



## (12)发明专利

(10)授权公告号 CN 104704489 B

(45)授权公告日 2018.08.14

(21)申请号 201380043295.4

(73)专利权人 高通股份有限公司

(22)申请日 2013.06.17

地址 美国加利福尼亚

(65)同一申请的已公布的文献号

(72)发明人 M·韦伯 P·M·奥尔特戈  
M·S·福勒

申请公布号 CN 104704489 A

(74)专利代理机构 永新专利商标代理有限公司  
72002

(43)申请公布日 2015.06.10

代理人 张立达 王英

(30)优先权数据

61/684,002 2012.08.16 US

(51)Int.Cl.

606F 17/30(2006.01)

61/684,593 2012.08.17 US

13/722,048 2012.12.20 US

(56)对比文件

(85)PCT国际申请进入国家阶段日

US 2012/0110437 A1, 2012.05.03,  
US 6662341 B1, 2003.12.09,  
CN 102325188 A, 2012.01.18,  
US 2012/0110433 A1, 2012.05.03,

2015.02.13

审查员 胡武扬

(86)PCT国际申请的申请数据

权利要求书6页 说明书33页 附图17页

PCT/US2013/046110 2013.06.17

(87)PCT国际申请的公布数据

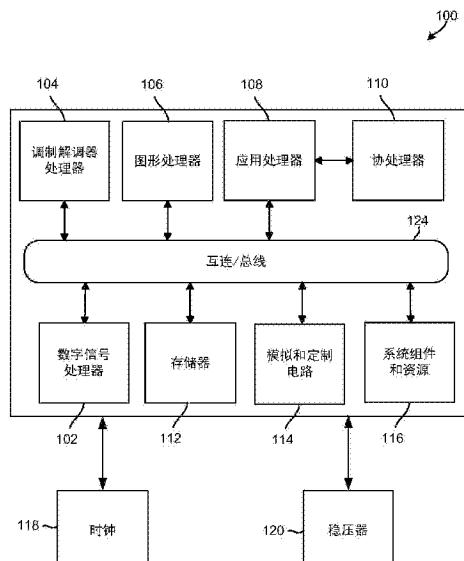
W02014/028116 EN 2014.02.20

## (54)发明名称

预测文档资源的使用

## (57)摘要

加载/呈现网页的浏览器系统和方法包括：使用推测/预测技术来预处理网络文档(HTML页面)以利用信息的不完整集合来识别可能需要的资源，以及请求/预取被确定为为了正确呈现网络文档而具有较高概率的需要的资源。推测/预测技术可以包括启发法的使用以改进文档加载和网络通信的效率和速度。



1.一种处理网页的方法,包括:

由第一过程扫描HTML文档以发现所述HTML文档中引用的外部资源,在计算设备的处理器中执行的所述第一过程与HTML解析器过程是并发的;

由所述第一过程调用对发现的外部资源的资源文档的下载,在所述第一过程继续扫描所述HTML文档的同时,所述下载被执行;

由第二过程扫描所下载的资源文档以发现另外的外部资源,在所述第一过程继续扫描所述HTML文档的同时,所述第二过程扫描所下载的资源文档;

在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性;

由所述第二过程接收关于由所述第一过程所识别的属性的信息;

由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载;以及使用所下载的资源在所述计算设备的电子显示器上呈现所述HTML文档。

2.根据权利要求1所述的方法,其中:

由第一过程扫描HTML文档包括:由HTML文档扫描器过程扫描所述HTML文档;以及

由第二过程扫描所下载的资源文档包括:由级联样式表文档扫描器过程扫描所下载的资源文档。

3.根据权利要求1所述的方法,其中,由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载包括:

推测为了在所述计算设备的所述电子显示器上呈现所述HTML文档而需要的相关外部资源。

4.根据权利要求1所述的方法,其中,扫描所下载的资源文档以发现另外的外部资源包括:

由所述第二过程扫描样式表文档以发现另外的外部资源。

5.根据权利要求4所述的方法,其中:

在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性包括:由所述第一过程识别与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性;以及

接收关于由所述第一过程所识别的属性的信息包括:由所述第二过程接收关于所识别的与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性的信息。

6.根据权利要求5所述的方法,其中,由所述第二过程基于所接收的信息来确定是否下载发现的另外的资源包括:

确定每个与样式规则相关联的识别的HTML id、类别和样式属性是否已经被所述第一过程遇到;

响应于确定每个与所述样式规则相关联的识别的HTML id、类别和样式属性都已经被第一过程遇到,立即请求所述样式规则引用的资源;以及

响应于确定不是每个与所述样式规则相关联的识别的HTML id、类别和样式属性都被第一过程遇到,将所述样式规则存储在存储器中。

7.根据权利要求6所述的方法,还包括:

当对所述HTML文档的扫描完成时,由所述第一过程生成通知;以及

由所述第二过程接收所述通知。

8. 根据权利要求7所述的方法,还包括:

响应于所述第二过程接收所述通知,由所述第二过程从存储器取回所存储的样式规则;

确定每个HTML id、类别和样式属性是否都已经被所述第一过程遇到;以及

响应于确定每个HTML id、类别和样式属性都已经被所述第一过程遇到,请求所取回的样式规则引用的资源。

9. 一种计算设备,包括:

用于由第一过程扫描HTML文档以发现所述HTML文档中引用的外部资源的单元,执行的所述第一过程与HTML解析器过程是并发的;

用于由所述第一过程调用对发现的外部资源的资源文档的下载的单元,在所述第一过程继续扫描所述HTML文档的同时,所述下载被执行;

用于由第二过程扫描所下载的资源文档以发现另外的外部资源的单元,在所述第一过程继续扫描所述HTML文档的同时,所述第二过程扫描所下载的资源文档;

用于在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性的单元;

用于由所述第二过程接收与由所述第一过程识别的属性相关的信息的单元;

用于由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载的单元;以及

用于使用所下载的资源在所述计算设备的电子显示器上呈现所述HTML文档的单元。

10. 根据权利要求9所述的计算设备,其中:

