



(12) 发明专利

(10) 授权公告号 CN 102804175 B

(45) 授权公告日 2016. 04. 06

(21) 申请号 201180015863. 0

(56) 对比文件

(22) 申请日 2011. 03. 24

US 6763388 B1, 2004. 07. 13,

(30) 优先权数据

US 2009/0070413 A1, 2009. 03. 12,

12/732088 2010. 03. 25 US

US 5557722 A, 1996. 09. 17,

(85) PCT国际申请进入国家阶段日

CN 1679028 A, 2005. 10. 05,

2012. 09. 25

US 2001/0032216 A1, 2001. 10. 18,

(86) PCT国际申请的申请数据

US 2006/0092454 A1, 2006. 05. 04,

PCT/US2011/029722 2011. 03. 24

审查员 康厚萍

(87) PCT国际申请的公布数据

W02011/119792 EN 2011. 09. 29

(73) 专利权人 微软技术许可有限责任公司

地址 美国华盛顿州

(72) 发明人 E. N. 韦塞罗夫 R. P. 阿塔纳索夫

M. J. 乔尔森

(74) 专利代理机构 中国专利代理(香港)有限公司 72001

代理人 李舒 汪扬

(51) Int. Cl.

G06F 17/21(2006. 01)

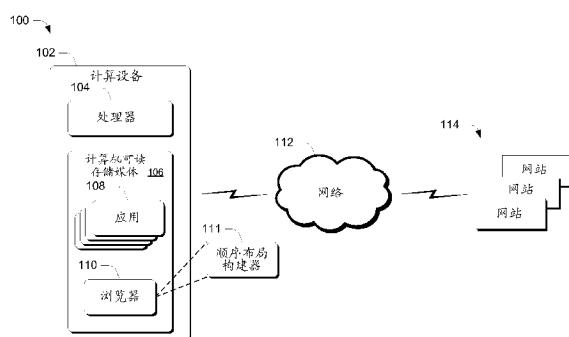
权利要求书3页 说明书9页 附图7页

(54) 发明名称

顺序布局构建器

(57) 摘要

描述了布局处理技术，所述布局处理技术允许实施结构化文档的处理，以及允许以顺序的方式完成相关联的布局。布局过程被分成可以顺序地执行的多个更小的步骤。在至少一些实施例中，布局构建器被结合栈使用以控制在结构化文档的结构层次上的布局执行。不可变的数据结构被使用，并且其允许顺序布局操作的并行执行。在至少一些实施例中，可以完成所述布局序列的部分执行来产生可用的数据。



1. 一种用于对结构化文档进行布局处理的计算机实施的方法，包括：

接收结构化文档；

在结构化文档内遇到打开标签；

创建与所述打开标签相关联的框构建器，该框构建器是多个不同类型的框构建器之一，该多个不同类型的框构建器中的每一个被配置成构建特定类型的内容；

使用被配置来统一原始内容和计算的内容的布局节点，该计算的内容是仅依赖于内容属性而不依赖于空间特性的数据且是按需要预先计算的；

通过以下方式来创建在随后的和并行的布局构建过程的不同位置中可再次使用的相关联的布局框，即：组织相关联的布局框来包含内部定位的内容和大小调整信息而不包含定位信息；以及

使用所述框构建器来顺序地处理所述结构化文档，以及用可再现的内容填充所述布局框。

2. 权利要求 1 的方法，还包括确定所述结构化文档中的随后的标签是否是与所述打开标签相关联的关闭标签，并且如果是的话，使得所述布局框插入到显示树中。

3. 权利要求 1 的方法，还包括：

确定所述结构化文档中的随后的标签是否是与所述打开标签相关联的关闭标签，并且如果是的话，使得所述布局框插入到显示树中；和

再现所述显示树中的内容。

4. 权利要求 1 的方法，还包括：

确定所述结构化文档中的随后的标签是否是与所述打开标签相关联的关闭标签，并且如果是的话，使得所述布局框插入到显示树中；和

再现所述显示树中的内容，其中所述再现是在所述结构化文档被完全地顺序处理之前执行的。

5. 权利要求 1 的方法，还包括确定所述结构化文档中的随后的标签是否是与所述打开标签相关联的关闭标签，并且如果不是的话，继续顺序地处理所述结构化文档，以有效于用可再现的内容填充所述布局框。

6. 权利要求 1 的方法，其中所述使用包括使用被组织为分层地嵌套的框构建器的数据栈来存储临时数据。

7. 权利要求 1 的方法，其中所述框构建器包括特定于所述打开标签的布局构建逻辑。

8. 权利要求 1 的方法，其中所述框构建器被组织为有限状态机。

9. 权利要求 1 的方法，其中所述布局框是可再次使用的。

10. 权利要求 1 的方法，其中所述创建是由布局构建器执行的，以及其中所述结构化文档的顺序处理利用并发的布局构建器而并行地发生。

11. 一种用于对 HTML 文档进行布局处理的计算机实施的方法，所述方法包括：

接收 HTML 文档；

在 HTML 文档内遇到打开标签；

创建与所述打开标签相关联的框构建器，该框构建器是多个不同类型的框构建器之一，该多个不同类型的框构建器中的每一个被配置成构建特定类型的内容；

使用被配置来统一原始内容和计算的内容的布局节点，该计算的内容是仅依赖于内容

属性而不依赖于空间特性的数据且是按需要预先计算的；

通过以下方式来创建在随后的和并行的布局构建过程的不同位置中可再次使用的相关联的布局框，即：组织相关联的布局框来包含内部定位的内容和大小调整信息而不包含定位信息；

使用所述框构建器来顺序地处理所述 HTML 文档，以及用可再现的内容填充所述布局框；

确定所述 HTML 文档中的随后的标签是否是与所述打开标签相关联的关闭标签，并且如果不是的话，继续顺序地处理所述 HTML 文档，以有效于用可再现的内容填充所述布局框；和

如果所述随后的标签是与所述打开标签相关联的关闭标签，则使得所述布局框插入到显示树中。

12. 权利要求 11 的方法，还包括再现所述显示树中的内容。

13. 权利要求 11 的方法，还包括在所述 HTML 文档被完全地顺序处理之前再现所述显示树中的内容。

14. 权利要求 11 的方法，其中所述使用包括使用被组织为分层地嵌套的框构建器的数据栈来存储临时数据。

15. 权利要求 11 的方法，其中所述框构建器包括特定于所述打开标签的布局构建逻辑。

16. 一种用于对结构化文档进行布局处理的计算机实施的系统，包括：

用于接收结构化文档的装置；

用于在结构化文档内遇到打开标签的装置；

用于创建与所述打开标签相关联的框构建器的装置，该框构建器是多个不同类型的框构建器之一，该多个不同类型的框构建器中的每一个被配置成构建特定类型的内容；

用于使用被配置来统一原始内容和计算的内容的布局节点的装置，该计算的内容是仅依赖于内容属性且不依赖于空间特性的数据且是按需要预先计算的；

用于通过以下方式来创建在随后的和并行的布局构建过程的不同位置中可再次使用的相关联的布局框的装置，即：组织相关联的布局框来包含内部定位的内容和大小调整信息而不包含定位信息；以及

