(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 110622128 A
(43)申请公布日 2019.12.27

(21)申请号 201880031856.1

(22)申请日 2018.04.27

(30)优先权数据
15/596,306 2017.05.16 US

(85)PCT国际申请进入国家阶段日
2019.11.14

(86)PCT国际申请的申请数据
PCT/US2018/029769 2018.04.27

(87)PCT国际申请的公布数据
WO2018/212958 EN 2018.11.22

(71)申请人 甲骨文国际公司
地址 美国加利福尼亚

(72)发明人 S·科拉查拉 许健午 T·宏 L·罗登贝瑞 黄登胜 M·科斯拉维 P·霍兰德 B·帕特尔 A·莫汉 K·谢提 郎卫刚 E·葆莱米克 黄志斌 S·库尔特

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 冯薇

(51)Int.Cl.
G06F 8/00(2006.01)
G06Q 10/10(2006.01)

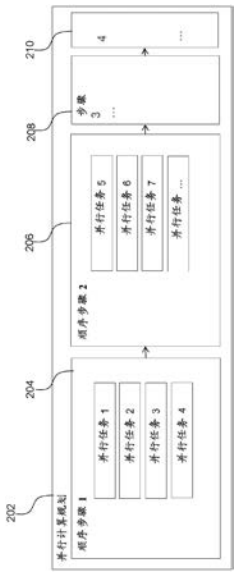
权利要求书2页 说明书17页 附图6页

(54)发明名称

计算处理的动态并行化

(57)摘要

一种系统，该系统用于通过将基于规则的表达式列表划分到多个任务单元中并将处于相同计算级别的所有独立任务单元重新配置到若干并行化任务组中，生成评估规则集合或基于规则的表达式列表的并行计算规划，使得每个任务组内的任务单元可以被调度用于跨处理节点集群的并行执行。并行化可以基于所生成的任务被动态地确定，但是可以基于将每个任务基于范围地划分到多个并行可执行子任务中而进一步进行附加层的并行化。最终的并行化计算规划可以包括基于利用关于每个任务组的并行化执行的信息的问题分区和逻辑依赖性的顺序排序的任务组集合。



CN 110622128 A

1.一种具有存储在其上的指令的非暂态计算机可读介质,所述指令在由处理器执行时使所述处理器生成评估基于规则的表达式集合的并行计算规划,所述生成包括:

将所述基于规则的表达式集合划分为多个任务单元;以及

将所述多个任务单元布置到可并行化任务组的顺序集合中,其中顺序排序通过与所述基于规则的表达式集合相关联的逻辑依赖性问题分区来确定;以及

生成所述并行计算规划,所述并行计算规划包括可并行化任务组的顺序执行排序和每个可并行化任务组内的任务单元的并行执行排序。

2.如权利要求1所述的非暂态计算机可读介质,其中所述多个任务单元包括评估所述基于规则的表达式集合中的每个基于规则的表达式所需的一个或多个初始化、聚合或计算操作。

3.如权利要求1所述的非暂态计算机可读介质,还包括将来自所述可并行化任务组的顺序集合中的一个或多个可并行化任务组划分为可并行化子任务组的并行集合。

4.如权利要求3所述的非暂态计算机可读介质,其中所述可并行化子任务组的并行集合是通过来自所述可并行化任务组的顺序集合中的一个或多个可并行化任务组的基于范围的并行化生成的。

5.如权利要求1所述的非暂态计算机可读介质,其中所述并行计算规划是由计算引擎生成的,并且被分派到中间层用于调度和执行。

6.如权利要求1所述的非暂态计算机可读介质,其中所述并行计算规划被调度用于跨计算机集群的并行执行。

7.如权利要求1所述的非暂态计算机可读介质,其中所述并行计算规划是由零售预测应用服务器(RPAS)的计算引擎生成的。

8.一种用于生成评估基于规则的表达式集合的并行计算规划的计算机实现的方法,所述方法包括:

将所述基于规则的表达式集合划分为多个任务单元;

将所述多个任务单元布置到可并行化任务组的顺序集合中,其中顺序排序通过与所述基于规则的表达式集合相关联的逻辑依赖性和问题分区来确定;以及

生成所述并行计算规划,所述并行计算规划包括可并行化任务组的顺序执行顺序和每个可并行化任务组内的任务单元的并行执行排序。

9.如权利要求8所述的计算机实现的方法,其中所述多个任务单元包括评估所述基于规则的表达式集合中的每个基于规则的表达式所需的一个或多个初始化、聚合或计算操作。

10.如权利要求8所述的计算机实现的方法,还包括将来自所述可并行化任务组的顺序集合中的一个或多个可并行化任务组划分为可并行化子任务组的并行集合。

11.如权利要求10所述的计算机实现的方法,其中所述可并行化子任务组的并行集合是通过来自所述可并行化任务组的顺序集合中的一个或多个可并行化任务组的基于范围的并行化生成的。

12.如权利要求8所述的计算机实现的方法,其中所述并行计算规划是由计算引擎生成的,并且被分派到中间层用于调度和执行。

13.如权利要求12所述的计算机实现的方法,其中所述计算引擎是零售预测应用服务

器(RPAS)的一部分。

14.如权利要求8所述的计算机实现的方法,其中所述并行计算规划被调度用于跨计算机集群的并行执行。

15.一种用于生成评估基于规则的表达式集合的并行计算规划的系统,包括:

第一单元,用于将一个或多个基于规则的表达式集合划分为多个任务单元;

第二单元,用于将所述多个任务单元布置到可并行化任务组的顺序集合中,其中顺序排序通过与所述基于规则的表达式集合相关联的逻辑依赖性和问题分区来确定;以及

第三单元,用于生成并行计算规划,所述并行计算规划包括可并行化任务组的顺序执行排序和每个可并行化任务组内的任务单元的并行执行排序。

16.如权利要求15所述的系统,其中所述多个任务单元包括评估所述基于规则的表达式集合中的每个基于规则的表达式所需的一个或多个初始化、聚合或计算操作。

17.如权利要求15所述的系统,其中所述第二单元还将来自所述可并行化任务组的顺序集合中的一个或多个可并行化任务组划分为可并行化子任务组的并行集合。

18.如权利要求17所述的系统,其中所述可并行化子任务组的并行集合是通过来自所述可并行化任务组的顺序集合中的一个或多个可并行化任务组的基于范围的并行化生成的。

19.如权利要求15所述的系统,其中所述第一单元和所述第二单元构成计算引擎,并且其中由所述计算引擎生成的所述并行计算规划被分派到中间层级别用于调度和执行。

20.如权利要求19所述的系统,其中所述计算引擎是零售预测应用服务器(RPAS)的一部分,并且其中所述并行计算规划被调度用于跨计算机集群的并行执行。

计算处理的动态并行化

技术领域

[0001] 一个实施例一般而言涉及规划系统(planning system),并且特别地,涉及改善规划系统中的计算处理。

背景技术

[0002] 规划系统帮助零售商规划和管理零售业务中的销售额、营业毛利和库存周转率。在这样的系统中,可以使用“规则”来表达业务逻辑。例如,规则:销售额＝价格*单位表示“销售额”作为每单位“价格”和所售出的“单位”的数量的函数,并通过将每单位“价格”乘以所售出的“单位”的数量来评估它。规则中的每项映射到规划数据库中的数据对象。大型规划系统可能涉及数百至数千个这样的规则。这些规则是预先配置的或者是动态生成的。必须频繁重新评估这些规则,有时是在线评估,有时是以批处理的形式评估。由于系统中定义的大量数据和大量规则,计算的性能成为规划系统的总体性能的关键方面。