用于由第一过程扫描HTML文档的单元包括:用于由HTML文档扫描器过程扫描所述HTML文档的单元;以及

用于由第二过程扫描所下载的资源文档的单元包括:用于由级联样式表文档扫描器过程扫描所下载的资源文档的单元。

11. 根据权利要求9所述的计算设备,其中,用于由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载的单元包括:

用于推测为了在所述电子显示器上呈现所述HTML文档而需要的相关外部资源的单元。

12. 根据权利要求9所述的计算设备,其中,用于扫描所下载的资源文档以发现另外的外部资源的单元包括:

用于由所述第二过程扫描样式表文档以发现另外的外部资源的单元。

13. 根据权利要求12所述的计算设备,其中:

用于在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性的单元包括:用于由所述第一过程识别与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性的单元;以及

用于接收关于由所述第一过程所识别的属性的信息的单元包括:用于由所述第二过程接收关于所识别的与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性的信息的单元。

14. 根据权利要求13所述的计算设备,其中,用于由所述第二过程基于所接收的信息来

确定是否下载发现的另外的资源的单元包括：

用于确定每个与样式规则相关联的识别的HTML id、类别和样式属性是否已经被第一过程遇到的单元；

用于响应于确定每个与所述样式规则相关联的识别的HTML id、类别和样式属性都已经被第一过程遇到，立即请求所述样式规则引用的资源的单元；以及

用于响应于确定不是每个与所述样式规则相关联的识别的HTML id、类别和样式属性都被第一过程遇到，将所述样式规则存储在存储器中的单元。

15. 根据权利要求14所述的计算设备，还包括：

用于当对所述HTML文档的扫描完成时，由所述第一过程生成通知的单元；以及

用于由所述第二过程接收所述通知的单元。

16. 根据权利要求15所述的计算设备，还包括：

用于响应于所述第二过程接收所述通知，由所述第二过程从存储器取回所存储的样式规则的单元；

用于确定每个HTML id、类别和样式属性是否都已经被所述第一过程遇到的单元；以及

用于响应于确定每个HTML id、类别和样式属性都已经被所述第一过程遇到，请求所取回的样式规则引用的资源的单元。

17. 一种计算设备，包括：

处理器，所述处理器配置有用于执行操作的处理器可执行指令，所述操作包括：

由第一过程扫描HTML文档以发现所述HTML文档中引用的外部资源，所述第一过程与HTML解析器过程是并发执行的；

由所述第一过程调用对发现的外部资源的资源文档的下载，在所述第一过程继续扫描所述HTML文档的同时，所述下载被执行；

由第二过程扫描所下载的资源文档以发现另外的外部资源，在所述第一过程继续扫描所述HTML文档的同时，所述第二过程扫描所下载的资源文档；

在所述第二过程继续扫描所下载的资源文档的同时，由所述第一过程识别所述HTML文档的属性；

由所述第二过程接收关于由所述第一过程所识别的属性的信息；

由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载；以及使用所下载的资源在所述计算设备的电子显示器上呈现所述HTML文档。

18. 根据权利要求17所述的计算设备，其中，所述处理器配置有处理器可执行指令，从而：

由第一过程扫描HTML文档包括：由HTML文档扫描器过程扫描所述HTML文档；以及

由第二过程扫描所下载的资源文档包括：由级联样式表文档扫描器过程扫描所下载的资源文档。

19. 根据权利要求17所述的计算设备，其中，所述处理器配置有处理器可执行指令，从而：

由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载包括：推測为了在所述计算设备的所述电子显示器上呈现所述HTML文档而需要的相关外部资源。

20. 根据权利要求17所述的计算设备，其中，所述处理器配置有处理器可执行指令，从

而：

扫描所下载的资源文档以发现另外的外部资源包括：由所述第二过程扫描样式表文档以发现另外的外部资源。

21. 根据权利要求20所述的计算设备，其中，所述处理器配置有处理器可执行指令，从而：

在所述第二过程继续扫描所下载的资源文档的同时，由所述第一过程识别所述HTML文档的属性包括：由所述第一过程识别与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性；以及

接收关于由所述第一过程所识别的属性的信息包括：由所述第二过程接收关于所识别的与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性的信息。

22. 根据权利要求21所述的计算设备，其中，所述处理器配置有处理器可执行指令，从而：

由所述第二过程基于所接收的信息来确定是否下载发现的另外的资源包括：

确定每个与样式规则相关联的识别的HTML id、类别和样式属性是否已经被所述第一过程遇到；

响应于确定每个与所述样式规则相关联的识别的HTML id、类别和样式属性都已经被第一过程遇到，立即请求所述样式规则引用的资源；以及

响应于确定不是每个与所述样式规则相关联的识别的HTML id、类别和样式属性都被第一过程遇到，将所述样式规则存储在存储器中。

23. 根据权利要求22所述的计算设备，其中，所述处理器配置有用于执行操作的处理器可执行指令，所述操作还包括：

当对所述HTML文档的扫描完成时，由所述第一过程生成通知；以及

由所述第二过程接收所述通知。

24. 根据权利要求23所述的计算设备，其中，所述处理器配置有用于执行操作的处理器可执行指令，所述操作还包括：

响应于所述第二过程接收所述通知，由所述第二过程从存储器取回所存储的样式规则；

确定每个HTML id、类别和样式属性是否都已经被所述第一过程遇到；以及

响应于确定每个HTML id、类别和样式属性都已经被所述第一过程遇到，请求所取回的样式规则引用的资源。

25. 一种非暂时性计算机可读存储介质，所述非暂时性计算机可读存储介质具有在其上存储的处理器可执行的软件指令，所述处理器可执行的软件指令被配置为使得处理器执行用于处理网页的操作，所述操作包括：

