署的各种异质传感器设备和控制系统。该系统提供了一种在这些边缘计算基础设施设置中智能地部署和管理容器技术的方法。

[0195] 以下组成元素描述了一种用于在边缘基础设施环境中管理容器的方法并以软件装置的形式来实现:

[0196] 1.集中管制:系统在其针对IoT的工业AppStore(AppStore for IoT(TM))中管理软件应用程序,并且所有部署决策在该集中式设置中进行控制和管理。AppStore for IoT是FogHorn Systems,Inc.的商标。

[0197] 2.部署拓扑模板:系统具有用于每个软件应用程序模块的模板,该模板自包含有在边缘基础设施上进行部署所需的所有拓扑细节和运行库。部署包括将所需的网络协议、消息协议和数据格式解析器库编排为服务链,应用程序可以使用该服务链来访问来自传感器网络中的系统的数据。使用目标边缘基础设施中的系统配置信息来以智能且自动的方式执行服务编排。

[0198] 3.应用程序容器的移动性:系统对包含所配置的应用程序的部署模板对象进行序列化,并通过互联网将它们流送至边缘基础设施处的软件装置中的代理。

[0199] 4.边缘处的零接触部署:边缘处的系统软件装置能够解析从工业AppStore for IoT接收的所配置的容器对象,并通过在平台中实现的所有分析和数据管理服务对其进行部署。虽然在配置中可能需要一些手动操作,但部署过程是完全自动化的,无需人工干预。

[0200] 5.应用程序容器的监视:系统在实现每个容器中的遥测代理的边缘处的软件平台(装置)中部署基于微服务的架构。这种代理会测量并报告对容器中应用程序的性能和可用性的详细度量。如果这些度量中的任一者表现异常,该平台还能够发出警报。

[0201] 6.容器的响应迁移:在遥测代理观察到资源争用和服务降级的情况下,系统提供触发容器从边缘平台迁移回云的方法。这种响应式迁移将受限边缘基础设施中的资源释放,并有望提高所托管的应用程序的整体服务质量。应用程序的优先级也作为调度此类迁移的指导标准。

[0202] 用于针对超大型传感器网络的自动模型-并行机器学习的方法以及用于在传感器网络与数据中心之间进行数据并行传输学习的装置

[0203] 机器学习已经发展成为一种关键的计算构造,用于自动发现数据中的模式,并使用所构建的模型在各种工业领域进行智能预测。机器学习在传感器数据上的应用与整个物联网(IoT)分块市场中的数据驱动业务和运营规划用例相关。但是,传感器数据对机器学习工作流程的可扩展性和效率提出了如下几个难题:

[0204] 1.流数据的量:在作业中部署的控制系统和机器产生极大量的连续流式数据,这对用于机器学习分析的汇集造成成本和逻辑上的限制。

[0205] 2.数据质量和复杂性:来自工业系统的数据具有非常少量的与泛滥的数据损坏和缺失值相关联的结构和元数据。构成所生成的数据的这些特征在类似操作系统的不同实例之间表现出的差异很大。这严重降低了通过传统机器学习分析开发的模型的保真度。

[0206] 3.数据预处理方面的缺口:缺少计算基础设施来以智能且有效的方式预处理离源头更近的数据。

[0207] 4.地理分布:IoT场景中的工业作业涉及在大片地理区域中部署的系统,这对任何计算和数据的分发施加了成本、带宽和延迟限制。

[0208] 系统提供了一种平台的形式的软件装置,该平台被设计为跨越数据中心或“云”执行机器学习工作流程,并计算传感器网络或“边缘”附近可用的资源,例如来自中型服务器(例如,双核处理器和4千兆字节的存储器)至小型化服务器(例如,具有少于1千兆字节的存储器的单核处理器核心)的资源。系统的方法可以包括以下步骤:

[0209] 1.在边缘处进行流数据处理和网内汇集以预处理可馈送至机器学习分析的连续的原始数据流—处理方法可由构建机器学习分析的开发者通过API进行编程。

[0210] 2.针对由机器学习工作流程构建的模型的传感器数据查询处理,以对边缘处的经预处理的数据执行并将执行结果发送到云用以进行模型评估。

[0211] 3.在边缘处进行传感器数据分割,用于基于它们的组成特征集对原始数据进行分区,并与机器学习工作流程所建立的模型进行智能匹配。

[0212] 4.基于上述预处理、基于查询的模型执行和基于特征的数据分割的工作流程,在边缘处的软件平台中进行实时模型训练和选择。

[0213] 包括这三个步骤的方法使机器学习的系统构造作为一种服务。来自FogHorn Systems的具体实现被称为FogHorn Cortex。

[0214] 1.系统的软件装置跨越边缘基础设施的实例(跨越广阔的地理区域)协调模型的执行。这些模型在跨越不同特征空间段的数据上执行。该装置通过使用云作为转移学习的平台来跨越特征空间传输所学习的知识。这通过避免数据标签和利用带有异质标签的数据为机器学习过程带来了更高的效率。

[0215] 边缘计算平台系统的一个用例是工厂产量优化。例如,一家现代化的制造工厂希望通过尽早发现可能的缺陷以便在装配线中不可逆阶段之前对其进行修正,来使产量最大化。使用边缘计算平台系统的一些好处包括:应用FogHorn的实时数据处理、VEL表达式引擎和与Apache Hadoop的集成以用于大数据分析的组合,制造商可期望废品率降低至少50%。

[0216] 一些解决方案包括:

[0217] 1.在边缘处实时收集和转换来自设备控制器的数百个MQTT和OPC-UA数据值以及各个部件的RFID数据。

[0218] 2.使用VEL表达式进行测量结果的标准化和汇集并将其与规格说明书进行比较。

[0219] 3.将原始数据和经强化的数据传输到Hadoop,在此处数据学家可以建立表征测量结果序列的模型,这种模型可进行缺陷检测。

[0220] 4.使用VEL表达式来向技术人员发送警报,以便在不可逆转的步骤(例如,加热、干燥和喷漆)之前舍弃产品或将产品返工。

[0221] 实时分析和智能包括:收集和转换测量结果和标识符或ID。用VEL计算和汇集测量结果。历史分析包括:识别预测特征。构建和测试针对废品的序列挖掘模型。

[0222] 边缘计算平台系统的一个用例是机车燃料效率。例如,运输公司需要一种可定制的解决方案来主动监控和优化列车和机组性能以降低油耗。使用边缘计算平台系统的一些好处包括:能够通过基于实时引擎和地理位置数据检测过度闲置和低效的节气门位置来为

每台发动机节省数千加仑的燃油。只有在异常情况下才会触发视频上传,从而降低了移动网络成本。

[0223] 一些解决方案包括:

[0224] 1.从机载硬化数据收集系统摄取传感器和GPS数据。

[0225] 2.使用VEL表达式来基于位置、速度、时间、及其他参数来确定怠速和节气门调节中的状况。

[0226] 3.向指挥中心发送警报,在指挥中心处操作员可以采取主动行动来减少燃油消耗,同时维持最佳列车操作。

[0227] 4.使用实时应用程序来检测不安全状况并触发用于进行分析的到数据中心的视频流上传。

[0228] 实时分析和智能包括:数据汇集。规则和警报。触发数据传输。历史分析包括:针对列车操作、安全性和燃料消耗的规则和预测模型。

[0229] 边缘计算平台系统的一个用例是泵气蚀警报。例如,设备制造商需要一种解决方案来检测气蚀事件,这些气蚀事件可能会导致在工厂和发电厂中安装的泵和控制阀产品的损坏。使用边缘计算平台系统的一些好处包括:在气蚀发生时采取纠正措施来提高设备使用寿命。通过预测性维护避免意外停机和灾难性故障。

[0230] 一些解决方案包括:

[0231] 1.从附接到设备的多个传感器收集诸如进气和出气压力、流量和温度之类的测量结果。

[0232] 2.使用VEL表达式来实时计算蒸汽差以检测气蚀形成。

[0233] 3.向技术人员发送警报,以便增加流体压力或流量,或者降低温度以避免造成密封件、轴承和叶轮的损坏。

[0234] 4.分析历史信息以识别模式,以使技术人员能够在故障发生之前检测到气蚀或安排维护。

[0235] 实时分析和智能包括:实时收集和转换测量结果。用VEL计算导致气蚀的蒸汽差。历史分析包括:分析时间序列测量结果以识别气蚀事件的模式。

[0236] 边缘计算平台系统的一个用例是风能预测。例如,按照政府的要求,可再生能源公司需要在未来24小时内准确预测、报告并满足发电需求。使用边缘计算平台系统的一些好处包括:在涡轮机无法满足24小时能源生成预测之前,能够提前至少90分钟生成警报,并将其应用于优化涡轮机设置或修改预测。

[0237] 一些解决方案包括:

[0238] 1.从每台涡轮机的SCADA控制器系统收集六个月或更长时间段内的历史和实时数据,并增加天气、大气和地形数据。

[0239] 2.基于20个或更多属性训练模型来预测15分钟时间段内的发电量。

[0240] 3.在边缘处应用模型以生成发电量的实时评分。使用VEL表达式来将性能与预测进行比较以生成警报。

[0241] 4.使技术人员能够微调涡轮机设置或修改预测。

[0242] 实时分析和智能包括:收集和转换数据。评分和更新模型。用VEL识别状况。历史分析包括:统计分析。提取特征。构建和测试模型。

[0243] 已出于说明和描述的目的呈现了对本发明的描述。其并非意在穷举或将本发明限制于所描述的确切形式,并且鉴于上述教导,许多修改和变形是可能的。本说明书选取并具体描述这些实施例,是为了更好地解释本发明的原理和实际应用。本说明书使所属技术领域的其他技术人员能够通过具有适合于所考虑的特定用途的各种修改的各种实施例最好地利用和实践本发明。本发明的范围由所附权利要求限定。