发明内容

[0003] 一个实施例涉及一种生成并行计算规划以评估基于规则的表达式集合的系统。该系统将基于规则的表达式集合划分为多个任务单元。然后,该系统将多个任务单元重新布置到可并行化任务组的顺序集合(sequential set)中,其中顺序排序由与基于规则的表达式集合相关联的问题分区(partitioning)和逻辑依赖性来确定。

附图说明

[0004] 图1是根据本发明的实施例的计算机服务器/系统的框图,该计算机服务器/系统包括用于优化计算处理的并行计算规划生成模块。

[0005] 图2是根据本发明的实施例的计算并行化规划的框图。

[0006] 图3是根据本发明的实施例的在RPAS平台上的计算并行化规划实现的操作流程图。

[0007] 图4是根据本发明的实施例的并行计算规划中的顺序排序标准的图示。

[0008] 图5是根据本发明的实施例的基于度量(measure)的并行化计算规划的框图。

[0009] 图6是根据本发明的实施例的基于度量和基于范围的并行化计算规划的框图。

具体实施方式

[0010] 用于分析的规划和管理的基于规则的表达式的评估是如由表达式之间的逻辑相互依赖性确定的顺序处理。评估表达式集合通常涉及顺序模式,因为某些表达式的评估可能取决于列表中其它地方出现的表达式的计算结果。因此,与现有计算平台相关联的计算引擎识别现有逻辑依赖性,并相应地对执行规则集合或基于规则的表达式列表的评估所需的计算处理进行顺序化。但是,规则集合的顺序评估未能跟上新技术和高性能硬件体系架构的发展步伐,该高性能硬件体系架构为大数据分析和管理供应了高度可伸缩(scalable)

的计算和处理平台。需要使用创新方法来改善与基于规则的表达式列表或规则集合的评估相关联的速度和可伸缩性，该创新方法使得能够利用更新的高性能计算和处理体系架构。

[0011] 较早期的解决方案往往通过“向上伸缩(Scaling Up)”来解决性能问题(即，利用更昂贵的硬件)。但是，在云方案中，现有的解决方案往往通过“水平伸缩”来解决性能问题(即，添加更多的商用硬件并且并行运行执行)。对于基于规则的计算，水平伸缩方案中的关键问题是与并行执行基于规则的计算相关联的困难。在现有解决方案中，如果可能的话，应当由应用开发人员或设计人员指定如何对计算进行分区和并行执行。

[0012] 本发明的实施例通过将按顺序执行的计算步骤分解成并行化任务组的顺序集合来实现基于规则的表达式的计算的并行化。如本发明的实施例所公开的，基于规则的表达式的计算处理的并行化是新颖的方案，该方案能够充分利用更新的高性能可伸缩计算和处理平台，并且在速度、时间和效率方面产生超过常规方案的显著性能提高。

[0013] 应当注意的是，根据本发明的实施例，并行化或可并行化任务或子任务组是指涉及在相同层次计算级别的逻辑上独立的操作的、可以跨多个处理节点或使用多个处理线程并行执行的任务组或子任务组。任务或任务单元是指可以在执行规划内被独立执行的简单的操作单元。

[0014] 还应注意的是，根据本发明的实施例，并行化或可并行化任务组的顺序集合是指根据顺序执行次序布置的多个并行化任务组，使得每个并行化任务组的并行执行取决于在顺序集合中出现较早的一个或多个并行化任务组的完成。顺序次序可以由规则集合中的表达式之间的逻辑依赖性和/或问题分区(即，相关联的计算的层次级别)决定。

[0015] 实施例在计算处理的顺序方面的执行中引入并行化，以评估基于规则的表达式列表。实施例在顺序步骤的评估中所涉及的任务和操作之间引入并行化，同时维持由基于规则的表达式列表中固有的逻辑依赖性集合所强制执行的顺序次序。因此，按顺序执行的计算步骤集合中涉及的每个计算步骤将在时间和资源利用方面以更高的速度和更高的效率进行评估。与涉及执行多个顺序计算步骤的常规规则集合评估方案相比，本发明的实施例通过例如将顺序计算步骤重构为一个或多个并行化任务组集合来显著改善与规则集合评估处理相关联的时间和资源利用。

[0016] 如上所述，现有解决方案依赖于应用开发人员或分析人员来指定如何直接或间接并行执行计算处理。通常，这通过使用编程应用编程接口(API)或脚本(诸如，“Spark”或“Java流API(Java Stream API)”)或通过将执行流指定为管线(例如，“Apache Pig”或“Crunch”)来实现。这些现有方法中将并行化实施到执行处理中的共同特征是它们必须被预先定义。本发明的实施例描述了动态地将规则集合评估中涉及的执行处理进行并行化的独特方法，该方法极大地改善了性能并减少了操作时间。

[0017] 实施例通过自动分析基于规则的表达式集合并将其变换成可以跨一个或多个计算机集群并行执行的计算任务集合来提高规则集合计算性能。实施例分析基于规则的表达式集合、基于逻辑依赖性和问题分区将规则分解成较小的任务，并且然后基于所生成的任务和问题分区为规则集合自动生成并行执行规划。然后，利用计算机集群并行调度和执行该执行规划。

[0018] 现有技术(诸如可伸缩的处理器体系架构(SPARC))通过编程语言实现并行化。因此，是程序员指定计算处理中的依赖性并确定执行的顺序。作为对照，本发明的实施例公开

了用于计算并行化的高度可配置和动态的方案，其涉及为基于规则的表达式集合生成并行执行规划。因此，根据本发明的实施例，依赖性是基于被评估的基于规则的表达式动态解决的。

[0019] 在本发明的实施例中，与评估规则集合相关联的计算被划分为多个较小的任务并且被布置到多个并行化任务组中。布置到并行化任务组中的任务包括可以并行执行的每个基于规则的表达式的组件。这些组件可以作为一个或多个并行化任务组的一部分从前面拉出并且并行执行，该一个或多个并行化任务组可以进一步被分解成一个或多个并行化子任务组。相对于SPARC的一个区别是所公开的实施例基于表达式以及范围执行并行化。

[0020] 图1是根据本发明的实施例的计算机服务器/系统(即，系统10)的框图。虽然被示出为单个系统，但是系统10的功能可以被实现为分布式系统。此外，本文公开的功能可以在可以通过网络耦合在一起的单独的服务器或设备上实现。此外，可以不包括系统10的一个或多个组件。例如，对于数据库管理系统的功能，系统10可以是通常不需要显示器24或图1中所示的一个或多个其它组件的服务器。

[0021] 系统10包括用于传送信息的总线12或其它通信机制，以及耦合到总线12用于处理信息的处理器22。处理器22还可以包括处理节点的集群，使得每个节点可以被独立地调度以使处理器22能够并行执行多个计算。处理器22可以是任何类型的通用或专用处理器。系统10还包括用于存储信息和要由处理器22执行的指令的存储器14。存储器14还可以包括用于一个或多个规则集合的一个或多个并行化执行表或计算规划，其可以被调度为跨多个处理节点并行执行。存储器14可以包括随机存取存储器(“RAM”)、只读存储器(“ROM”)(诸如磁盘或光盘的静态存储器)、或任何其它类型的计算机可读介质的任何组合。系统10还包括通信设备20，诸如网络接口卡，以提供对网络的访问。因此，用户可以直接或通过网络或任何其它方法远程地与系统10对接。

[0022] 计算机可读介质可以是处理器22可以访问的任何可用介质，并且包括易失性和非易失性介质、可移动和不可移动介质以及通信介质。通信介质可以包括计算机可读指令、数据结构、程序模块或调制数据信号(诸如载波或其它传输机制)中的其它数据，并且包括任何信息传递介质。