由第一过程扫描HTML文档以发现所述HTML文档中引用的外部资源，所述第一过程与HTML解析器过程是并发执行的；

由所述第一过程调用对发现的外部资源的资源文档的下载，在所述第一过程继续扫描所述HTML文档的同时，所述下载被执行；

由第二过程扫描所下载的资源文档以发现另外的外部资源，在所述第一过程继续扫描所述HTML文档的同时，所述第二过程扫描所下载的资源文档；

在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性;

由所述第二过程接收关于由所述第一过程所识别的属性的信息;

由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载;以及使用所下载的资源在计算设备的电子显示器上呈现所述HTML文档。

26. 根据权利要求25所述的非暂时性计算机可读存储介质,其中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而:

由第一过程扫描HTML文档包括:由HTML文档扫描器过程扫描所述HTML文档;以及

由第二过程扫描所下载的资源文档包括:由级联样式表文档扫描器过程扫描所下载的资源文档。

27. 根据权利要求25所述的非暂时性计算机可读存储介质,其中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而:

由所述第二过程基于所接收的信息来确定是否发起发现的另外的资源的下载包括:推为了在所述计算设备的所述电子显示器上呈现所述HTML文档而需要的相关外部资源。

28. 根据权利要求25所述的非暂时性计算机可读存储介质,其中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而:

扫描所下载的资源文档以发现另外的外部资源包括:由所述第二过程扫描样式表文档以发现另外的外部资源。

29. 根据权利要求28所述的非暂时性计算机可读存储介质,其中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而:

在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性包括:由所述第一过程识别与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性;以及

接收关于由所述第一过程所识别的属性的信息包括:由所述第二过程接收关于所识别的与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性的信息。

30. 根据权利要求29所述的非暂时性计算机可读存储介质,其中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而:

由所述第二过程基于所接收的信息来确定是否下载发现的另外的资源包括:

确定每个与样式规则相关联的识别的HTML id、类别和样式属性是否已经被所述第一过程遇到;

响应于确定每个与所述样式规则相关联的识别的HTML id、类别和样式属性都已经被第一过程遇到,立即请求所述样式规则引用的资源;以及,

响应于确定不是每个与所述样式规则相关联的识别的HTML id、类别和样式属性都被第一过程遇到,将所述样式规则存储在存储器中。

31. 根据权利要求30所述的非暂时性计算机可读存储介质,其中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,所述操作包括:

当对所述HTML文档的扫描完成时,由所述第一过程生成通知;以及

由所述第二过程接收所述通知。

32. 根据权利要求31所述的非暂时性计算机可读存储介质,其中,所述存储的处理器可

执行的软件指令被配置为使得处理器执行操作,所述操作包括:

响应于所述第二过程接收所述通知,由所述第二过程从存储器取回所存储的样式规则;

确定每个HTML id、类别和样式属性是否都已经被所述第一过程遇到;以及

响应于确定每个HTML id、类别和样式属性都已经被所述第一过程遇到,请求所取回的样式规则引用的资源。

## 预测文档资源的使用

[0001] 相关申请的交叉引用

[0002] 本申请要求享有于2012年8月17日递交的、题目为“Predicting the Usage of Document Resources”的美国临时专利申请No.61/684,593以及于2012年8月16日递交的、题目为“Predicting the Usage of Document Resources”的美国临时专利申请No.61/684,002的优先权，故这两篇申请的全部内容以引用方式被并入本文。

### 技术领域

[0003] 本发明涉及用于在网络浏览器中呈现HTML文档的方法、系统和设备，并且更具体地，涉及用于使网络浏览器操作并行化的方法。

### 背景技术

[0004] 无线通信技术和移动电子设备(例如，蜂窝电话、平板计算机、膝上型计算机等)在过去几年变得普遍使用。为了跟上增加的消费者需求，移动电子设备的功能变得更加丰富，并且现在通常包括多处理器、片上系统(SoC)和使得移动设备用户能够在其移动设备上执行复杂且耗电量大的软件应用(例如，网络浏览器、视频流应用等)的其它元件。由于这些和其它改进，智能电话和平板计算机变得普遍，并且取代膝上型计算机和台式机成为许多用户选择的平台。

[0005] 移动设备用户现在可以通过经由其移动设备上的浏览器应用访问互联网来容易且方便地实现许多其日常任务。随着移动设备继续变得普及，消费者期待能够更好地使用现代移动设备的多处理能力的网络浏览器。

### 发明内容

[0006] 各个方面包括处理网页的方法，其可以包括：由第一过程扫描HTML文档以发现所述HTML文档中引用的外部资源，在计算设备的处理器中执行的所述第一过程与HTML解析器过程是并发的；由所述第一过程调用对发现的外部资源的资源文档的下载，在所述第一过程继续扫描所述HTML文档的同时，所述下载被执行；由第二过程扫描所下载的资源文档以发现另外的外部资源，在所述第一过程继续扫描所述HTML文档的同时，所述第二过程扫描所下载的资源文档；在所述第二过程继续扫描所下载的资源文档的同时，由所述第一过程识别所述HTML文档的属性；由所述第二过程接收关于由所述第一过程所识别的属性的信息；由所述第二过程基于所接收的信息来确定是否发起发现的另外的资源的下载；以及使用所下载的资源在所述计算设备的电子显示器上呈现所述HTML文档。

[0007] 在一个方面中，由第一过程扫描HTML文档可以包括：由HTML文档扫描器过程扫描所述HTML文档。在进一步的方面中，由第二过程扫描所下载的资源文档可以包括：由级联样式表文档扫描器过程扫描所下载的资源文档。在进一步的方面中，由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载可以包括：推测为了在所述计算设备的所述电子显示器上呈现所述HTML文档而需要的相关外部资源。

[0008] 在进一步的方面中，扫描所下载的资源文档以发现另外的外部资源可以包括：由所述第二过程扫描样式表文档以发现另外的外部资源。在进一步的方面中，在所述第二过程继续扫描所下载的资源文档的同时，由所述第一过程识别所述HTML文档的属性可以包括：由所述第一过程识别与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性；以及接收关于由所述第一过程所识别的属性的信息可以包括：由所述第二过程接收关于与所述HTML文档包括的HTML元素相关联的所识别的HTML id、类别和样式属性的信息。