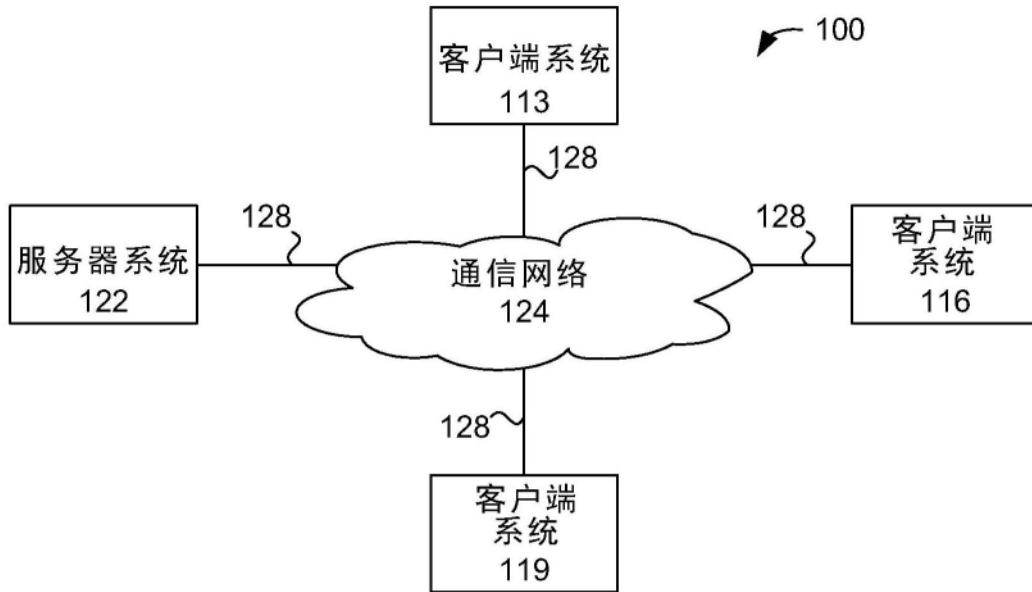


图1