[0023] 处理器22还可以经由总线12耦合到显示器24，诸如液晶显示器(“LCD”)。键盘26和光标控制设备28(诸如计算机鼠标)还可以耦合到总线12，以使用户能够根据需要与系统10对接。

[0024] 在一个实施例中，存储器14存储当由处理器22执行时提供功能的软件模块。模块包括操作系统15，该操作系统15为系统10提供操作系统功能。模块还包括用于生成并行化执行方案的并行计算规划生成模块16，以及本文所公开的所有其它功能。系统10可以是较大系统的一部分，诸如向来自Oracle公司的Oracle数据库系统或任何数据库管理系统添加的功能。因此，系统10可以包括一个或多个附加功能模块，诸如附加功能模块18。数据库17耦合到总线12，以向并行计算规划生成模块16和附加功能模块18(即，数据库管理模块)提供集中式存储。在一个实施例中，数据库17是非结构化查询语言(“NoSQL”)数据库，并且并行计算规划生成模块16被实现为Oracle零售预测应用服务器(“RPAS”)的一部分。与本发明的不同实施例相关联的这些示例性特征，即NoSQL数据库和RPAS，在下面进一步详细描述。

[0025] Oracle零售预测应用服务器(通常称为“RPAS”)是用于开发预测和规划应用的可

配置软件平台。RPAS平台提供诸如多维数据库结构批处理和在线处理、可配置的切片和切块(slice-and-dice)用户界面、复杂的可配置计算引擎、用户安全性和实用功能(诸如导入和导出)的能力。

[0026] RPAS是大量应用的基础,这些应用是Oracle零售解决方案覆盖(footprint)的一部分,诸如Oracle零售需求预测、Oracle商品财务规划、Oracle分类规划、Oracle项目规划、Oracle大小配置文件优化、Oracle补货优化和Oracle高级库存规划。RPAS当前提供两个数据持久性选项:基于Oracle Berkeley DB的专有数据存储库或Oracle RDBMS。

[0027] NoSQL数据库提供了用于存储和检索数据的机制,该机制的建模方式不同于关系数据库中使用的表格关系。NoSQL数据库越来越多地用于大数据和实时web应用中。NoSQL系统有时也称为“不仅SQL”,以强调它们可以支持类SQL的查询语言。这种方案的动机包括:设计简单、对机器集群更简单的“水平”伸缩(这是关系数据库的问题)、以及更好地控制可用性。由NoSQL数据库使用的数据结构(例如,键-值、宽列、图形或文档)与关系数据库中默认使用的数据结构不同,从而使NoSQL中的一些操作更快。给定NoSQL数据库的特定适用性取决于它必须解决的问题。有时,由NoSQL数据库使用的数据结构也被视为比关系数据库表“更灵活”。

[0028] 一般而言,本发明的实施例通过将每个步骤划分成多个独立的任务和子任务来在规则集合评估中涉及的每个顺序计算步骤期间进行并行化,该多个独立的任务和子任务可以在可能的情况下并行运行,并且在不可能的情况下依次运行子任务。

[0029] 图2图示了根据本发明的实施例的并行化规划的概述。并行计算规划202包括一个或多个顺序计算步骤,以执行如由在图2中的204、206、208和210指定的基于规则的表达式列表或规则集合的评估。顺序步骤维持顺序关系,使得每个顺序计算步骤需要先完成先前步骤才能执行。但是,每个顺序步骤可以作为并行任务集合执行。任务是可以在执行规划内被独立执行的简单的操作单元。

[0030] 参考图2中的计算规划202,第一顺序步骤204被分解成可以被独立并同时执行的并行任务1-4。作为这种并行执行的结果,第一顺序步骤204将更快地完成,从而导致更早发起第二顺序步骤206。类似地,第二顺序步骤206可以作为如图2所示的可并行化任务集合执行,从而导致更快速地发起第三顺序步骤208。类似地,第四顺序步骤210进行相同操作。以这种方式,计算规划202被分解成可并行化任务组的序列,其维持计算步骤的顺序排序,同时通过并行化执行顺序步骤而供应显著的加速。

[0031] 图3图示了根据实施例的并行化计算处理的操作流程。服务器侧的微服务层302将发起规则集合评估处理。该指令被分派到RPAS层304,在RPAS层304,由计算引擎(CalcEngine)生成并行化的计算规划。CalcEngine在逻辑上确定执行规则集合评估并相应地生成执行规划所需的任务的顺序和并行执行模式。在后端上实现的CalcEngine不具有任务调度功能,因此,一旦规划由CalcEngine生成,它就被传递回中间层微服务层302,以跨处理节点集群实现和分布并行执行。例如,如果由CalcEngine生成的执行规划指定特定顺序步骤以例如包括可以并行运行的500个并行任务,则微服务层302可以将该操作分布在500个节点上,向每个节点指派一个计算任务。如图3所示,计算规划执行由微服务层302指引(direct),微服务层302调度跨RPAS层中的任务执行器308、310和310的第一任务集合的并行执行。在执行规划完成后,通知执行器,使得可以调度与计算的下一个顺序步骤相关联的

后续并行任务集合，并将其指派给任务执行器314、316和318。以这种方式，顺序步骤中包含的任务将跨多个任务执行器同时执行，从而显著加快了对规则集合的评估。

[0032] 如图3所示，微服务层(中间层)302调度任务的并行化执行，同时确保维持计算的顺序次序，因为仅在从运行先前并行化任务的任务执行器接收到完成通知后才调度每个并行任务集合。计算规划被存储在NoSQL数据库层320处的规划/任务表319中。任务执行包括从度量(measure)数据表中读取度量数据，并用任务执行的结果来更新度量数据表。因此，NoSql数据库层320还包括度量数据表320。度量表示容纳特定类型的数据的实体。例如，可以为“销售”创建度量。然后，销售度量将包含所有库存单位(“SKU”)的相关数据(即，销售数据)。

[0033] 由RuleEngine生成的初始基于规则的表达式列表被分解并重新分组为多个顺序排序的计算步骤。计算步骤的顺序次序由计算步骤之间的逻辑依赖性以及要在表达式列表中进行评估的度量的交集(intersection)决定。由此，可以将不依赖于其它表达式的结果的所有表达式(例如，A＝5或B＝今天)汇总到一个顺序步骤中并且并行执行。但是，某些表达式可能取决于其它表达式的评估结果。例如，对表达式“A＝B.top”的评估需要首先完成对“B”的计算。因此，在这种情况下，对“B”的评估必须在对“A”的评估之前，因此，顺序计算次序必须反映这种顺序关系。

[0034] 如上所述，用于确定规则集合的评估所需的顺序排序和顺序计算步骤的数量的一个标准是执行计算的层次或相交级别(intersecting level)。例如，较高级别的度量的计算可能需要聚合与较低级别的该度量相关联的值。计算级别更改可能涉及并行性的级别的更改，并且因此需要新的计算步骤。因此，与表达式的级别间评估相关联的计算在不同的顺序步骤中进行评估，而与同一级别的表达式相关联的计算可以在同一步骤中并行地进行评估。除了如上所述的相关联的计算级别之外，规则集合中的表达式之间的逻辑依赖性也作为刻画与根据本发明的实施例的规则集合的完整评估相关联的多个顺序计算步骤的障碍。