[0009] 在进一步的方面中，由所述第二过程基于所接收的信息来确定是否下载发现的另外的资源可以包括：确定与样式规则相关联的每个识别的HTML id、类别和样式属性是否已经被所述第一过程遇到；响应于确定与所述样式规则相关联的每个识别的HTML id、类别和样式属性都已经被第一过程遇到，立即请求所述样式规则引用的资源；以及，响应于确定与所述样式规则相关联的识别的HTML id、类别和样式属性未都被第一过程遇到，将所述样式规则存储在存储器中。

[0010] 在进一步的方面中，所述方法可以包括：当对所述HTML文档的扫描完成时，由所述第一过程生成通知；以及由所述第二过程接收所述通知。在进一步的方面中，所述方法可以包括：响应于所述第二过程接收所述通知，由所述第二过程从存储器取回所存储的样式规则；确定每个HTML id、类别和样式属性是否都已经被所述第一过程遇到；以及响应于确定每个HTML id、类别和样式属性都已经被所述第一过程遇到，请求所取回的样式规则引用的资源。

[0011] 进一步的方面包括一种计算设备，其包括：用于由第一过程扫描HTML文档以发现所述HTML文档中引用的外部资源的单元，执行的所述第一过程与HTML解析器过程是并发的；用于由所述第一过程调用对发现的外部资源的资源文档的下载的单元，在所述第一过程继续扫描所述HTML文档的同时，所述下载被执行；用于由第二过程扫描所下载的资源文档以发现另外的外部资源的单元，在所述第一过程继续扫描所述HTML文档的同时，所述第二过程扫描所下载的资源文档；用于在所述第二过程继续扫描所下载的资源文档的同时，由所述第一过程识别所述HTML文档的属性的单元；用于由所述第二过程接收与由所述第一过程识别的属性相关的信息的单元；用于由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载的单元；以及用于使用所下载的资源在所述计算设备的电子显示器上呈现所述HTML文档的单元。

[0012] 在一个方面中，用于由第一过程扫描HTML文档的单元可以包括：用于由HTML文档扫描器过程扫描所述HTML文档的单元；以及用于由第二过程扫描所下载的资源文档的单元可以包括：用于由级联样式表文档扫描器过程扫描所下载的资源文档的单元。在进一步的方面中，根据权利要求9所述的计算设备，其中，用于由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载的单元可以包括：用于推测为了在所述电子显示器上呈现所述HTML文档而需要的相关外部资源的单元。

[0013] 在进一步的方面中，用于扫描所下载的资源文档以发现另外的外部资源的单元可以包括：用于由所述第二过程扫描样式表文档以发现另外的外部资源的单元。在进一步的方面中，用于在所述第二过程继续扫描所下载的资源文档的同时，由所述第一过程识别所述HTML文档的属性的单元可以包括：用于由所述第一过程识别与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性的单元；以及用于接收关于由所述第一过程所识别

的属性的信息的单元包括:用于由所述第二过程接收关于与所述HTML文档可以包括的HTML元素相关联的所识别的HTML id、类别和样式属性的信息的单元。

[0014] 在进一步的方面中,用于由所述第二过程基于所接收的信息来确定是否下载发现的另外的资源的单元可以包括:用于确定与样式规则相关联的每个识别的HTML id、类别和样式属性是否已经被第一过程遇到的单元;用于响应于确定与所述样式规则相关联的每个识别的HTML id、类别和样式属性都已经被第一过程遇到,立即请求所述样式规则引用的资源的单元;以及,用于响应于确定与所述样式规则相关联的识别的HTML id、类别和样式属性未都被第一过程遇到,将所述样式规则存储在存储器中的单元。

[0015] 在进一步的方面中,所述计算设备可以包括:用于当对所述HTML文档的扫描完成时,由所述第一过程生成通知的单元;以及用于由所述第二过程接收所述通知的单元。在进一步的方面中,所述计算设备可以包括:用于响应于所述第二过程接收所述通知,由所述第二过程从存储器取回所存储的样式规则的单元;用于确定每个HTML id、类别和样式属性是否都已经被所述第一过程遇到的单元;以及用于响应于确定每个HTML id、类别和样式属性都已经被所述第一过程遇到,请求所取回的样式规则引用的资源的单元。

[0016] 进一步的方面包括一种具有处理器的计算机设备,所述处理器配置有用于执行操作的处理器可执行指令,所述操作包括:由第一过程扫描HTML文档以发现所述HTML文档中引用的外部资源,在计算设备的处理器中执行的所述第一过程与HTML解析器过程是并发的;由所述第一过程调用发现的外部资源的资源文档的下载,在所述第一过程继续扫描所述HTML文档的同时,所述下载被执行;由第二过程扫描所下载的资源文档以发现另外的外部资源,在所述第一过程继续扫描所述HTML文档的同时,所述第二过程扫描所下载的资源文档;在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性;由所述第二过程接收关于由所述第一过程所识别的属性的信息;由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载;以及使用所下载的资源在所述计算设备的电子显示器上呈现所述HTML文档。

[0017] 在一个方面中,所述处理器配置有处理器可执行指令,从而由第一过程扫描HTML文档可以包括:由HTML文档扫描器过程扫描所述HTML文档;以及由第二过程扫描所下载的资源文档可以包括:由级联样式表文档扫描器过程扫描所下载的资源文档。

[0018] 在进一步的方面中,所述处理器配置有处理器可执行指令,从而由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载可以包括:推测为了在所述计算设备的所述电子显示器上呈现所述HTML文档而需要的相关外部资源。在进一步的方面中,所述处理器配置有处理器可执行指令,从而扫描所下载的资源文档以发现另外的外部资源可以包括:由所述第二过程扫描样式表文档以发现另外的外部资源。