用于使用所述框构建器来顺序地处理所述结构化文档，以及用可再现的内容填充所述布局框的装置。

17. 一种用于对 HTML 文档进行布局处理的计算机实施的系统，包括：

用于接收 HTML 文档的装置；

用于在 HTML 文档内遇到打开标签的装置；

用于创建与所述打开标签相关联的框构建器的装置，该框构建器是多个不同类型的框构建器之一，该多个不同类型的框构建器中的每一个被配置成构建特定类型的内容；

用于使用被配置来统一原始内容和计算的内容的布局节点的装置，该计算的内容是仅依赖于内容属性且不依赖于空间特性的数据且是按需要预先计算的；

用于通过以下方式来创建在随后的和并行的布局构建过程的不同位置中可再次使用的相关联的布局框的装置，即：组织相关联的布局框来包含内部定位的内容和大小调整信

息而不包含定位信息；

用于使用所述框构建器来顺序地处理所述 HTML 文档, 以及用可再现的内容填充所述布局框的装置；

用于进行如下操作的装置, 即 : 确定所述 HTML 文档中的随后的标签是否是与所述打开标签相关联的关闭标签, 并且如果不是的话, 继续顺序地处理所述 HTML 文档, 以有效于用可再现的内容填充所述布局框 ; 和如果所述随后的标签是与所述打开标签相关联的关闭标签, 则使得所述布局框插入到显示树中。

顺序布局构建器

背景技术

[0001] 使用 Web 浏览器再现结构化的 Web 内容,诸如 HTML 和 CSS 内容,典型地牵涉到处理包括标记的结构化文档以确定内容的布局(layout),使得其可以由 Web 浏览器呈现。这样的处理可以包括内容位置、大小和形状计算,以便从所述标记确定相关联的内容将如何被呈现在屏幕或显示器上。对于可用性和用户感知而言,期望的是布局和呈现算法按照可适用的标准快速并正确地工作。

[0002] 在整个业界,布局处理普遍以递归方式执行。例如,这样的处理可以包括在 HTML 标记的根节点上发起处理操作,并以递归的方式贯穿该标记向下处理到叶节点。该处理典型地在 HTML 文档被呈现之前在整个 HTML 文档上进行。此外,现代的多核体系结构可以准许处理被并行地执行。然而,因为布局处理的递归性质,从并行处理所得到的优势不能被充分利用。

[0003] 布局处理的递归性质也可能导致处理中的重复,这进而又可能使性能和用户体验降级。

发明内容

[0004] 本概要被提供来以简化的形式介绍概念的选择,这些概念还将在下面的详细说明中进行描述。本概要既不打算确认所要求保护的主题的关键特征或必要特征,也不打算被使用来限制所要求保护的主题的范围。

[0005] 按照一个或多个实施例,对结构化文档的布局处理以顺序的方式实施(conduct)。布局过程被分成可以顺序地执行的多个更小的步骤。在至少一些实施例中,布局构建器被结合栈使用以控制在文档的结构层次的各层次上的布局执行。不可变的数据结构被使用,并且其允许顺序布局操作的并行执行。在至少一些实施例中,可以完成布局序列的部分执行来产生可用的数据。

附图说明

[0006] 在全部附图中,使用相同的标号来标注同样的特征。

[0007] 图 1 图示了按照一个或多个实施例的、在其中可以利用这里描述的各种原理的操作环境。

[0008] 图 2 图示了按照一个或多个实施例的示例性体系结构。

[0009] 图 3 描述了按照一个或多个实施例的、在顺序布局构建过程中的布局空间分配协议。

[0010] 图 4 是描述按照一个或多个实施例的方法中的步骤的流程图。

[0011] 图 5 图示了按照一个或多个实施例的布局框的类层次。

[0012] 图 6 图示了按照一个或多个实施例的框构建器的类层次。

[0013] 图 7 图示了可以被使用来实施一个或多个实施例的示例性系统。

具体实施方式

[0014] 综述

[0015] 按照一个或多个实施例,布局处理以顺序的方式实施。布局过程被分成可以顺序地执行的多个更小的步骤。在至少一些实施例中,布局构建器被结合栈使用以控制在结构化文档的结构层次的各层次上的布局执行。不可变的数据结构被使用,并且允许顺序布局操作的并行执行。在至少一些实施例中,可以完成布局序列的部分执行来产生可用的数据。这里描述的技术可以结合任意类型的结构化文档一起被使用,作为例子而不是限制,结构化文档诸如是 HTML 文档。

[0016] 在一个或多个实施例中,在布局构建器的影响下,顺序布局处理被组织为按一组顺序的步骤执行的非递归树径(tree walk),所述的一组顺序的步骤诸如像“进入块”、“退出块”、“构建行”等等。一个个步骤可以具有恒定的时间来执行,并且整个过程可以在一个个步骤后被停止,以便产生可用于再现和用户交互的部分布局。

[0017] 在一个或多个实施例中,通过把在布局构建期间使用的临时数据与提供用于再现和交互的布局结果的持久性数据进行分离,而将布局构建数据结构组织成达到高效紧致性。临时数据由一组布局构建器代表,每种布局一个布局构建器。布局构建器是在布局处理期间临时存在的临时对象。另一方面,持久性数据被表示为存储在最终得到的显示树中的布局框,显示树可以被处理来呈现相关联的内容。这样做可以比过去更加高效地利用存储器资源。例如,在 HTML 处理的上下文中,在任一特定时间存在的布局构建器的数量由 HTML 文档的深度来限定,而不是由 HTML 文档的宽度来限定。因此,由布局构建器代表的临时数据不必驻留在最终得到的显示树中。

[0018] 另外,被处理的数据和处理算法都可以可伸缩的方式被组织。这允许任意的布局片段的执行。这种顺序可伸缩性导致了对于可以被使用于回溯针对滚动、分页、栏目化等等的尝试的算法的改进性能。

[0019] 此外,在至少一些实施例中,布局结果由被设计为不可变的布局框所代表。这使得布局框能够在动态情景中被再次使用,动态情景诸如是渐增的部分更新、并行布局、渐进式再现以及回溯布局执行。

[0020] 此外,在至少一些实施例中,在规定的、计算的以及已用值和结构之间强制分离。这允许跨整个布局流水线的、部分的和可再次使用的计算。在 HTML 上下文中,规定的值涉及在原始的 HTML 标记中如何规定值。这些值可能是矛盾的、不完整的和无效的。这些不一致性可使规定的值很难利用。计算的值是与归一化属性和归一化值的组合相关联的。这些值是不矛盾的且是可靠的。已用值代表在布局框中存储的值。

[0021] 在随后的讨论中,提供了标题为“操作环境”的章节,其描述了一种在其中可以利用一个或多个实施例的环境。在这之后,标题为“示例性体系结构”的章节描述了按照一个或多个实施例的示例性体系结构。接着,标题为“示例性方法”的章节描述了按照一个或多个实施例的示例性方法。在这之后,标题为“实施细节”的章节描述了按照一个或多个实施例的实施细节。最后,标题为“示例性系统”的章节描述了可以被使用来实施一个或多个实施例的示例性系统。

[0022] 操作环境

[0023] 图 1 图示了按照一个或多个实施例的操作环境,总地标为 100。环境 100 包括计

算设备 102, 其具有一个或多个处理器 104、一个或多个计算机可读存储媒体 106、和驻留在计算机可读存储媒体上并可由处理器执行的一个或多个应用 108。作为例子而不是限制, 所述计算机可读存储媒体可以包括典型地与计算设备相关联的、所有形式的易失性和非易失性存储器和 / 或存储媒体。这样的媒体可以包括 ROM、RAM、闪存、硬盘、可拆卸媒体等等。下面在图 7 中显示和描述了计算设备的一个特定的例子。

[0024] 另外, 计算设备 102 包括以 web 浏览器 110 形式的软件应用。可以使用任何适当的 web 浏览器, 其例子是从本文档的受让人和其它方可得到的。另外, 计算机可读存储媒体 106 可以包括顺序布局构建器 111, 其按上面和下面所描述的那样进行操作。顺序布局构建器 111 可以被实施为可由应用 108 和浏览器 110 使用的独立的组件。替换地或另外地, 顺序布局构建器 111 可以被实施为应用 108 和 / 或浏览器 110 的一部分。