[0035] 一旦基于规则的表达式列表基于相关联的计算之间的现有逻辑依赖性以及相交或级别变化而被分解或重新分组到顺序计算步骤集合中，下一步就是执行与每个步骤相关联的计算。基于规则的表达式可以被表示为方程式或表达式，该方程式或表达式包括表示为一个或多个右手侧(“RHS”)度量(项)的函数的左手侧(“LHS”)度量(项)。因此，执行与顺序计算步骤相关联的计算可能涉及根据RHS度量对LHS度量进行评估。该动作可能首先需要执行聚合子任务，以便量化RHS度量。聚合子任务的执行还可能需要完成初始化子任务，该初始化子任务包括从相关联的存储器位置清除一个或多个LHS度量实例，以及在聚合级别为新的RHS度量实例创建数据结构。因此，计算步骤可以包括顺序排序的子任务集合，即，用于清理度量数据和创建用于存储聚合级别度量实例的新数据结构的初始化子任务、用于生成聚合级别度量实例的聚合子任务、以及最后用于使用聚合级别度量实例作为参数来评估实际表达式的计算子任务。

[0036] 由于操作的独立性质，可以同时执行多个初始化子任务。类似地，可以并行执行涉及独立度量实例的多个聚合子任务。因此，取决于所涉及的计算类型，一些计算步骤可能不包括初始化和/或聚合子任务中的一个或两个。因此，计算规划的制定是动态的处理。在计算规划的最后，在最后的计算步骤之后，可以执行最终的清理任务。

[0037] 图4图示了根据实施例的并行计算规划402。如图4所示，基于依赖性和/或相交或

问题分区标准，计算规划402被分解成顺序计算步骤404至410的集合。

[0038] 图5图示了将第一顺序步骤404并行分区到分别包括并行化任务组502、504和506的三个子步骤的序列中。并行化任务组502执行初始化操作，该初始化操作包括清理任务508和数据结构创建任务510。清理任务508可以表示与正被评估的规则集合中的所有表达式相关联的度量清除任务，其可以被布置到初始化步骤502中以及并行执行。与正被评估的规则集合中的所有表达式相关联的数据结构或度量创建任务510也被布置到初始化子步骤502中，并与清理任务508并行执行。根据图5所示的本发明的实施例，初始化子步骤502必须在第二并行化任务组504的执行之前，该第二并行化任务组504涉及由聚合子任务512、514和516执行的聚合度量A、B和C。例如，必须清除指定的存储器块，并且必须在502处分配和初始化所需的存储器，用于在504处对度量A、B和C进行聚合之前存储度量A、B和C的聚合级别实例。

[0039] 如上所述，数据结构或度量实例创建任务510可以与度量清除任务508同时且并行地发生，以显著加速初始化处理。

[0040] 包括并行化任务组504的第二顺序子步骤涉及度量数据的聚合，具体而言是用于聚合度量A的聚合子任务512、用于聚合度量B的聚合子任务514和用于聚合度量C的聚合子任务516。聚合操作可能涉及对跨与规划系统相关联的多个商店和/或部门的所有库存单位(“SKU”)的所有数据求和。这可能是非常耗时的任务。因此，为可以被独立聚合的每个度量创建单独的聚合任务，并且并行运行所有此类聚合任务可以显著改善规则集合的评估时间。由于如图5所示分别用于聚合度量A、B和C的聚合子任务512、514和516涉及聚合不同独立度量的数据，因此它们可以被并行执行。

[0041] 在聚合步骤504完成后，一旦对规则集合中的一个或多个表达式的实际评估所需的数据可用，就可以执行计算任务组506中的计算任务518，以使用聚合级别度量值来评估规则集合中的表达式。不涉及共同组件的计算任务可以被并行执行。然后可以对计算步骤2、3、4重复相同的三个顺序子步骤集合。

[0042] 如上所述，初始化、聚合和计算子步骤遵循执行的顺序次序。但是，每个子步骤可以包括多个子任务，这些子任务与规则集合中的不同表达式相关联，并且被调度为并行执行。例如，初始化处理可能涉及创建多个子任务以清理规则集合中的一些或所有LHS度量实例，同时在与规则集合中的一些或所有表达式相关联的聚合级别上创建RHS度量实例。因此，可以将规则集合中的多个表达式的清理和创建任务布置到单个初始化子步骤中并且并行执行。类似地，聚合子步骤可以包括从规则集合中的所有表达式收集聚合的RHS度量实例，并且为可以并行运行的所有聚合任务创建多个子任务。还可以基于规则集合中表达式之间现有的依赖性和/或计算级别将评估一个或多个表达式的计算子步骤进行并行化。在当前批次中的所有计算完成之后，或者即使批次在执行中间失败，最终清理任务也可以运行。最终清理任务移除当前批次中创建的所有聚合度量实例。

[0043] 除了基于依赖性的并行化之外，并行化任务组中的每个顺序子步骤还可以按范围进行并行化。基于范围的并行化涉及在同一层次级别跨多个独立的域、实体或渠道(channel)(即，商店、部门等)将单个子任务的执行并行化。单个子任务可以包括多个表达式、聚合等。基于范围的并行化为计算规划添加了附加的并行化层，从而进一步加快了规则集合评估处理。

[0044] 图6图示了根据本发明的实施例的包括基于度量和基于范围的并行化的最终计算规划602。第一顺序计算步骤的并行化执行模式603被划分成三个并行化任务组的序列，用于分别执行初始化、聚合和计算操作的第一集合。并行化执行模式603包括与度量数据清理操作606和新的聚合级别度量实例创建操作608相关联的第一顺序子步骤604。并行化执行模式603的第一顺序子步骤604构成与规则集合中的所有基于规则的表达式相关联的所有初始化操作的并行执行。这包括可以跨规则集合被并行化的所有清理和新实例创建操作。并行化执行模式603的第二顺序子步骤或并行化任务组610包括用于度量A和B的聚合子任务，其进一步按渠道被分区到聚合与渠道1的度量A和B相关联的数据的并行化子任务组612、聚合与渠道2的度量A和B相关联的数据的并行化子任务组614、以及聚合与一个或多个渠道的度量C相关联的数据的并行化子任务组616中。

[0045] 应当注意的是，并行化子任务组612、614和616被调度为同时执行，如由与计算规划602的第一顺序计算步骤相关联的并行化执行模式603所指定的。因此，计算规划602中的第一顺序计算步骤的第二顺序子步骤610(聚合子步骤)首先按跨度量A、度量B和度量C的度量进行并行化，并且然后按范围进行并行化，以同时执行跨多个渠道的度量A、度量B和度量C的并行聚合。

[0046] 一般而言，例如，如果在计算域中有n个渠道，则可以创建n个并行任务来跨所有n个渠道并行聚合度量。图6中的计算规划602利用了两种主要的并行化方案，即，基于依赖性的并行化方案以及基于分区或基于范围的并行化方案。

[0047] 与计算规划602的第一顺序计算步骤相关联的并行化执行模式603还包括第三顺序子步骤618。并行化执行模式603的第三顺序子步骤618包括用于评估渠道1的表达式列表的并行化子任务组620、用于评估渠道2的表达式列表的并行化子任务组622、以及用于评估一个或多个其它渠道的表达式列表的并行化子任务组624。应当注意的是，并行化子任务组620、622和624被调度用于并行执行，如由与计算规划602的第一顺序计算步骤相关联的并行执行模式603所指定的。因此，计算规划602中的第一顺序计算步骤的第三顺序子步骤618(计算子步骤)首先按跨度量A、B和C的度量进行并行化，并且然后按范围进行并行化，以同时执行跨多个渠道的表达式列表的并行计算。然后可以对计算规划中的其它计算步骤执行相同集合或子集的并行化初始化、聚合和计算任务组，如626所示。

[0048] 根据本发明的实施例，为具有基于空的数组(array)的所有度量创建顺序聚合任务，如果聚合为标量则创建源范围的聚合任务，如果最外面的维度(outermost dimension)小则创建二维范围的聚合任务，并且最后按目的地的最外面维度创建目的地范围的聚合任务。