[0019] 在进一步的方面中,所述处理器配置有处理器可执行指令,从而在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性可以包括:由所述第一过程识别与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性;以及接收关于由所述第一过程所识别的属性的信息可以包括:由所述第二过程接收关于与所述HTML文档包括的HTML元素相关联的所识别的HTML id、类别和样式属性的信息。

[0020] 在进一步的方面中,所述处理器配置有处理器可执行指令,从而由所述第二过程基于所接收的信息来确定是否下载发现的另外的资源可以包括:确定与样式规则相关联的

每个识别的HTML id、类别和样式属性是否已经被所述第一过程遇到;响应于确定与所述样式规则相关联的每个识别的HTML id、类别和样式属性都已经被第一过程遇到,立即请求所述样式规则引用的资源;以及,响应于确定与所述样式规则相关联的识别的HTML id、类别和样式属性未都被第一过程遇到,将所述样式规则存储在存储器中。在进一步的方面中,所述处理器配置有用于执行操作的处理器可执行指令,所述操作还包括:当对所述HTML文档的扫描完成时,由所述第一过程生成通知;以及由所述第二过程接收所述通知。

[0021] 在进一步的方面中,所述处理器配置有用于执行操作的处理器可执行指令,所述操作还包括:响应于所述第二过程接收所述通知,由所述第二过程从存储器取回所存储的样式规则;确定每个HTML id、类别和样式属性是否都已经被所述第一过程遇到;以及响应于确定每个HTML id、类别和样式属性都已经被所述第一过程遇到,请求所取回的样式规则引用的资源。

[0022] 进一步的方面包括一种非暂时性计算机可读存储介质,所述非暂时性计算机可读存储介质具有在其上存储的处理器可执行的软件指令,所述处理器可执行的软件指令被配置为使得处理器执行用于处理网页的操作,所述操作包括:由第一过程扫描HTML文档以发现所述HTML文档中引用的外部资源,在计算设备的处理器中执行的所述第一过程与HTML解析器过程是并发的;由所述第一过程调用发现的外部资源的资源文档的下载,在所述第一过程继续扫描所述HTML文档的同时,所述下载被执行;由第二过程扫描所下载的资源文档以发现另外的外部资源,在所述第一过程继续扫描所述HTML文档的同时,所述第二过程扫描所下载的资源文档;在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性;由所述第二过程接收关于由所述第一过程所识别的属性的信息;由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载;以及使用所下载的资源在所述计算设备的电子显示器上呈现所述HTML文档。

[0023] 在一个方面中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而由第一过程扫描HTML文档可以包括:由HTML文档扫描器过程扫描所述HTML文档;以及由第二过程扫描所下载的资源文档可以包括:由级联样式表文档扫描器过程扫描所下载的资源文档。

[0024] 在进一步的方面中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而由所述第二过程基于所接收的信息来确定是否发起对发现的另外的资源的下载可以包括:推测为了在所述计算设备的所述电子显示器上呈现所述HTML文档而需要的相关外部资源。

[0025] 在进一步的方面中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而扫描所下载的资源文档以发现另外的外部资源可以包括:由所述第二过程扫描样式表文档以发现另外的外部资源。

[0026] 在进一步的方面中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而在所述第二过程继续扫描所下载的资源文档的同时,由所述第一过程识别所述HTML文档的属性可以包括:由所述第一过程识别与所述HTML文档包括的HTML元素相关联的HTML id、类别和样式属性;以及接收关于由所述第一过程所识别的属性的信息可以包括:由所述第二过程接收关于与所述HTML文档包括的HTML元素相关联的所识别的HTML id、类别和样式属性的信息。

[0027] 在进一步的方面中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而由所述第二过程基于所接收的信息来确定是否下载发现的另外的资源可以包括:确定与样式规则相关联的每个识别的HTML id、类别和样式属性是否已经被所述第一过程遇到;响应于确定与所述样式规则相关联的每个识别的HTML id、类别和样式属性都已经被第一过程遇到,立即请求所述样式规则引用的资源;以及,响应于确定与所述样式规则相关联的识别的HTML id、类别和样式属性未都被第一过程遇到,将所述样式规则存储在存储器中。

[0028] 在进一步的方面中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而当对所述HTML文档的扫描完成时,由所述第一过程生成通知;以及由所述第二过程接收所述通知。在进一步的方面中,所述存储的处理器可执行的软件指令被配置为使得处理器执行操作,从而响应于所述第二过程接收所述通知,由所述第二过程从存储器取回所存储的样式规则;确定每个HTML id、类别和样式属性是否都已经被所述第一过程遇到;以及响应于确定每个HTML id、类别和样式属性都已经被所述第一过程遇到,请求所取回的样式规则引用的资源。

## 附图说明

[0029] 被并入本文且构成本说明书的一部分的附图示出了本发明的示例性方面。结合上文给出的概括性描述和下文给出的详细描述,这些附图用于说明本发明的特征而不是对公开的方面进行限制。