[0025] 在操作中, 顺序布局构建器 111 允许诸如 HTML 文档这样的结构化文档的顺序处理, 以及允许布局以顺序的方式被计算。顺序布局构建器使用高效的体系结构, 该体系结构利用了布局构建器的用来监督布局处理的过程的用途、利用了熟知它们自己的针对内容处理的要求的框构建器类型、以及利用了由框构建器类型的实例构建且持有最终要在显示器或屏幕上再现的数据的相关联的布局框。

[0026] 另外, 环境 100 包括诸如因特网那样的网络 112, 以及可以从其接收内容和向其发送内容的一个或多个网站 114。这样的内容可以包括诸如 HTML 文档的结构化文档, 以及可以由顺序布局构建器 111 如上面和下面描述的在其上进行操作的其它 web 内容。

[0027] 计算设备 102 可以被具体化为任何适当的计算设备, 作为例子而不是限制, 诸如是台式计算机, 便携式计算机, 比如个人数字助理(PDA)、蜂窝电话那样的手持式计算机, 等等。

[0028] 已经描述了示例性操作环境后, 现在考虑可以被使用来顺序地处理诸如 HTML 的结构化文档的示例性体系结构的讨论。在下面的讨论中, 使用 HTML 做为结构化文档的例子。应当理解和明白, 下面描述的技术可以结合其它类型的结构化文档一起被利用, 而不背离所要求保护的主题的精神和范围。

[0029] 示例性体系结构

[0030] 图 2 图示了按照一个或多个实施例的示例性体系结构, 总地标为 200。在这个特定的例子中, 体系结构 200 包括布局构建器 202、框构建器 204、布局框 206、容器框 208 和行框 210。另外, 体系结构 200 包括可以在 HTML 的顺序布局处理期间采取的动作 212、214、216 和 218 的表示。该体系结构被使用来构建布局框的显示树, 其中布局框包含内容或数据, 诸如文本、图像或图形元素, 比如将要在屏幕或显示器上再现的顺序矢量图形(Svg)元素。

[0031] 在图示的和描述的实施例中, 布局构建器 202 构成监督或组织整个顺序布局处理的布局引擎或对象。布局构建器 202 负责实例化布局框 206, 然后与框构建器 204 进行通信以务必使(see)布局框 206 被填充以要在屏幕或显示器上再现的内容。框构建器 204 是负责构建特定类型的布局框的对象。特别地, 有不同类型的框构建器, 在 204 上只显示了其中的一个。有用于构建表、图像、多色、行等等的框构建器。框构建器被特定地配置成构建特定的一段内容。如本领域的技术人员所理解的, 不同类型的内容具有不同的布局要求。每个不同类型的框构建器熟知针对它的相关联的内容类型的不同布局要求, 并能构建相对应的一段内容。

[0032] 在图示的和描述的实施例中,框构建器 204 是代表这些不同类型的构建器的共同属性和方法的抽象类,并因此定义了框构建的通用联系(contact)。

[0033] 布局构建器 202 接收 HTML 文档并且顺序地移动穿过该文档,与框构建器 204 进行通信,框构建器 204 进而又用它的内容来填充布局框 206。在这个特定的例子中,容器框 208 和行框 210 代表布局框 206 的子类别或子类。行框 210 代表对应于一行文本的个别情形。容器框 208 是可包含诸如容器框和行框这样的其它框的框。诸如图像和 Svg 图元这样的内容被作为容器框对待。

[0034] 动作 212、214、216 和 218 构成代表在顺序布局处理期间发生的一系列基本构建步骤的动作或函数。这里,有四个动作 :BuildLine(f)、EnterBlock(f)、ReEnterBlock(f) 和 ExitBox(f)。所述函数加有代表布局构建器的缩写“LB”的前缀,因为它们被实施为这个类的方法。针对这些方法中的每个方法的主要参数是当前的框构建器,其在图中由从框构建器 204 朝向所述方法中的每个的数据流箭头来反映。

[0035] BuildLine(f)

[0036] 这个方法对应于在其上构建一行文本的布局构建器 202 的基本步骤。这是可以由布局构建器构建的内容的最小原子部分。要指出的是,所有可能的嵌套的块,诸如内联块 (inline block)、浮动物 (floater)、或锚定在该行的绝对定位的块,并不在 BuildLine 函数内部递归地构建。而是,它们作为该行的兄弟 (sibling) 经由 EnterBlock/ExitBox 回调而被单独地构建。

[0037] EnterBlock(f)

[0038] 当布局构建器 202 遇到诸如 DIV 或 TABLE 的某个块元素的打开标签时执行这个方法。

[0039] ReEnterBlock(f)

[0040] 在块被前一页上的分页符中断后,当该块在后一页上被进入时,调用这个方法。

[0041] ExitBox(f)

[0042] 当布局构建器 202 到达块元素的结束标签时,或在当分段的(分页的)空间在页边界上被中断时的情形下,执行这个方法。

[0043] 在操作中,布局处理被组织为对应于 HTML 的基本结构单元(即,纯文本的元素标签或顺串(run))的一系列步骤。布局构建器 202 接收 HTML 文档且一个接一个地读取这些单元,并执行适当的动作。当布局构建器遇到打开标签时,它识别其类型、创建适当的类型特定的框构建器(比如框构建器 204)、以及通过将所述框构建器推给构建栈来激活该框构建器。然后所述类型特定的框构建器将控制针对特定元素构建布局内容的过程。当布局构建器遇到元素的关闭标签时,它与活动的、类型特定的框构建器通信以使得最终得到的布局框被完成并被附连到显示树上。在这个过程期间,当遇到一行文本时,行框形式的布局框(比如行框 210)被创建并被插入到显示树中。

[0044] 图 3 在某些方面与图 2 类似,其描述了按照一个或多个实施例的、在顺序布局构建过程中的布局空间分配协议 300。该图也代表可扩展性模型。图 3 描述了在通用布局构建器和多个特定框构建器之间的协议。可扩展性通过简单地将新的框构建器类型插入到协议中,而在想要新的布局类型的事件中变得容易。如将在下面更详细地描述的,所述协议使用了一组虚拟函数,该组虚拟函数被定义为框构建器的抽象类。不同类型的框构建器可以不

同地实施这些虚拟函数。从布局构建器的观点来看,这些函数中仅有少数函数看起来是和布局构建器一样的。如上面所指出的,它们的实现取决于框构建器类型而不同。当布局构建器贯穿 HTML 进行处理时,以及当它遇到标签和其它内容时,它在相关联的框构建器上调用相应的函数。当布局构建器移动穿过特定标签的内容时,它可以在框构建器上酌情调用这些不同的函数,使框构建器能执行它的构造,以使得相关联的容器框被构建并为显示作准备。然而,在各段 HTML 内容之间移动的过程由相关联的个体的框构建器控制,所述框构建器明确地熟知所述内容的结构。

[0045] 在图示的和描述的实施例中,协议 300 包括布局构建器 302、ContainerBox.BoxBuilder(容器框 . 框构建器) 304、ContainerBox.BoxBuilder 306 和容器框 308。

[0046] 如以上所讨论的,布局构建器 302 代表组织整个布局构建过程的对象。布局构建器 302 与由图中的 ContainerBox.BoxBuilder 304 代表的各个框构建器一起工作。如以上所指出的,ContainerBox.BoxBuilder 是一种抽象类,其定义框构建器的许多具体的子类型要遵循的总合同(general contract)。它定义了布局构建器 302 在不同的构建步骤期间调用的一组虚拟回调,这里被描绘为列举的椭圆。

[0047] 下表描述了在椭圆的列举和它的相关联的虚拟函数之间的关联。在表的下面,描述了每个虚拟函数。

[0048]