[0049] 根据本发明的实施例，根据以下规范来设置用于任何范围可并行化任务的范围信息：对于源范围的聚合，范围是按源数组的最外面维度来设置的。对于目的地范围的聚合，范围是按目的地数组的最外面维度来设置的。对于二维范围的聚合，范围是按目的地数组的最外面和中间维度来设置的。对于非标量计算，范围是按LHS度量实例的最外面维度来设置的。

[0050] 根据本发明的实施例，利用关于每个步骤的识别每个顺序步骤内的一个或多个可并行化任务集合(基于度量和/或基于范围)的信息以及关于每个任务的识别每个任务内的一个或多个可并行化子任务集合(基于度量和/或基于范围)的信息，可以将包括基于度量

(基于表达式)和基于范围的并行化方案的并行化计算规划平坦化为顺序步骤的线性表达式。

[0051] 根据本发明的实施例,包括顺序计算步骤的线性集合以及相关联的计算步骤并行化信息以使得能够按顺序更快评估一个或多个规则集合的平坦化计算规划可以被存储在诸如NoSQL数据库的存储模块上并且在RPAS平台上实现。

[0052] 描述本发明的实施例的一种特定实现的样本伪代码如下:

```
// 生成并行计算规划
// 输入是由规则引擎预先排序的表达式列表
generate_plan(expr_list)
{
```

[0053]

```
 for (expr_list 中的 expr)
 {
    if expr 不取决于其它，then
    {
        从 expr_list 中取出这个 expr out
```

```
            将这个expr 放在 independ_expr_list 中
        }
    }

    if independ_expr_list 不为空，then
    {

generate_measure_based_parallel_plan_for_ind_expressions(
            independent_expr_list
        )
    }

    generate_range_based_parallel_plan(expr_list)
    }

    //提取表达式的范围可并行化维度
[0054]  dimension_name
    extract_parallelizable_level(expr)
    {
     if (expr 是特殊表达式)
     {
        返回这个expr 的rangeableIntx 的最外面维度
     }
    else
     {
        表达式是常量表达式?
        {
            //  常量表达式被认为是由空串识别的标量
            返回 ""
        }
        else
        {
            返回LHS 度量实例的交集的最外面维度
```

```
        }
      }
    }

    generate_measure_based_parallel_plan_for_ind_expressions(
    independent_expr_list)
    {
     // 按并行化的级别分解所有表达式，创建维度名称到表达式
列表的映射
     for (independent_expr_list 中的 expr)
     {
       parallelizable_dim = extract_parallelizable_level (expr)
       将 expr 添加到 parallelizable_dim 的映射条目
     }

     for 映射中的每个条目
     {
       令 dim_name 为条目中的键
       令 sub_expr_list 为条目的表达式列表

       if sub_expr_list 的大小太大
       我们进一步将 sub_expr_list 分解为多个较小的
sub_expr_list
       for 每个较小的 sub_expr_list 或仅 sub_expr_list 本身
       {
        generate_plan_for_same_level_same_group(dim_name ,
sub_expr_list)
       }
     }
    }

    generate_range_based_parallel_plan(expr_list)
```

[0055]

```
{
 令 outerdim_divided_list 为对列表 (
    维度名称,
    表达式列表)

 for (expr_list 中的 expr)
 {
    outer_dim_name = extract_parallelizable_level (expr)

    if outer_dim_name 与 outerdim_divided_list 中的最后一个
元素的 dim 名称不同
    {
        将新对 (outer_dim_name , 空 expr 表) 附加到
outerdim_divided_list
    }
```

[0056]

```
    将 expr 附加到 outerdim_divided_list 中的最后一个元素的
表达式列表
    }

    for outerdim_divided_list 中的每对(dim_name, expr_list)
    {
    generate_plan_for_same_level(dim_name, expr_list)
    }

 存储所生成的规划并返回
 }

// 生成范围并行化的计算规划
generate_plan_for_same_level(parallelizable_dim, expr_list)
{
 if (parallelizable_dim 为空)
```

```
        {
          // 处理标量评估情况

 generate_plan_for_same_level_same_group(parallelizable_dim ,
expr_list)
        }
        else
        {
          // 按范围并行化，首先基于障碍计算进行分解
          令expr_list_list 为表达式列表中的列表
          for (expr_list 中的expr)
          {
              令current_expr_list 为expr_list_list 中的最后一个元素

              首先检查障碍计算
[0057]        收集这个expr 的RHS (右手侧)度量实例
              if 任何RHS 度量实例满足:
                  1. RHS 是通过 current_expr_list 中的先前表达式
计算的
                  2. 并且在高于parallelizable_dim 的聚合级别
              then
              {
                  // Expr 必须在 current_expr_list 中的所有表达式
被评估之后进行评估
                  将新表达式列表附加到expr_list_list
                  将current_expr_list 重置为刚创建的新表达式列表
              }

              将expr 附加到current_expr_list
          }
```

```
        for (expr_list_list 中的每个 sub_list)
        {

 generate_plan_for_same_level_same_group(parallelizable_dim ,
sub_list)
        }
      }
     }

    generate_plan_for_same_level_same_group(parallelizable_dim ,
expr_list)
    {
     // 首先收集需要聚合或清理的所有度量实例
     令 lhs_set 为 LHS 度量实例集合
     令 agg_map 为从聚合的交集映射到度量实例的映射
```

[0058]

```
     for (expr_list 中的 expr)
     {
        收集 expr 中的度量实例,
        if 该度量实例处于聚合级别
        向 agg_map 添加条目，键是聚合交集，
        值是度量实例集合

        将 expr 中的 LHS 度量实例收集到 lhs_set
     }

     // 处理不重要的情况 – 不需要进一步的并行化
     if agg_map 为空，并且 parallelizable_dim 也为空
     {
        创建新的计算步骤
        在该步骤中创建顺序计算任务
        将 expr_list 添加到这个新任务
```

```
        顺序计算任务将简单地评估所有表达式
        在顺序模式下，不需要进一步的并行化

        返回;
    }

    // 处理聚合
    if (agg_map 不为空，或 lhs_set 不为空)
    {
        创建新计算步骤 (init_step)，将它附加到当前规划
        还创建进行度量创建和清理的新任务(init_task)
        将 init_task 添加到 init_step

        for (lhs_set 和 agg_map 中的所有度量实例)
        {
            将度量实例添加到 init_task
            初始任务在执行时，如果度量实例不存在，则将创建它
            或者如果它已经存在则清理它
        }
    }

    // 创建聚合步骤
    令 seq_agg_map、src_range_agg_map、dst_range_agg_map、twodim_range_agg_map 为 4 个映射，每个映射与度量实例集合相交
    if agg_map 不为空
    {
    for 每个交集，在 agg_map 中设置度量实例对
        if 聚合的度量没有数据
        {
```

[0059]

```
        // 如果不涉及数据则使用常规顺序聚合
          将这个交集及其相关联的度量集合添加到seq_agg_map
        }
        else if 交集是标量的，如将所有东西聚合到所有产品/所
有位置
        {
          // 使用源范围的聚合
              将这个交集及其相关联的度量集合添加到
          src_range_agg_map
        }
        else if 最外面维度具有由系统阈值定义的少量位置
          但下一个外部维度具有较大数量的位置
        {
          // 使用两个Dim 范围聚合
          将这个交集及其相关联的度量集合添加到
[0060] twodim_range_agg_map
        }
        else
        {
          // 默认情况按目的地的范围
          将该交集及其相关联的度量集合添加到
dst_range_agg_map
        }
      }

      创建新计算步骤 (agg_step)，将它附加到当前规划
      if seq_agg_map 不为空
      {
        创建顺序聚合任务，将 seq_agg_map 中的所有度量放到
该任务中
        将该任务添加到当前聚合步骤
      }
```