[0030] 图1是示出了可以用在实现各个方面的计算设备中的示例性片上系统(SoC)架构的组件框图。

[0031] 图2是示出了可以用于实现各个方面的示例性多核处理器架构的功能框图。

[0032] 图3A是示出了用于呈现HTML文档的示例性浏览器方法的过程流程图。

[0033] 图3B是示出了在示例性浏览器系统中的示例性逻辑组件、信息流、操作和转换的功能和过程流程图。

[0034] 图4是示出了示例性浏览器系统中的示例性逻辑组件、功能组件、信息流和子系统的功能框图。

[0035] 图5是根据示例性方法示出的实现并行的浏览器基础结构的示例性浏览器系统的功能框图。

[0036] 图6是示出了处理HTML文档以在页面加载/呈现操作之前发现并且预取资源的示例性浏览器方法的过程流程图。

[0037] 图7A是示出了使用推测技术和启发法来预测文档资源的使用的示例性浏览器方法的过程流程图。

[0038] 图7B是示出了并行地推测性地预取资源的示例性浏览器方法的过程流程图。

[0039] 图7C是示出了并行地预处理脚本的示例性浏览器方法的过程流程图。

[0040] 图8是示出了对被预取的资源进行处理的示例性浏览器方法的过程流程图。

[0041] 图9是示出了适合与各个方面结合使用的CSS引擎中的示例性功能组件的功能框图。

[0042] 图10是示出了用于执行规则匹配并且对若干个节点并行进行级联操作的示例性

样式化方法的过程流程图。

[0043] 图11A是适合用于各个方面中的示例性文档对象模型(DOM)树的说明。

[0044] 图11B是对应于图11A中示出的DOM树的任务有向无环图(directed acyclic graph,DAG)的说明。

[0045] 图12是适合与各个方面结合使用的示例性移动设备的组件框图。

[0046] 图13是适合与各个方面结合使用的示例性服务器的组件框图。

[0047] 图14是适合于实现各个方面的膝上型计算机的组件框图。

## 具体实施方式

[0048] 将参照附图对各个方面进行详细描述。只要可能,将贯穿附图使用相同的附图标记来表示相同或相似的部分。对特定例子和实施方式的引用是出于举例说明的目的,并不旨在限制本发明或权利要求的范围。

[0049] 本文中使用“示例性”一词表示“用作例子、实例或说明”。本文被描述为“示例性”的任何实施方式不必被解释为比其它实施方式更优选或更具优势。

[0050] 术语“移动设备”和“计算设备”在本文可互换地使用以指代蜂窝电话、智能电话、个人或移动多媒体播放器、个人数据助理(PDA)、膝上型计算机、平板计算机、智能本、掌上型计算机、无线电子邮件接收器、具有多媒体互联网功能蜂窝电话、无线游戏控制器和包括可编程处理器和存储器的类似的个人电子设备中的任何一种或全部。各个方面在可能具有受限的处理功率的移动设备(例如,蜂窝电话)中是特别有用的,同时这些方面通常在执行以动态、脚本和/或标记语言编写的脚本和/或应用的任何计算设备中也是有用的。

[0051] 网络浏览器是实现多个标准,需要支持传统行为,并且是高度动态且交互的复杂的软件应用。网络浏览器设计者通常以实现页面加载的快速响应时间(即使是在出现长网络延迟的情况下)、高性能(例如,支持网络应用的交互性)和高用户接口响应能力的混合优化以提供良好的用户体验为目标。

[0052] 各个方面通过启发法、推测、资源预取和/或采用了由现代多处理器移动设备架构实现的并发/并行的技术的使用来提供网络浏览器、浏览器方法和浏览器系统,所述网络浏览器、浏览器方法和浏览器系统被配置为实现快速响应时间、高性能和高用户接口响应能力。

[0053] 现代网络文档(例如,HTML网页、HTML文档等)可能引用大量的外部资源,并且每个被引用的外部资源可能包括对其它外部资源的引用。例如,HTML文档通常包括对图像、音频、级联样式表(CSS)和JavaScript®的引用,并且被引用的资源(例如,CSS、音频、JavaScript®)还可能包括对另外的外部资源(例如,图像、音频等)的引用。通常,并非需要(或者甚至使用)全部被引用的外部资源以在移动设备的电子显示器上正确地呈现网页。

[0054] 在移动设备上加载并呈现网页通常需要网络浏览器解析HTML文档以识别文档中引用的外部资源(图像、音频、CSS等)。传统浏览器解决方案可以向一个或多个网络服务器请求所识别的外部资源的每个资源,以及暂停网页的进一步处理直到从网络服务器接收到全部外部资源。由于现代网络文档中引用的大量资源(以及在外部资源下载的同时浏览器操作的暂停),当下载HTML和CSS代码/内容中发现的资源时,移动设备可能经历较慢的文档加载速度和较高的延迟时间。

[0055] 传统的网络浏览器解决方案可以通过在存储器中高速缓存部分网页来减少下次访问网页时必须下载的信息量,以尝试加速网页/文档的加载速度。然而,使用这些传统解决方案,网络浏览器无法在未先分析整个文档(即,网页),请求并接收文档和子文档中引用的大多数资源(如果不是全部资源)并且分析所接收的资源的情况下,在第一时间识别为呈现网页所需要的外部资源。因此,使用传统解决方案,直到整个文档已经被完全分析之后,才能确定文档需要的资源的精确集合。

[0056] 网络浏览器可以被配置为扫描网络文档以提前发现并下载全部被引用的外部资源。然而,由于可能要引用的大量资源,与当发现外部引用时对其请求的传统网络浏览器解决方案相比,盲目下载全部被引用的资源对于文档加载速度和延迟时间可能是更加有害的。例如,网站当中常见的做法是,对任何给定文档,例如使用站点常见的样式文件来引用比实际需要更多的资源。下载全部这些被引用的外部资源还可能降低文档加载速度、消耗过量带宽并且增加延迟时间。

[0057] 各个方面包括用于执行以下操作的浏览器系统和方法:通过使用推测/预测技术来对网络文档(HTML页面)进行预处理,以利用信息的不完整集合来识别可能需要的资源,以及请求/预取被确定为为了正确呈现网络文档而具有较高概率的需要的资源。这些资源的预取可以使得网络浏览器(因此也使得移动设备)能够更好地使用可用带宽,交迭传送延迟并且改进文档加载时间。

[0058] 各个方面可以基于启发法推测性地下载资源以改进文档加载和网络通信的效率和速度。各个方面可以计算、生成、选择和/或应用一个或多个启发法来使真阳性和真阴性的数量最大化,同时使假阳性和假阴性下载决定的数量最小化。各个方面可以通过使用在网络文档的初始扫描期间获得的信息来识别可能需要的资源,以使真阳性和真阴性的数量最大化。

[0059] 各个方面可以基于“CSS规则”启发法来推测性地下载资源,这种启发法对于发现所需资源和哪些不正确推测对网络浏览器(因此对于移动设备也是如此)没有显著负面影响是高度有效的。