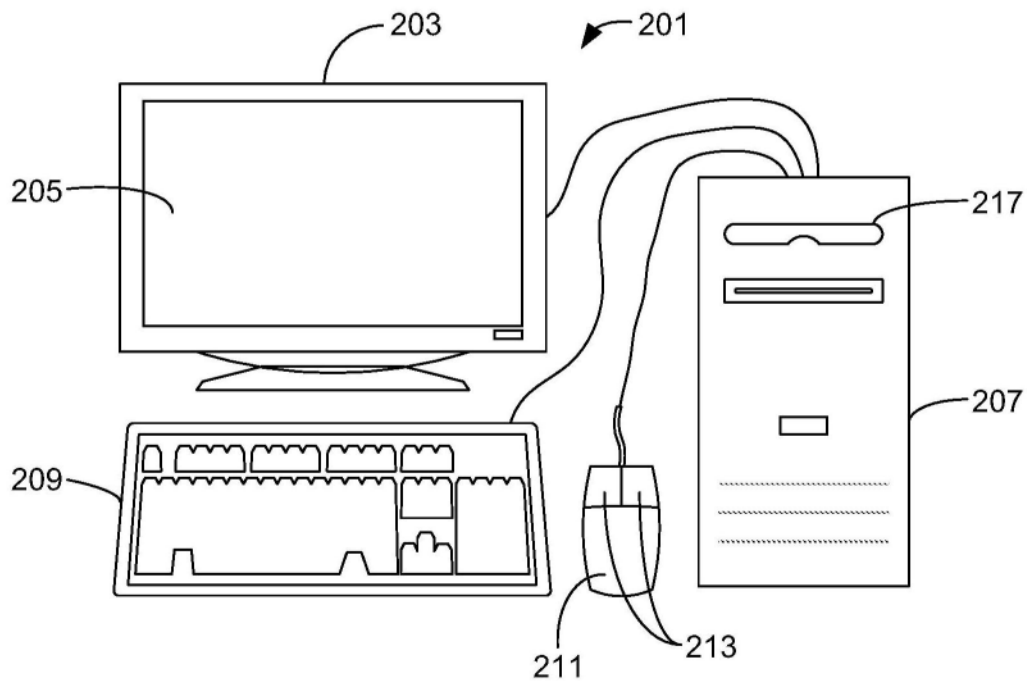


图2

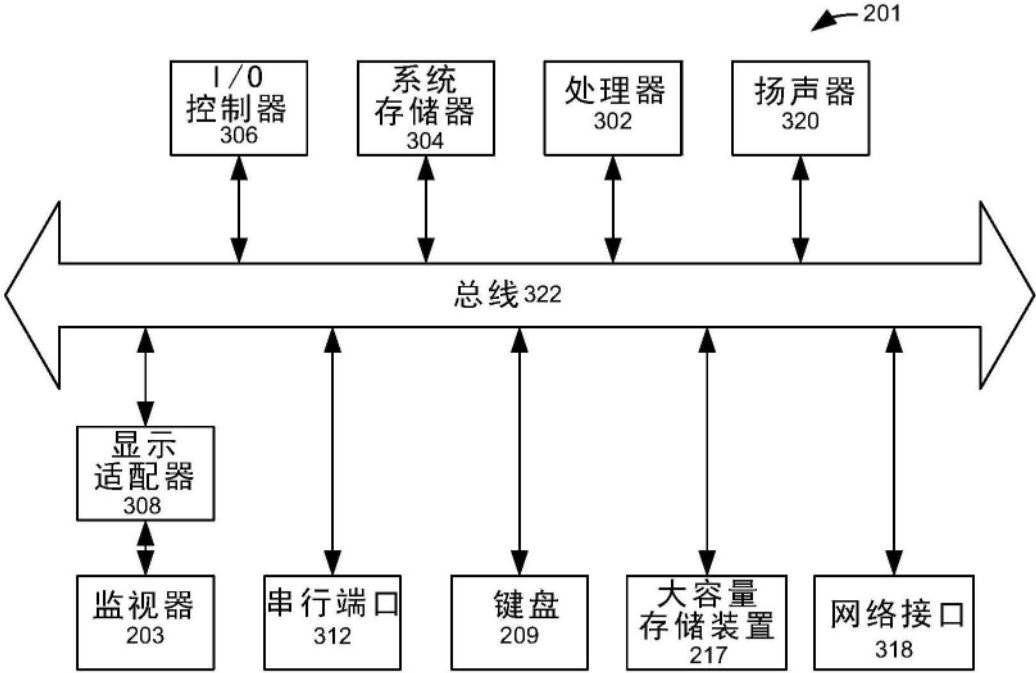