[0061]

```
if src_range_agg_map 、 dst_range_agg_map 、
twodim_range_agg_map 不为空
{
  for 三个映射中的每个(交集，度量对集合)
  {
    if 对于每个交集，度量集合太大
    将度量集合分解成较小的集合

    为 src_range_agg_map 中的条目创建源范围聚合任务
      或为 dst_range_agg_map 中的条目创建目的地范围
聚合任务
      或为 twodim_range_agg_map 中的条目创建 twodim
范围聚合任务
    将度量聚合添加到这个任务
    将该任务添加到当前聚合步骤
    通过向任务添加分区信息进一步按范围将任务并行化
  }
}

创建计算步骤并将该步骤附加到当前规划
if (parallelizable_dim 为空)
{
  创建顺序计算任务
  将 expr_list 添加到顺序计算任务
  将该任务添加到计算步骤
}
else
{
  创建范围并行化计算任务
  将 expr_list 添加到基于范围的计算任务
```

```
        将该任务添加到计算步骤
        通过向任务添加分区信息进一步按范围将任务并行化
[0062]  }
    }
```

[0063] 如所公开的，实施例允许基于表达式属性的计算处理的并行化，通过同时基于范围的并行化计算实现了进一步的性能增强。

[0064] 在本文中具体图示和/或描述了若干实施例。但是，将认识到的是，在不脱离本发明的精神和预期范围的情况下，以上教导涵盖了所公开的实施例的修改和变型并且这些修改和变型在所附权利要求的范围内。

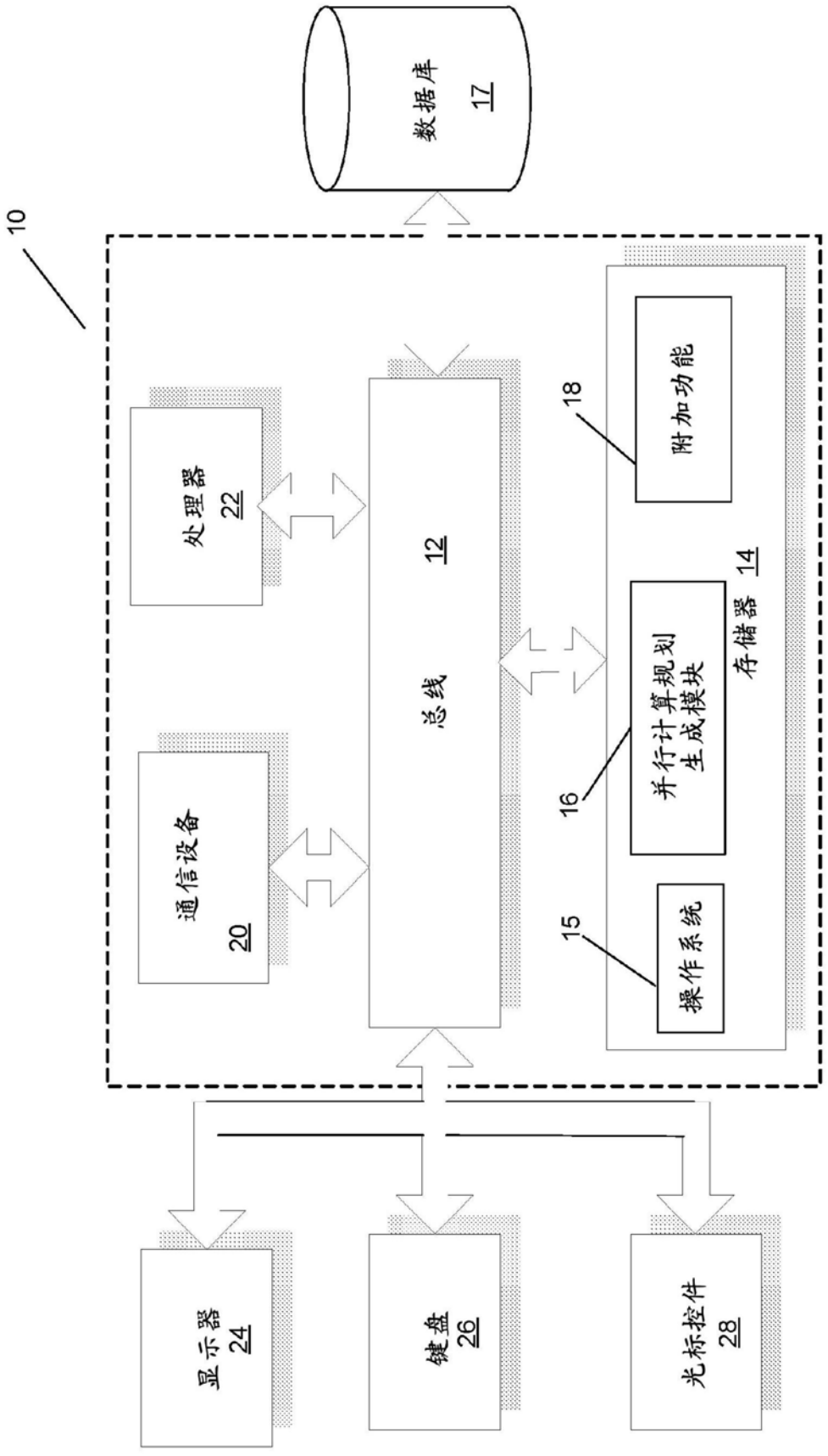

数据库
17
10
处理器
22
总线
12
附加功能
18
存储器
14
并行计算规划
生成模块
16
通信设备
20
操作系统
15
显示器
24
键盘
26
光标控件
28