椭圆列举	虚拟函数名
1	CBB. Constructor
2	CBB. MoveToNextPosition
3	CBB. GetChildBoxToReuse
4	CBB. OnChildBoxReuse
5	CBB. OnChildBoxEntry
6	CBB. OnChildBoxExit
7	CBB. Destructor
8	CBB. InitializeBoxSizing
9	CBB. CompleteBoxSizing

[0049] 表 1

[0050] CBB. Constructor

[0051] 该虚拟函数是回调,其在 ContainerBox.BoxBuilder 304 被创建时被调用来初始化它,并将它置于当前的布局构建过程的上下文中。

[0052] CBB. MoveToNextPosition

[0053] 该虚拟函数是回调,被调用来请求布局构建器 302 要处理的随后的位置。该回调给予 ContainerBox.BoxBuilder 在任意希望的方向上引导布局构建过程的选项。例如,它可以被使用于数次格式化相同的内容(例如,用于内容测量或滚动条分配或列平衡目的)。这个回调的结果应当是布局构建器 302 要处理的下一个块。要指出的是,该虚拟函数可以在不同的框构建器中不同地实施。这允许可扩展性,以及如果需要的话允许以不同方向和不同顺序导航通过内容。

[0054] CBB. GetChildBoxToReuse

[0055] 该回调被布局构建器 302 使用来询问某个已存在的框是否可以被再次使用于给定的块。已存在的框可在渐增更新情景中找到,比如当布局的大的部分保持不变而只有小的子树被修改和重新格式化时。

[0056] CBB.OnChildBoxReuse

[0057] 当找到可再次使用的框时调用这个回调，并且其应该被当前的 ContainerBox.BoxBuilder 插入到新的布局树结构中。

[0058] CBB.OnChildBoxEntry

[0059] 当针对当前的块没有找到可再次使用的框并且整个框要被再次重新构建时调用这个回调。为此目的，布局构建器 302 通知父框构建器有关子框输入。在这样的通知后，针对子块的新的框构建器将被创建、被推到栈上并成为当前的框构建器，以使得布局构建器 302 将在嵌套的层上继续它与框构建器的通信，直到到达所述块的终点。

[0060] CBB.OnChildBoxExit

[0061] 该回调被使用来通知父框构建器：它的子框之一的构建已结束，并且该子框可以被附连到父框上从而进入最终得到的布局树。父框构建器更新可用的布局空间来为分配下面的框（如果有的话）作准备。

[0062] CBB.Destructor

[0063] 该回调通知框构建器：布局构建器 302 已经到达它的终点，并即将把它从上下文栈中弹出。在此刻，框构建所需的所有过渡期数据将被释放。

[0064] CBB.InitializeBoxSizing

[0065] 这是一个回调，其被父框构建器从 OnChildBoxEntry 回调中调用，以便在进入到嵌套的子层前给出指派给其子框的空间。

[0066] CBB.CompleteBoxSizing

[0067] 这是回调，其被父框构建器从 OnChildBoxExit 回调中调用来最终确定子框的大小调整。

[0068] 示例性方法

[0069] 图 4 是描述按照一个或多个实施例的方法中的步骤的流程图。该方法可以结合任何适当的硬件、软件、固件或它们的组合来实施。在至少一些实施例中，该方法可以由适当地配置的顺序布局构建器——比如图 1 中的顺序布局构建器 111——来实施。

[0070] 步骤 400 接收结构化文档。可以接收任何适当类型的结构化文档。在至少一些实施例中，结构化文档包括 HTML 文档。另外，该步骤可以以任何适当的方式执行。例如，该步骤可以通过响应于用户动作来接收 HTML 文档而被执行，用户动作比如是使用浏览器请求网页。一旦接收到结构化文档，顺序布局处理就可以开始。于是，步骤 402 遇到结构化文档中的打开标签。步骤 404 创建相关联的框构建器。在上面提供了框构建器的例子。步骤 406 创建相关联的布局框。在图示的和描述的实施例中，各个框构建器类型熟知如何构建它们与之相关联的内容。该内容——在上面被称为持久性数据——被置于布局框中，并构成要在显示屏上再现的内容。

[0071] 步骤 408 使用框构建器来顺序地处理结构化文档，并用可再现的内容填充布局框。在其顺序处理期间，步骤 410 确定随后的标签是否是与打开标签相关联的关闭标签。如果不是，则该方法返回到步骤 408 继续顺序处理。另一方面，如果随后的标签是与打开标签相关联的关闭标签的话，步骤 412 使得布局框插入到显示树中。

[0072] 此时，如果结构化文档有更多的剩余，则该方法可以返回到步骤 402 且继续处理该文档。如果该文档没有更多剩余，则该方法可以终止，以及在对应的显示树中的内容可以

被再现。

[0073] 已经在按照一个或多个实施例的示例性方法中进行了描述,现在考虑按照一个或多个实施例的一些实施细节。

[0074] 实施细节

[0075] 在下面的章节中,描述了实现的不同方面。该材料为上面讨论的内容做了补充和添加了细节。首先,标题为“从布局结果数据中分离布局构建数据”的章节描述了处理效率可以如何通过数据分离技术来达到,所述数据分离技术至少部分地通过上面描述的体系结构而变为可能。接着,标题为“布局构建状态机”的章节描述了按照一个或多个实施例的状态机的诸方面。在这之后,标题为“布局框不可变性”的章节描述了不可变的布局框的诸方面。最后,标题为“内容计算”的章节描述了与规定的值、计算的值和已用值相关联的一些细节。

[0076] 从布局结果数据中分离布局构建数据

[0077] 在未由当前所描述的方案采用的递归算法方案中,组织的临时数据通常在执行栈上被作为递归地调用的函数的局部变量来分配。这对资源利用(比如存储器使用)产生不利影响。在图示的和描述的顺序方案中,数据栈被使用于存储这样的临时数据或“布局构建上下文”。然而,这个方案并不仅仅是广义的数据存储。而是,数据栈被组织为分层地嵌套的框构建器,正如上文所述的。回想上面的讨论,每一类型的布局元素(诸如框、表、图像、浮动物等等)使用它们自己的构建布局的逻辑和它们自己的临时数据组。类型特定的框构建器是实施和封装这样的特定数据和布局构建逻辑的对象。

[0078] 在图示的和描述的实施例中,每一类型的 HTML 元素如上所述地使用它自己的显示对象或布局框。布局框存储布局大小调整和定位结果,并被使用来适当地再现所述元素的内容。

[0079] 因此,按照本发明的方案,在布局构建器中使用数据结构二元性(duality)。具体地,框构建器充当在布局构建过程中被使用的活动的易变对象,而布局框充当稍后在再现和用户交互中将使用的、布局构建的最终的不可变结果。

[0080] 在图示的和描述的实施例中,按照各种各样类型的布局元素的 CSS 定义而定义了布局框的类层次。这在图 5 中进行了图示,并将被本领域的技术人员所理解。

[0081] 平行于布局框的层次,存在针对框构建器的类的平行的层次。在面向对象的实现中,框构建器类被定义为嵌套在适当的布局框中以便允许其在构建框的过程中访问它们的私有成员。框构建器层次在图 6 中显示,并将被技术人员所理解。

[0082] 布局构建状态机

[0083] 如以上所指出的,每个框构建器具有它自己的逻辑,它使用该逻辑来构建布局。然而,框构建器具有一些共同的特性。一个可以非常有用的共同特性是:构建过程可以被中断和恢复、被重置到起点和从任何的任意点重新开始。这种能力可以被有效地利用(leveraging)于诸如用于分页的布局片段、用于滚动条尺寸调整的布局迭代、用于列平衡的布局回溯等等这样的过程。

[0084] 为此,框构建器被组织为有限状态机,其依赖于它们当前正在处理的内容项目的属性从一个状态过渡到另一个状态。这样的状态机中的一些状态机相当简单(例如,未被设想为有任何要布置的内容的已替换框),然而另一些状态机可能非常地复杂(例如,处置