图3

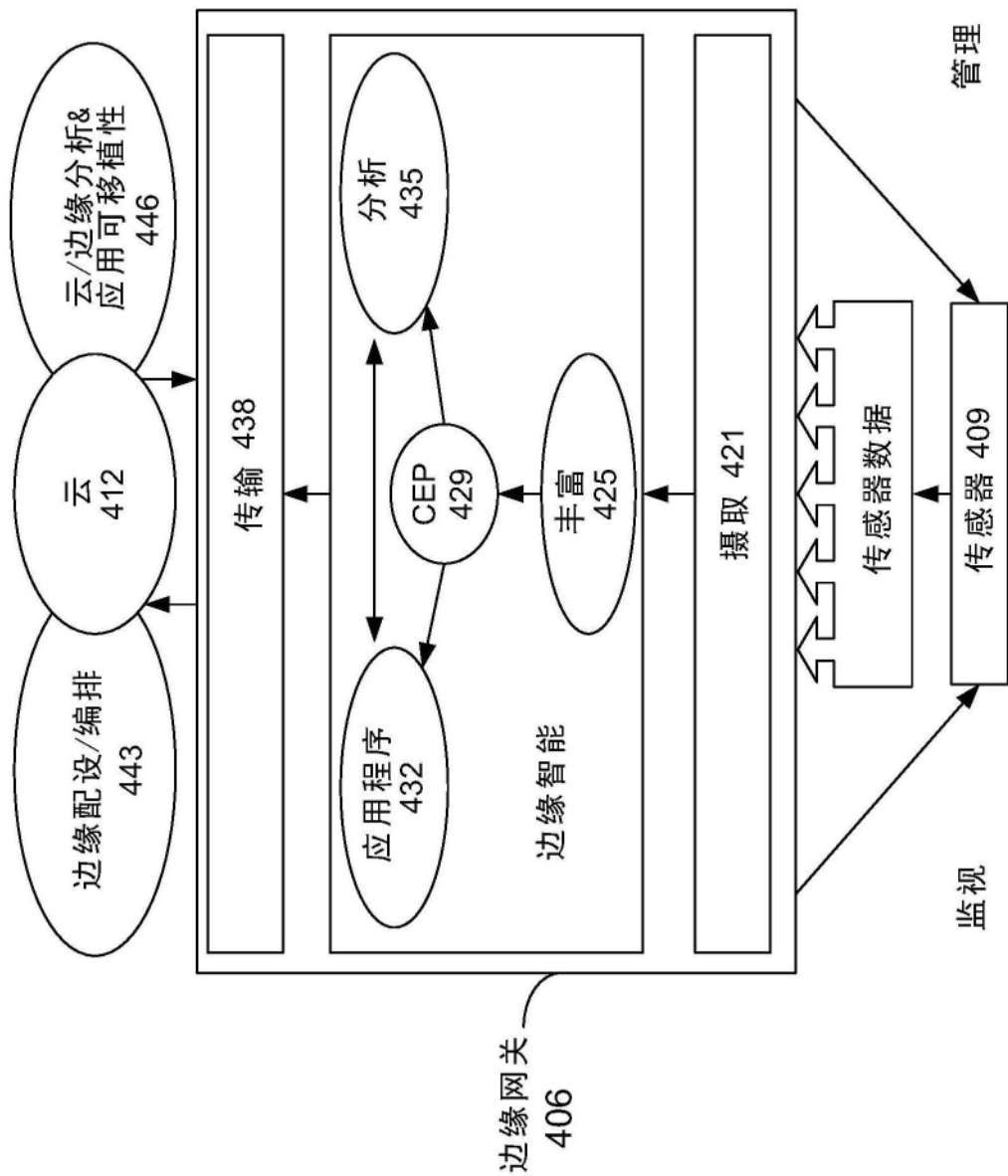


图4