[0060] 如在本申请中使用的,术语“组件”、“模块”、“系统”、“子系统”、“引擎”、“管理器”等旨在包括计算机相关实体,例如但不限于,硬件、固件、硬件和软件的组合、软件或被配置为执行特定操作或功能的执行的软件。例如,组件可以是但并不限于,在处理器上运行的过程、处理器、对象、可执行文件、执行的线程、程序、规程、软件应用和/或计算机。作为说明,运行在计算设备上的应用和计算设备二者都可以被称为组件。一个或多个组件可以位于执行的线程和/或过程内,并且组件可以位于一个处理器或核心上,和/或分布在两个或更多处理器或核心之间。另外,可以通过具有存储于其上的各种指令和/或数据结构的各种非暂时性计算机可读介质来执行这些组件。组件可以通过本地和/或远程过程、功能或规程调用、电子信号、数据分组、存储器读/写和其它已知的计算机、处理器和/或与过程有关的通信方法进行通信。

[0061] 本文使用术语“片上系统”以指代单个集成电路(IC)芯片,其包含集成在单个衬底上的多个资源和/或处理器。单个SOC可以包含用于数字、模拟、混合信号和无线频率功能的电路。单个SOC还可以包括任意数量的通用和/或专用处理器(数字信号处理器、调制解调处理器、视频处理器等)、存储器块(例如,ROM、RAM、闪存等)和硬件资源(例如,计时器、稳压

器、振荡器等)。SOC还可以包括用于控制集成硬件资源和处理器以及用于控制外围设备的软件。

[0062] 本文使用术语“多核处理器”以指代包含被配置为读取并且执行程序指令的两个或更多个独立处理核心(例如,CPU核心)的单个集成电路(IC)芯片或芯片封装。SOC可以包括多个多核处理器,并且SOC中的每个处理器都可以被称为核心。本文使用术语“多处理器”以指代包括被配置为读取并且执行程序指令的两个或更多个处理单元的系统或设备。

[0063] 本申请通常使用术语“应用编程接口”和其缩略语“API”以指代可以被第一软件组件用来与第二软件组件进行通信的任何软件接口。API可以包括用于例程、规程、功能、方法、数据结构、对象类型和变量的规范。API还可以包括将该API映射到另一高级编程语言的特征(句法或语义)的设施。这样的设施和/或映射本身可以是API,并被称为“语言绑定”或“绑定”。

[0064] 本申请通常使用术语“标记语言”以指代用于注释文本以使得处理器可以在句法上将注释与文本区分开的任何编程语言和/或系统。标记语言的例子包括:Scribe、标准通用标记语言(SGML)、超文本标记语言(HTML)、可扩展标记语言(XML)和可扩展超文本标记语言(XHTML)。

[0065] 本申请通常且可互换地使用术语“动态语言”和“脚本语言”以指代任何动态语言、脚本语言或用于编写在运行时间被解释和/或被编译的程序(本文作为“脚本”)的任何语言。这些术语还可以是指在受控的运行时间上运行并且被动态编译的任何语言。因此,各个方面的描述中的术语“动态语言”和“脚本语言”的使用不应当被解释为将权利要求限制为如下语言:通过源代码或字节代码来解释的语言,或者与被传统地编译为本地机器代码的程序一起执行的那些语言。本申请的范围内的动态语言和脚本语言的例子包括例如JavaScript®、Perl、Python和Ruby以及未来可能开发的其它类似语言。

[0066] 本申请通常使用术语“样式表语言”和“样式语言”以指代对结构文档的表示进行表达,从而文档的呈现样式可以与文档的内容分离的任何计算机语言。样式表语言的例子是级联样式表(CSS),其通常用于描述以标记语言编写的文档的表示语义。

[0067] 为了引用方便,贯穿本申请,HTML用作示例性标记语言,CSS用作示例性样式表语言,以及JavaScript®用作示例性动态脚本语言。然而,应当注意,除非权利要求明确地记载,否则本申请中HTML、CSS和JavaScript®的使用仅是为了说明的目的,而不应当被解释为将权利要求的范围限制到特定语言。

[0068] HTML是实现ISO/IEC 15445标准的标记语言。HTML的特征是一组标记标签(例如,注释),其用于描述网页,从而它们可以被诸如网络浏览器之类的软件应用显示。HTML通过为诸如标题、段落、列表、链接、引用和其它项目的文本指示结构语义来实现结构化文档的创建。

[0069] JavaScript®是实现ECMAScript语言标准(其被国际ECMA在ECMA-262规范中标准化)和/或ISO/IEC 16262标准的动态、弱类型、面向对象的脚本语言。JavaScript®支持对主机环境内的计算对象(例如,在移动设备处理器上执行的网络浏览器)进行有计划地访问。

[0070] 级联样式表(CSS)是用于描述网站的外观和格式的样式语言,并且旨在用于使文

档的表示与其内容分离。每个样式表可以包括具有以下格式的规则的有序集合:selector {property1:value;...propertyn:value;}。作为例子,以下CSS代码告知浏览器呈现其直接先辈是在红色背景上使用白色前景的

元素的全部cite元素:p>cite{color:white; background-color:red;}。对于网站常见的是包括上万个的这样的规则。

[0071] HTML可以嵌入和/或包括对JavaScript®代码的链接,其能够影响行为和/或包含HTML的页面的表示。嵌入/链接的JavaScript®代码还可以生成另外的HTML代码,其可以被插入到包含HTML的页面(即,嵌入了JavaScript®的HTML代码)中。

[0072] JavaScript®可以用于将功能嵌入到HTML代码以使得功能与HTML页面的文档对象模型(DOM)交互并且对其进行操纵。DOM是用于表示HTML中的对象并与其交互的与语言无关的约定,并且使得JavaScript®代码够访问且操纵包含HTML的页面。DOM树通常作为呈现网页的一部分而生成以识别组件、相对结构、关系和定义该页面的相应组件的行为。