包含静态块、txt 行、内联块、浮动物、绝对定位块的“完全堆叠的上下文”，以及同时操控诸如滚动条定位、尊重(respect)类似“orphans(孤立行)”、“widows(未排足的行)”、“keep-together(保持在一起)”、“keep-with-next(与下一个一起)”等等的片段属性这样的布局的诸方面)。这一切可以使得布局构建的逻辑相当复杂，而状态机方案简化了它的组织，并允许诸如中断、恢复、重置、重新开始等等这样的过程。此外，状态机方案很好地适合于布局构建器的逻辑的顺序性质。

[0085] 布局框不可变性

[0086] 在定义用于代表布局结果的数据结构时，可能希望将定位数据与内容及它的大小调整进行分离。这制造了在随后的和并行的布局构建过程的不同位置中再次使用相同的布局框的可能性。

[0087] 为此原因，布局框被组织成仅包含它们的内部定位的内容和大小调整信息，而不包含它们的定位信息。另外，布局框被设计成不可变的，以使得在多个布局过程中再次使用框就如采用对它们的新的引用一样简单。

[0088] 内容计算

[0089] 为了使布局构建更加高效，可以在布局开始前计算不依赖于可用的布局空间的值，并将之作为计算的内容高速缓存(cache)而存储在源标记树中(作为纯粹地函数相关的数据)。这允许在随后的针对不同的可用空间的布局构建期间多次地再次使用这个重新计算的数据。

[0090] 在上面描述的布局构建器方案中，“计算的内容”的抽象概念被定义为仅依赖于内容属性而不依赖于任何空间特性(如视口(viewport)大小)的数据。有几种具体的计算的内容的子类型，例如计算的样式、针对表的模式修补结构(schema fix-up structure)、弹性框(flex box)、生成的内容、列表标记、首字母、以及插入(run-in)。为了与原始的内容和计算的内容一起工作，布局构建器被设计成使得其使用布局节点的抽象概念，所述抽象概念使原始内容和计算的内容统一。这样的组织允许按需要对计算的内容进行预先计算，并使得布局构建器逻辑与它处置的是哪种内容无关。这样的方案可以增强性能并缓解与重复计算相关联的问题。

[0091] 因此，上面描述的体系结构允许 HTML 文档的顺序布局处理。这个方案比上面提到的递归方案更加高效，因为它允许更细粒度的外部控制。作为例子，考虑下面内容。

[0092] 不同于不可以被中断、恢复或从不同的位置开始的递归，上面和下面描述的顺序布局处理可以被任意地停止和恢复。顺序布局处理还允许有用整体递归更加难以实现的高效的技术。例如，布局构建器 202(图 2)可以在 HTML 文档内的任意位置处开始它的工作。例如，布局构建器可以从大的 HTML 文档的中间开始处理，继续进行布局处理并在 HTML 文档的终点之前停止。这样做可以产生有用的、可再现的内容。

[0093] 与此稍微有关的是如下事实：在布局框 206 中出现的结果可以在针对 HTML 文档的顺序布局处理完成之前被使用。这在整体执行是处于单线程环境中的上下文中会是所希望的。此外，当被顺序地组织时，漫长的布局过程可以在任何步骤之间被暂停以便再现当前的显示结果，然后恢复以完成布局的剩余部分。这可以通过使得布局结果更早地可用于用户交互而改进 HTML 布局的可用性，由此产生更好的感知性能。

[0094] 另外，顺序布局处理可以在并行执行环境中被利用(leveraged)。具体地，当多个

CPU 可用时,并发的(concurrent)布局构建器可以被使用来处理 HTML 文档的不同部分并加入结果以产生完整的显示树。

[0095] 已经描述了各种顺序布局构建器实施例,现在考虑可以被使用来实施一个或多个上面描述的实施例的示例性系统。

[0096] 示例性系统

[0097] 图 8 图示了可以被使用来实施上面描述的各种实施例的示例性计算设备 700。计算设备 700 例如可以是图 1 的计算设备 102。

[0098] 计算设备 700 包括一个或多个处理器或处理单元 702、一个或多个存储器和 / 或存储组件 704、一个或多个输入 / 输出(I/O)设备 706、以及允许各种组件和设备彼此通信的总线 708。总线 708 代表若干类型的总线结构中的任意的一种或多种,包括存储器总线或存储器控制器、外围总线、加速图形端口、和使用各种各样的总线体系结构中的任意一种的处理器或本地总线。总线 708 可以包括有线的和 / 或无线的总线。

[0099] 存储器 / 存储组件 704 代表一个或多个计算机存储媒体。组件 704 可以包括易失性媒体(比如随机存取存储器(RAM))和 / 或非易失性媒体(比如只读存储器(ROM)、闪存、光盘、磁盘等等)。组件 704 可以包括固定的媒体(例如, RAM、ROM、固定的硬驱动机等等)以及可拆卸的媒体(例如, 闪存驱动机、可拆卸的硬驱动机、光盘等等)。

[0100] 一个或多个输入 / 输出设备 706 允许用户将命令和信息输入到计算设备 700,并且还允许信息被呈现给用户和 / 或其它的组件或设备。输入设备的例子包括键盘、光标控制设备(例如, 鼠标)、话筒、扫描仪等等。输出设备的例子包括显示设备(例如, 监视器或投影仪)、扬声器、打印机、网卡等等。

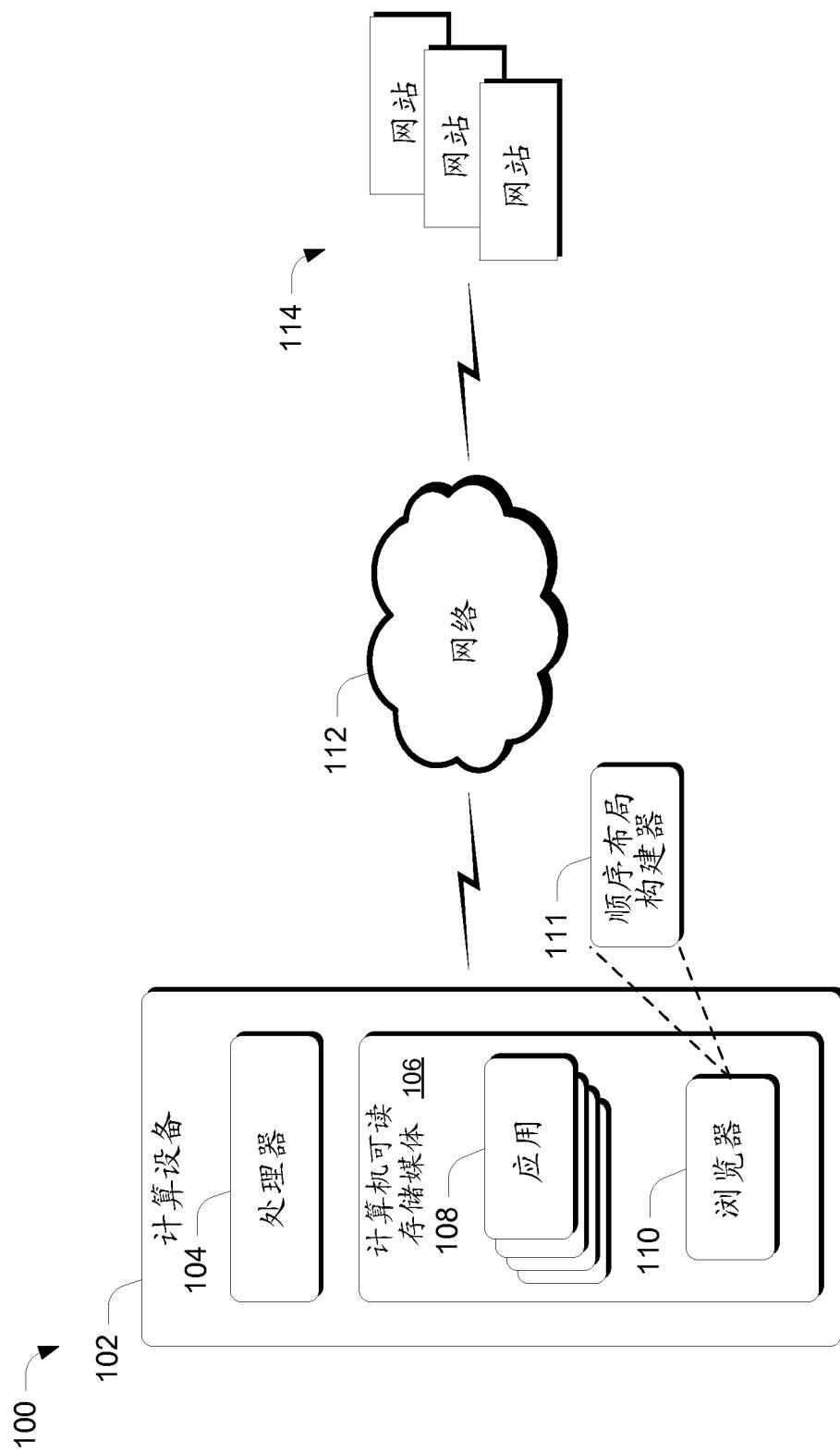
[0101] 这里,在软件或程序模块的一般上下文中描述了各种技术。通常,软件包括执行特定的任务或实施特定的抽象数据类型的例行程序、程序、对象、组件、数据结构等等。这些模块和技术的实现可以被存储在某种形式的计算机可读媒体上,或被跨越某种形式的计算机可读媒体进行传送。计算机可读媒体可以是由计算设备访问的、任何可用的介质或媒体。作为例子而不是限制,计算机可读媒体可包括“计算机可读存储媒体”。