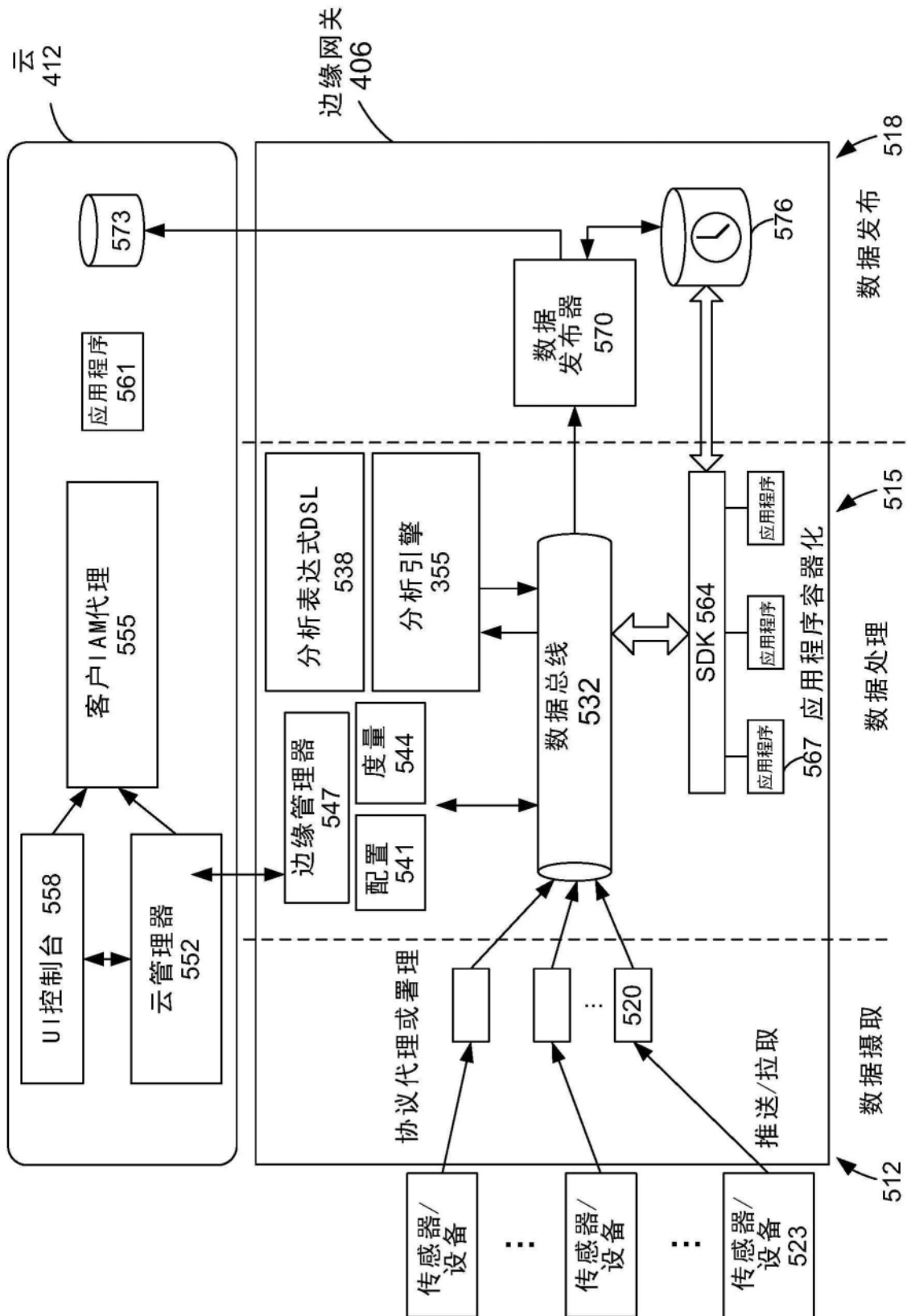


图5

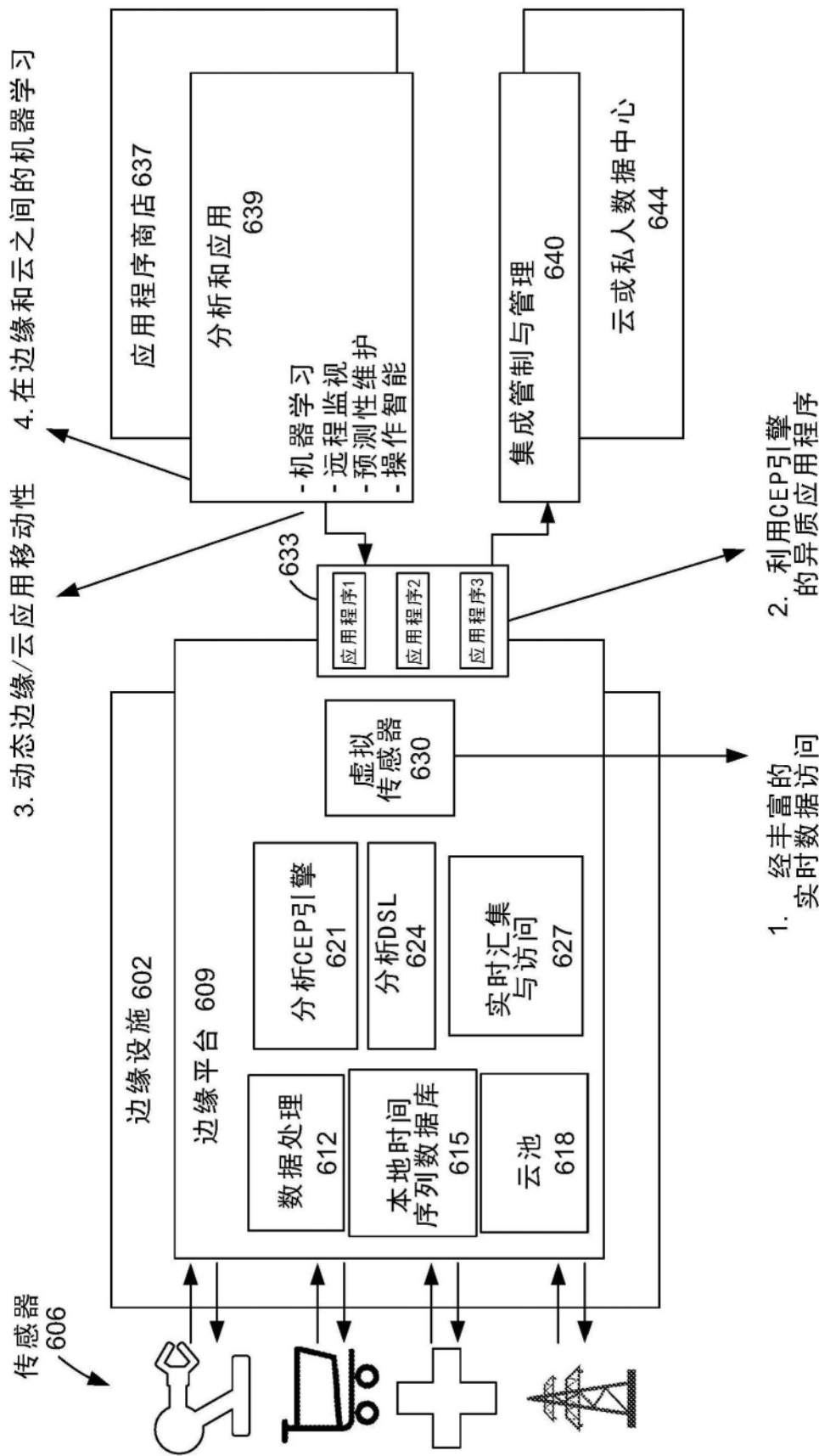