图1

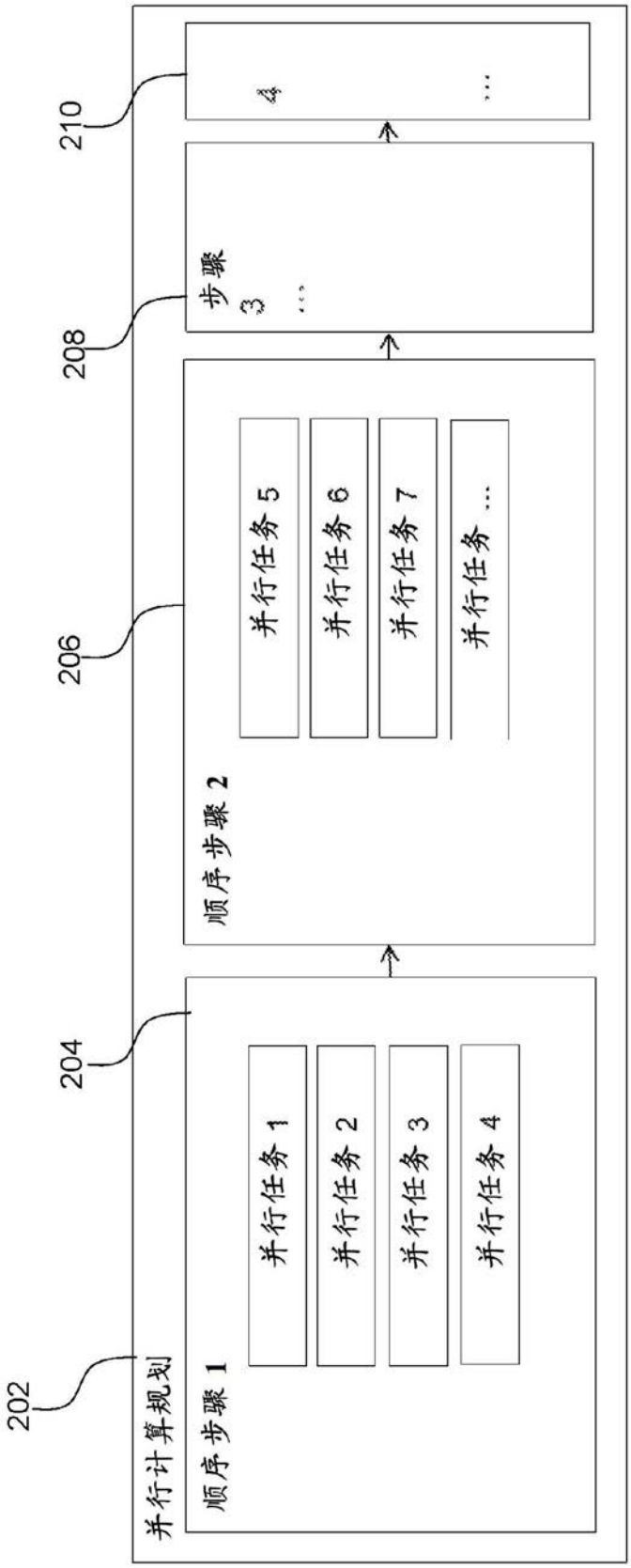

并行计算规划
顺序步骤 1
并行任务 1
并行任务 2
并行任务 3
并行任务 4
顺序步骤 2
并行任务 5
并行任务 6
并行任务 7
并行任务 ...
步骤
3
...
4
...
202
204
206
208
210

图2

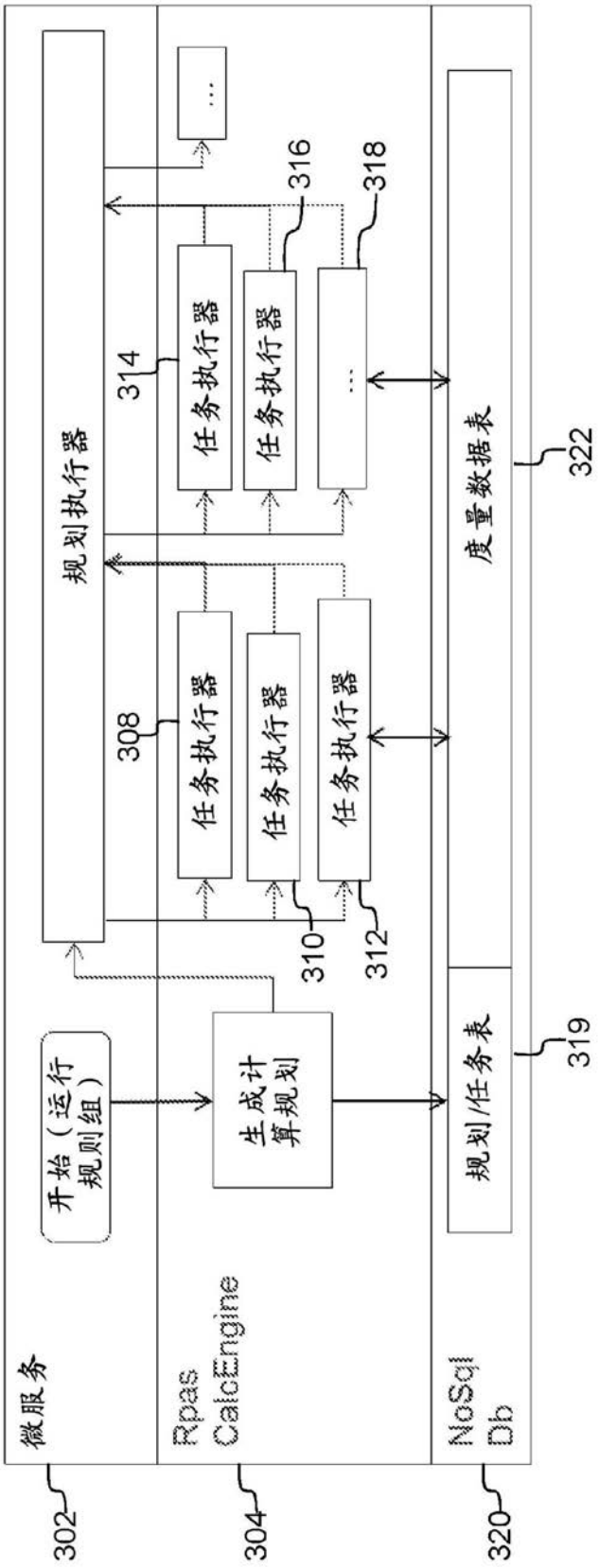

微服务
302
开始（运行规则组）
规划执行器
生成计算规则
Rpas CalcEngine
304
308
310
312
314
316
318
任务执行器
任务执行器
任务执行器
...
...
NoSql Db
320
规划/任务表
319
度量数据表
322

图3

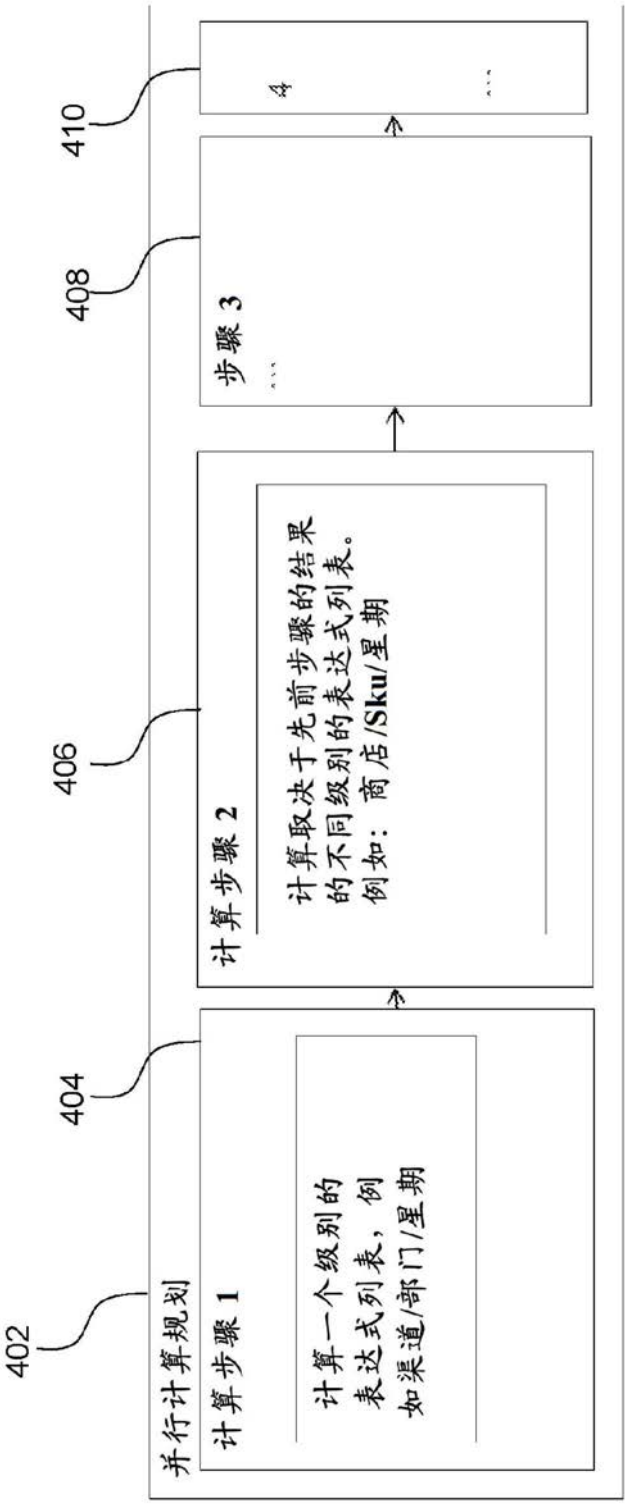

402
并行计算规划
404
计算步骤 1
计算一个级别的
表达式列表，例
如渠道/部门/星期
406
计算步骤 2
计算取决于先前步骤的结果
的不同级别的表达式列表。
例如：商店/Sku/星期
408
步骤 3
410
4