[0102] “计算机可读存储媒体”包括以任何方法或技术实施的、用于存储诸如计算机可读指令、数据结构、程序模块或其它数据这样的信息的易失性和非易失性、可拆卸和不可拆卸的媒体。计算机可读存储媒体包括,但不限于, RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、数字多功能盘(DVD)或其它光学存储装置、盒式磁带、磁带、磁盘存储装置或其它的磁存储设备、或可被使用来存储想要的信息并可被计算机访问的任何其它的介质。

[0103] 总结

[0104] 已经描述了布局处理技术,所述布局处理技术允许实施结构化文档的处理,并允许以顺序的方式完成相关联的布局。布局过程被分成可以顺序地执行的多个更小的步骤。在至少一些实施例中,布局构建器被结合栈使用以控制在诸如 HTML 文档这样的结构化文档的结构层次的各个层次上的布局执行。不可变的数据结构被使用,并且其允许顺序布局操作的并行执行。在至少一些实施例中,可以完成布局序列的部分执行来产生可用的数据。

[0105] 尽管本主题是以特定于结构特征和 / 或方法动作的语言被描述的,但应当理解在所附权利要求中限定的主题不是必需局限于上述的特定特征或动作。而是,上述的特定特征和动作是作为实施权利要求的示例性形式而公开的。