图6

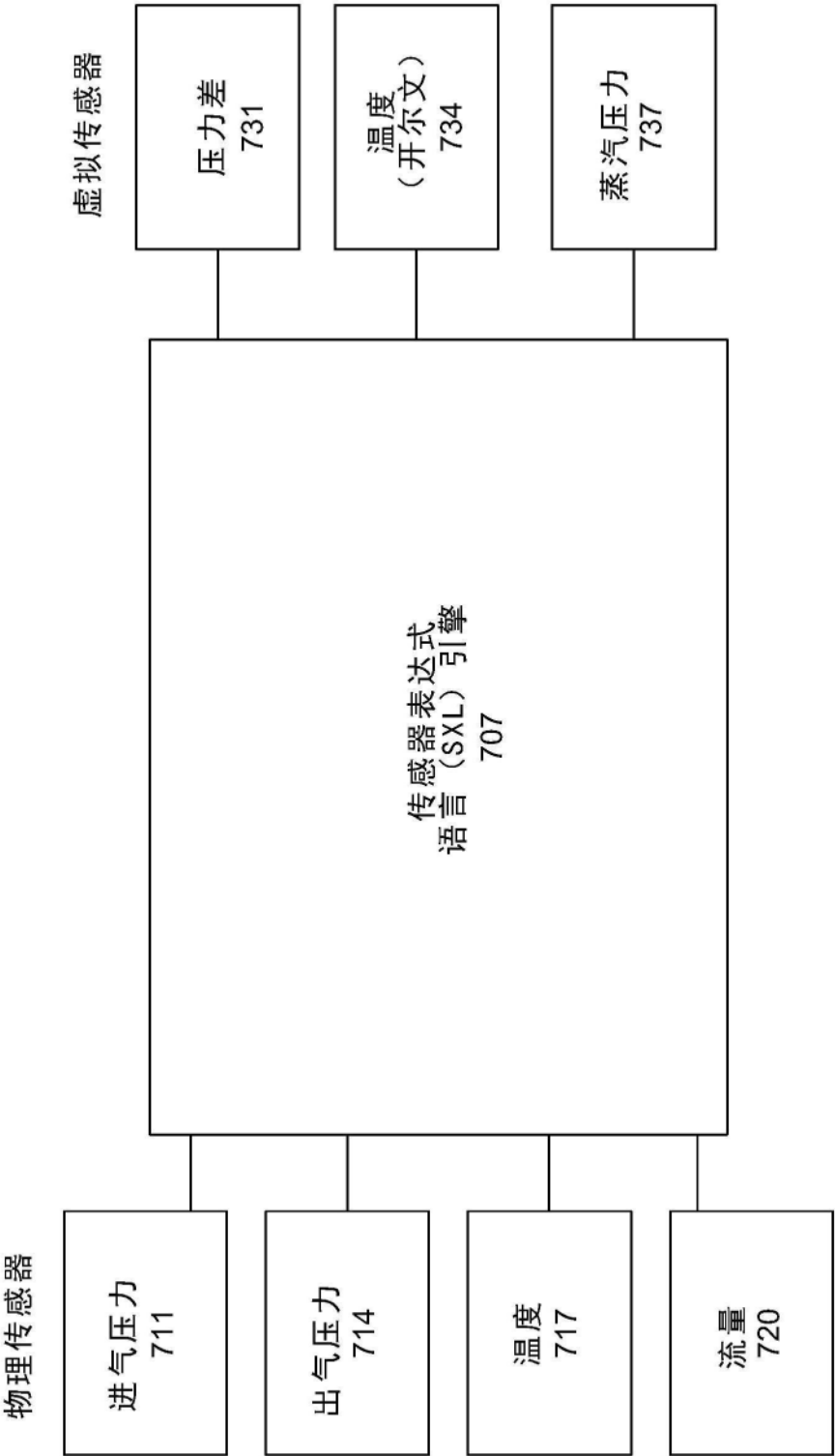


图7