图4

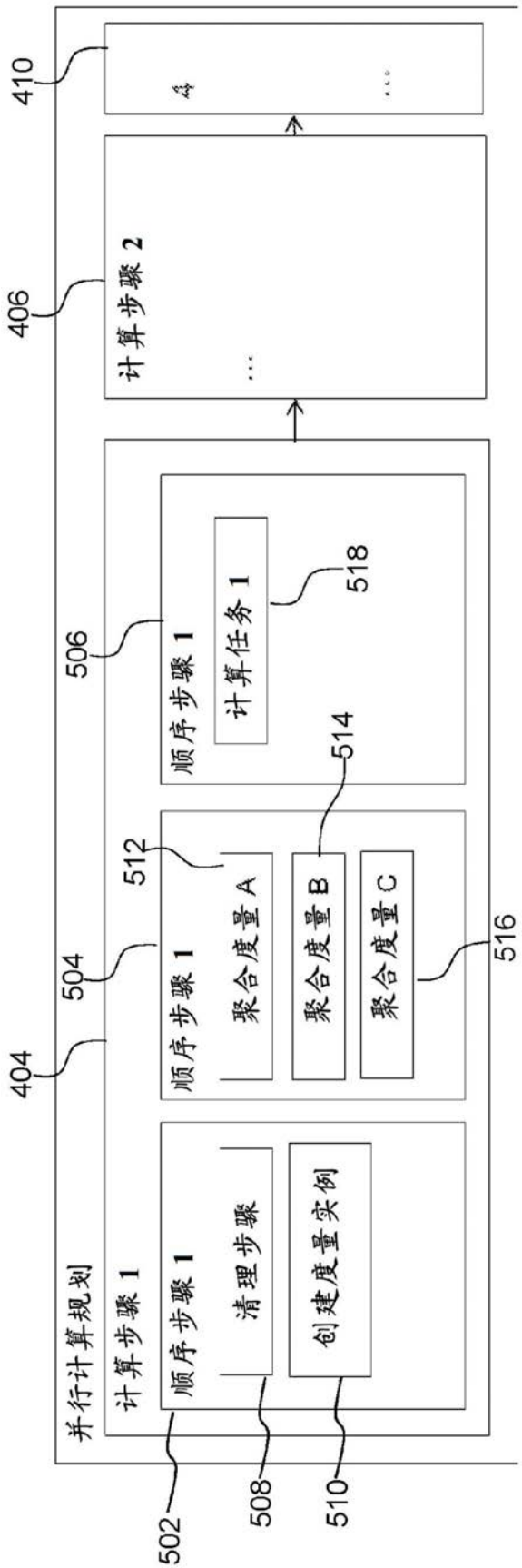

并行计算规划
计算步骤 1
顺序步骤 1
清理步骤
创建度量实例
顺序步骤 1
聚合度量 A
聚合度量 B
聚合度量 C
顺序步骤 1
计算任务 1
计算步骤 2
404
502
504
506
508
510
512
514
516
518
406
410

图5

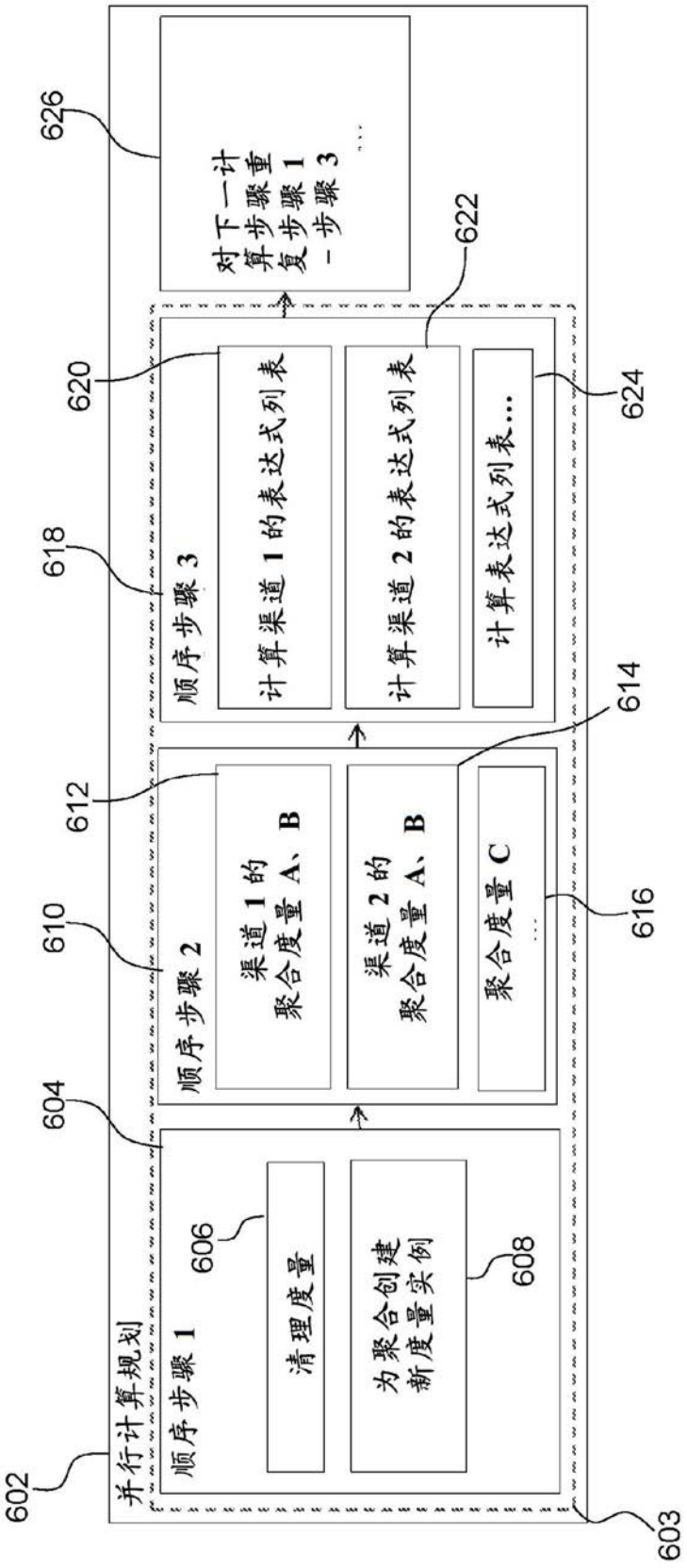

并行计算规划
602
603
顺序步骤 1
604
606
清理度量
608
为聚合创建
新度量实例
顺序步骤 2
610
612
渠道 1 的
聚合度量 A、B
渠道 2 的
聚合度量 A、B
614
聚合度量 C
...
616
顺序步骤 3
618
620
622
计算渠道 1 的表达式列表
计算渠道 2 的表达式列表
计算表达式列表...
624
626
对下一计
算步骤重
复步骤 1
- 步骤 3
...

图6