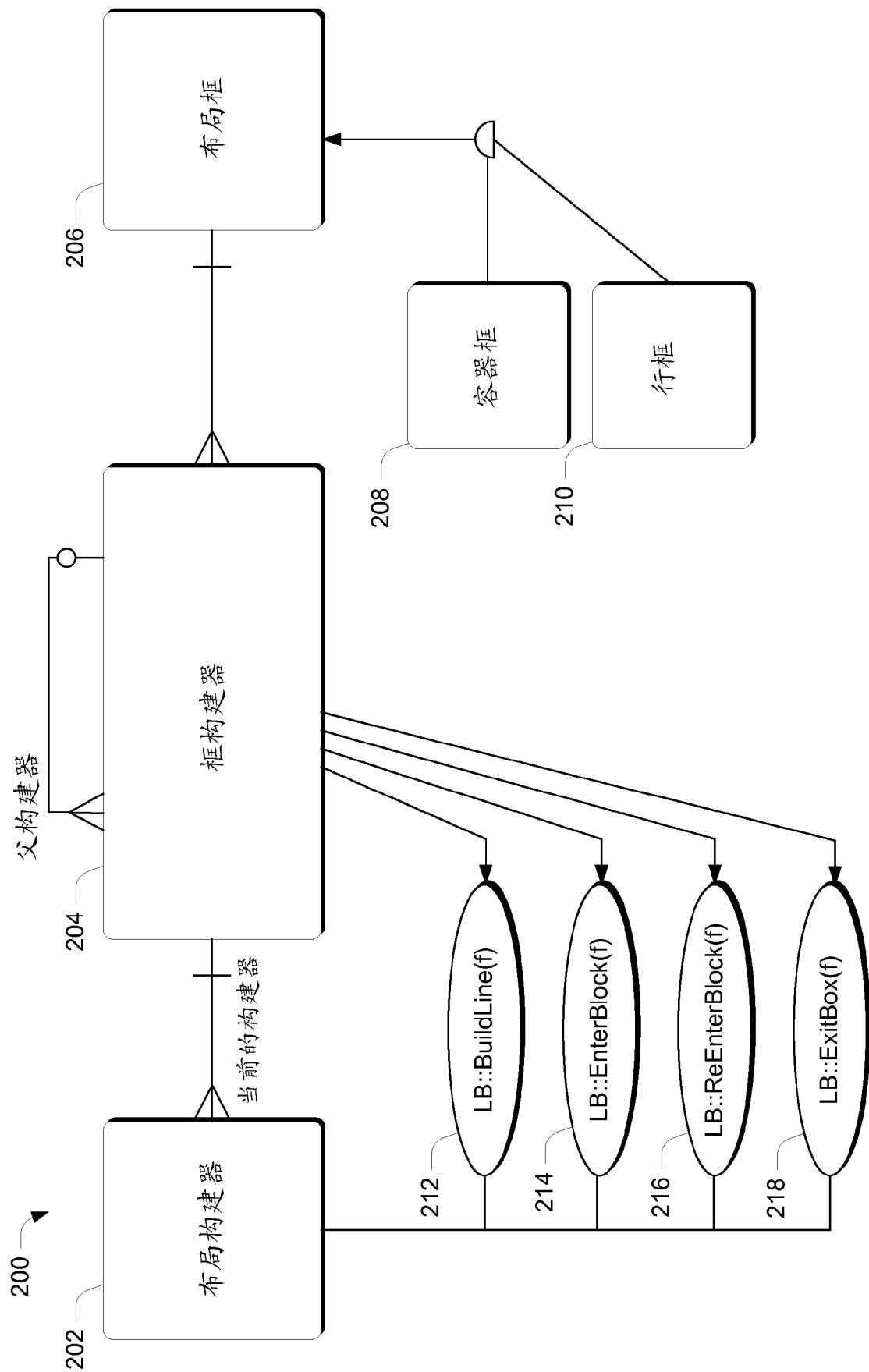


图 2

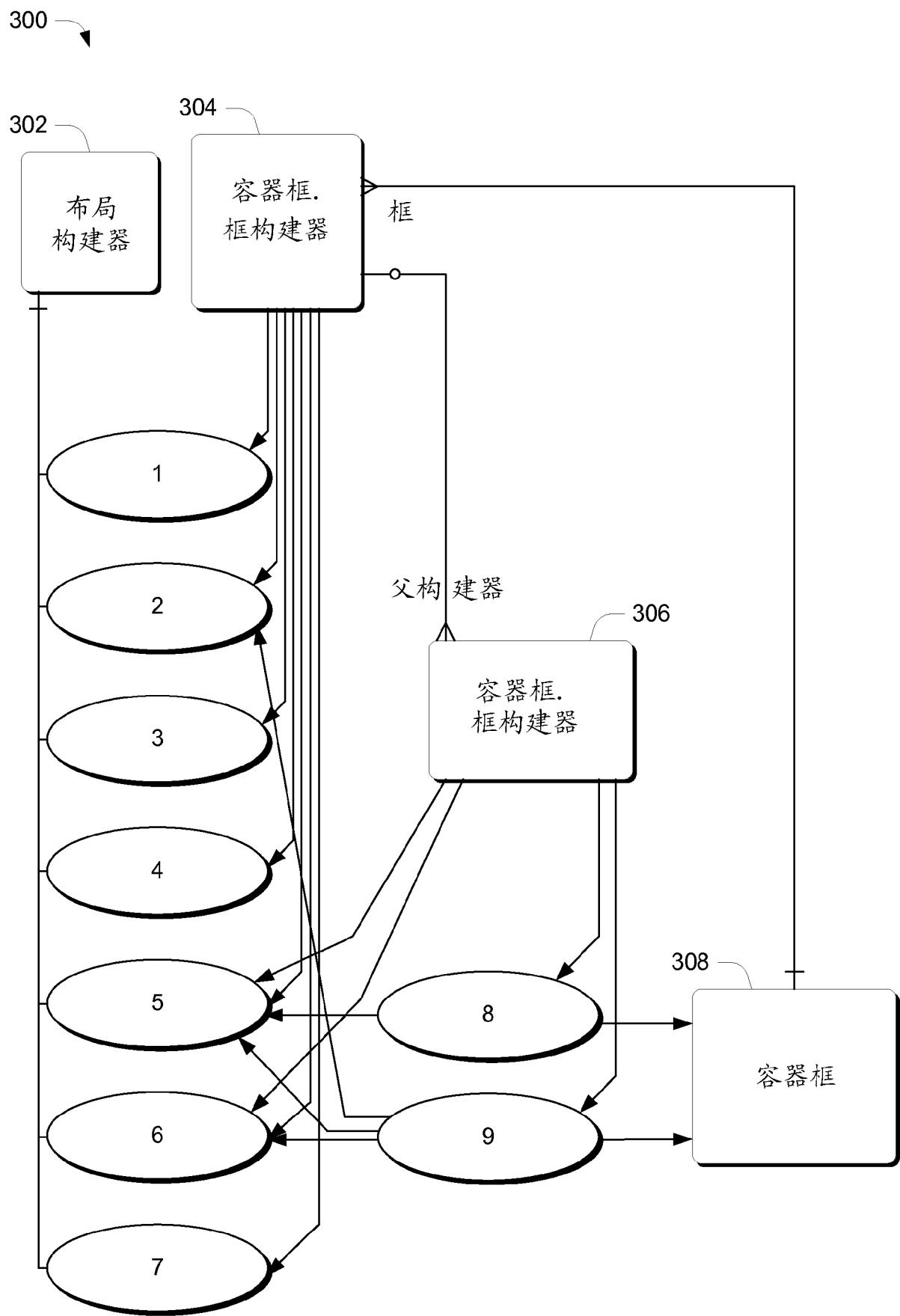


图 3

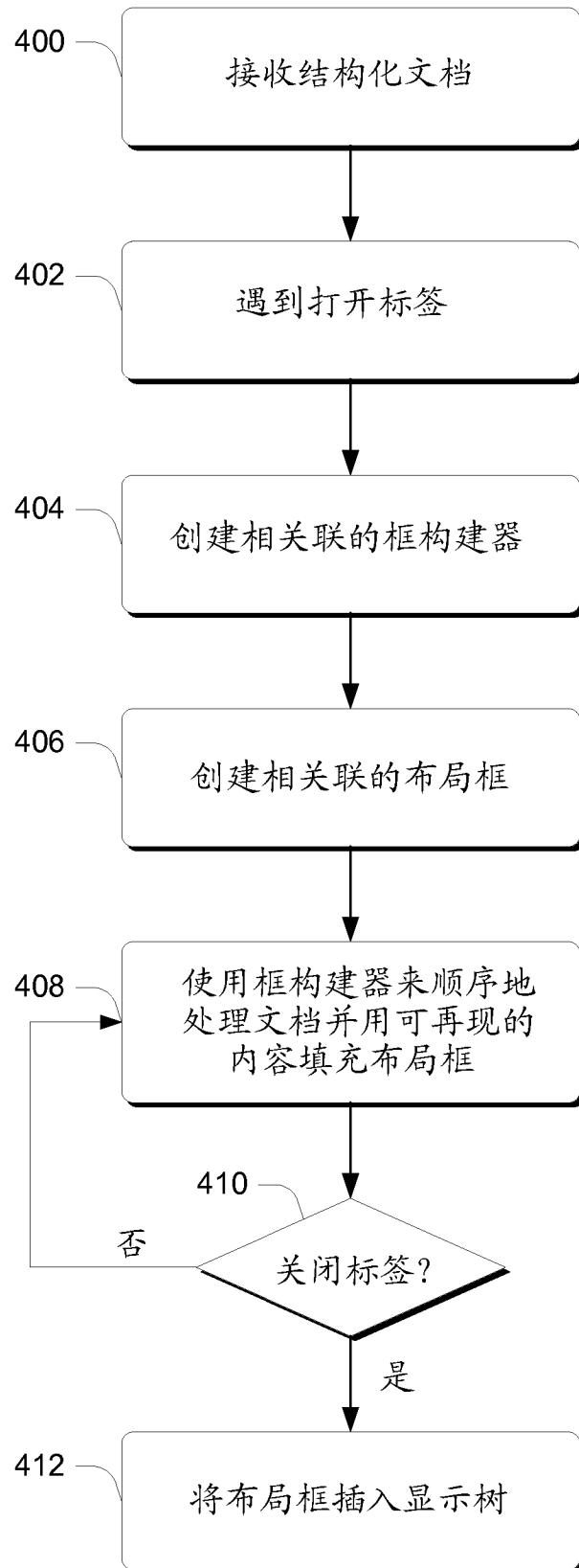
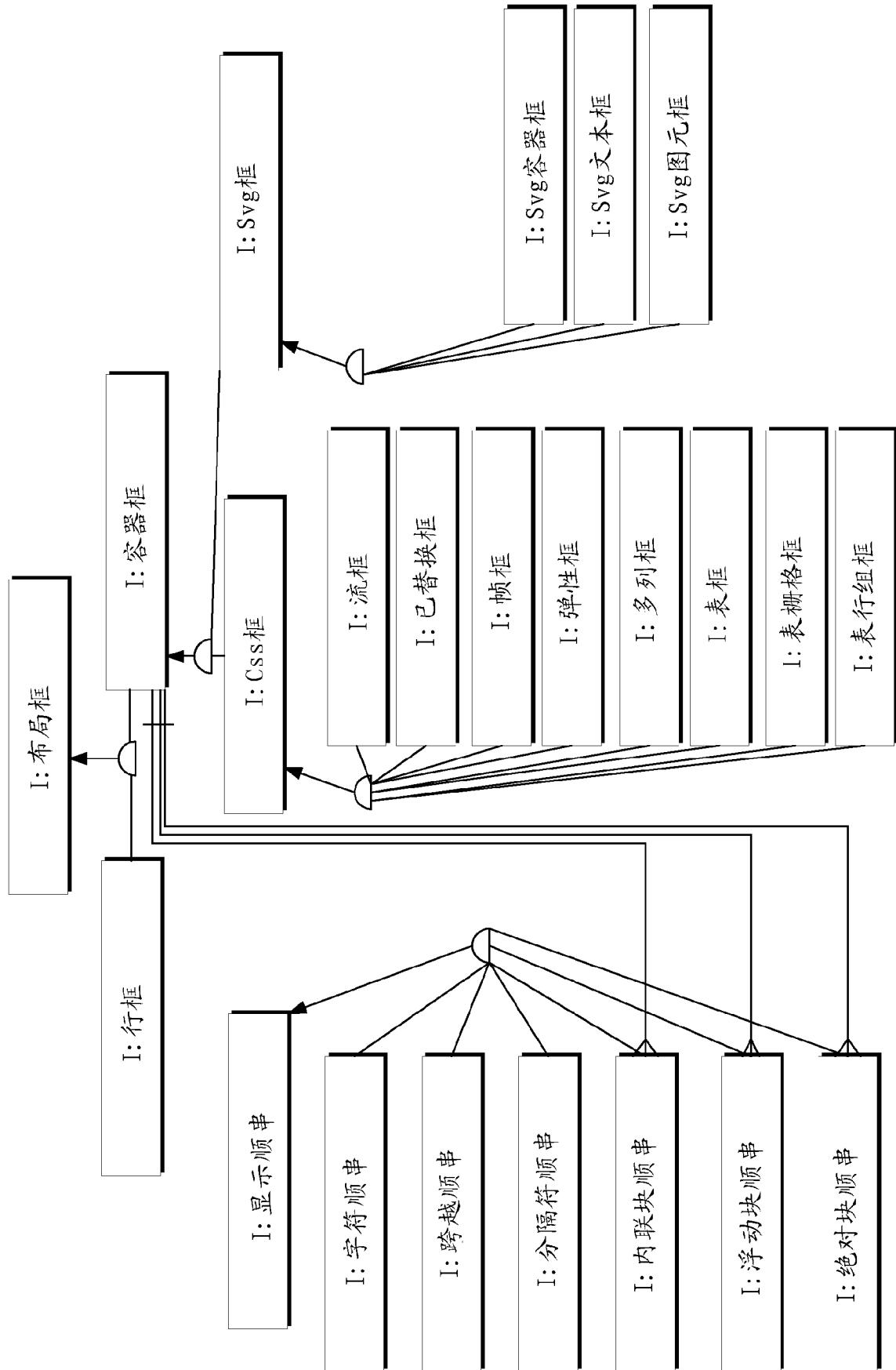


图 4



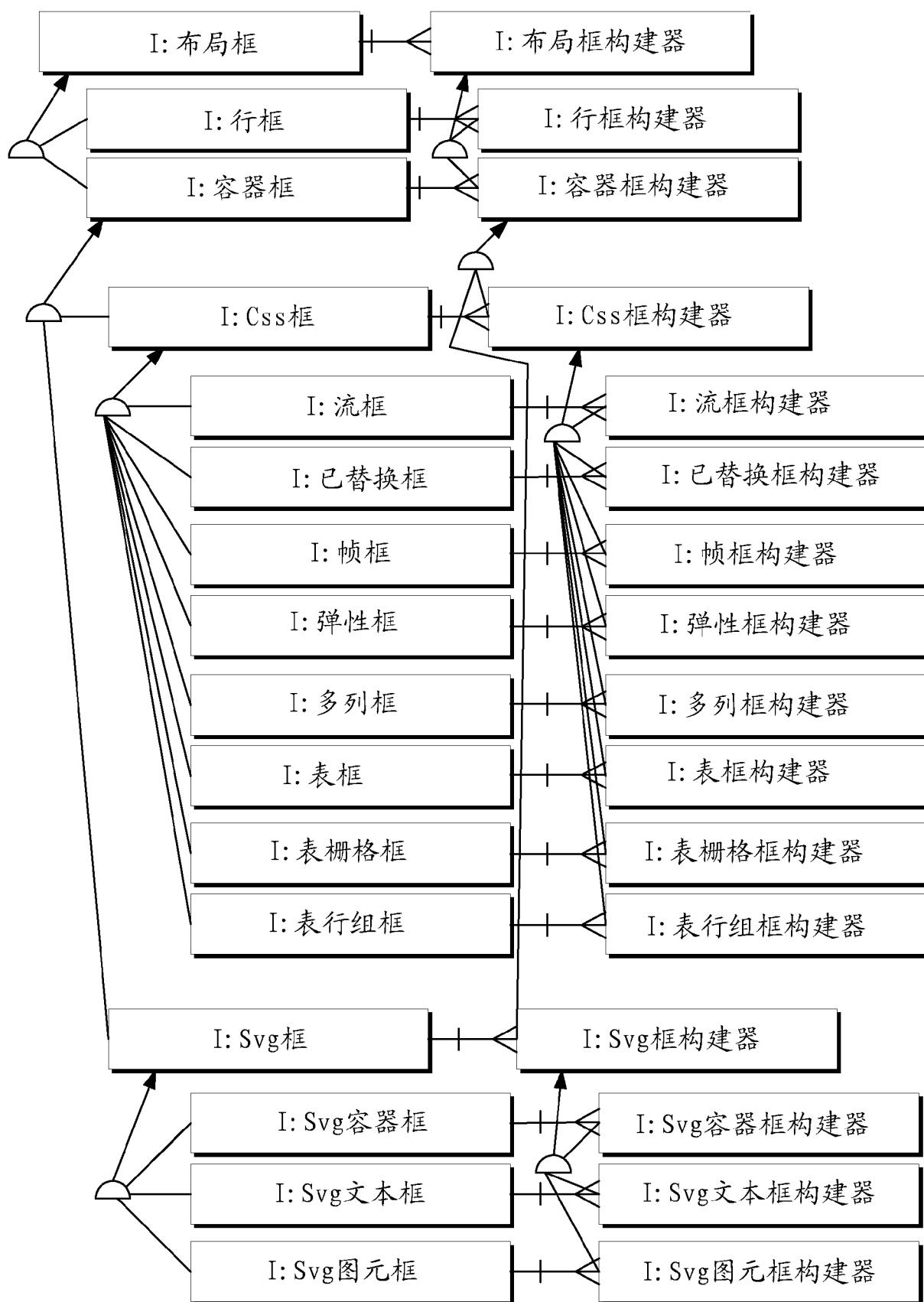


图 6

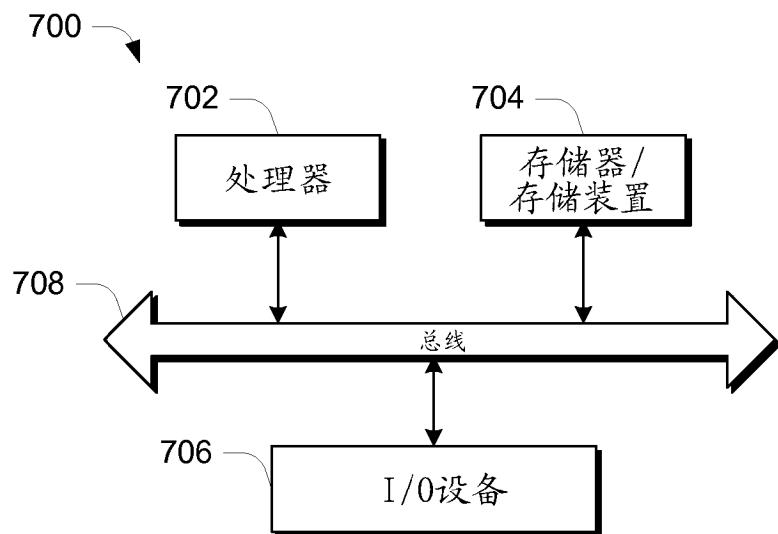


图 7