[0073] HTML可以包括(例如,嵌入的和/或链接的)CSS代码。CSS代码可以被规定为单独的文件,其可以被存储在远程服务器上。传统CSS处理引擎(例如,WebKit或Firefox)在主浏览器线程中顺序解析CSS,但不支持高度的并行性或并发性。例如,当CSS代码被嵌入到HTML文档中时,HTML解析器不能解析HTML文档的其余部分,直到CSS引擎已经解析HTML文档报头中的样式元素。当HTML文档包括对若干个CSS文件的链接时,传统CSS处理引擎将顺序解析全部被链接的CSS文件。由于这些和其它原因,传统CSS处理引擎可能导致严重减速,特别是在大CSS文件的情况下(这很普遍)。

[0074] 在最近几年,移动电子设备(例如,蜂窝电话、平板计算机、膝上型计算机等)变得具有更多功能,并且现在通常包括多个处理器、片上系统(SoC)、多个存储器和使得移动设备用户能够在其移动设备上执行复杂且耗电量大的软件应用(例如,网络浏览器、视频流应用等)的其它组件。由于这些和其它改进,智能电话和平板计算机变得流行,并且取代膝上型计算机和台式机成为许多用户选择的平台。移动设备用户现在可以经由其移动设备的网络浏览器接入互联网来容易且方便地实现其许多日常任务。然而,现有网络浏览器和网络浏览器解决方案不支持高度的并行性或并发性,因此不能充分使用移动设备的多处理能力。

[0075] 各种示例性方法、系统和浏览器利用在现代移动设备中可用的并行性来改进页面和文档加载、网络应用以及网络通信的速度和效率。

[0076] 利用网络浏览器中的并发性是相对新的方法。大多数现有浏览器(例如,Firefox浏览器、基于WebKit的Chrome浏览器和Safari浏览器等)在基本架构上都是使用事件驱动模型来实现交互性的顺序引擎。由于移动设备和/或浏览器子系统之间的大量相关性(并且因为许多现有数据结构不是线程安全的),现有解决方案不支持高度的并行性或并发性。

[0077] Chrome和WebKit2为每个浏览器标签生成单独的过程,这提供不同网站之间的一些隔离,但是向操作系统委派使用多核心的责任。另外,就存储器和启动开销二者而言,这些过程是重量级的。因此,这些解决方案并未加速单个页面加载或者改进网络通信的效率,而是相对于执行相同应用的多个实例,简单地执行并行性。这样的标签级别的并行性没有解决移动浏览器的需求,其中,单个标签的性能往往是不够的,而用户一次不会打开许多标

签。

[0078] OP和OP2浏览器可以为每个网页(其被称为“网络实例”)生成过程的新集合,并且浏览器组件(例如,联网)可以在不同的过程中运行。然而,这些解决方案,像全部其它现有浏览器解决方案一样,本质上仍然是顺序的。例如,当网络操作可以在单独的过程中作为解析操作被执行时,网络过程仍然必须等待解析过程(反之亦然),这是因为每个操作都依赖于其它操作。即,当OP和OP2浏览器允许使用多个过程或线程时,这些解决方案在呈现网页中没有实现高度并行性,这是因为其没有解决用于下载、处理和呈现网页的浏览器处理算法的串行/顺序本质。

[0079] 各个方面包括高性能的网络浏览器和浏览器解决方案,其克服现有浏览器处理算法的串行/顺序本质,使用高速处理器和多处理器移动设备架构的多线程执行和并行处理能力,以及深入地利用并行来改进浏览器性能,降低网络延迟,并且改进移动设备的用户的用户体验。

[0080] 各个方面可以被实现在多个单处理器计算机系统和多处理器计算机系统上,包括片上系统(SOC)。图1示出了示例性片上系统(SOC)100的架构,其可以用在实现各个方面的计算设备中。SOC 100可以包括多个异构处理器,例如数字信号处理器(DSP)102、调制解调器处理器104、图形处理器106和应用处理器108。SOC 100还可以包括一个或多个协处理器110(例如,向量协处理器),其连接到异构处理器102、104、106、108中的一个或多个。每个处理器102、104、106、108、110可以包括一个或多个核心,并且每个处理器/核心可以执行独立于其它处理器/核心的操作。例如,SOC 100可以包括执行第一类型的操作系统(例如,FreeBSD、LINUX、OS X等)的处理器和执行第二类型的操作系统(例如,微软视窗(Microsoft Windows)8)的处理器。

[0081] SOC 100还可以包括模拟电路和定制电路114,其用于管理传感器数据、模数转换、无线数据传输,以及用于执行其它其它专门的操作,例如处理经编码的音频和视频信号以便在网络浏览器中呈现。SOC 100还可以包括系统组件和资源116,例如稳压器、振荡器、锁相回路、外围桥路、数据控制器、存储器控制器、系统控制器、接入端口、计时器和用于支持在计算机设备上运行的处理器和软件客户端(例如,网络浏览器)的其它类似组件。

[0082] 系统组件和资源116和/或定制电路114可以包括与外围设备(例如,摄像机、电子显示器、无线通信设备、外部存储器芯片等)对接的电路。处理器102、104、106、108可以经由互连/总线模块124被互连到一个或多个存储单元112、系统组件和资源116、以及定制电路114,所述互连/总线模块124可以包括可配置逻辑门阵列和/或实现总线架构(例如,核心连接(CoreConnect)、AMBA等)。可以通过诸如高性能片上网络(NoC)之类的高级互连来提供通信。

[0083] SOC 100还可以包括输入输出/模块(未示出),其用于与SOC外部资源(例如,时钟118和稳压器120)通信。SOC外部资源(例如,时钟118和稳压器120)可以被内部SOC处理器/核心(例如,DSP 102、调制解调器处理器104、图形处理器106和应用处理器108等)中的两个或更多个共享。

[0084] 除了上述讨论的SOC 100之外,还可以在各种各样的计算系统中实现各个方面,所述计算系统可以包括单个处理器、多个处理器、多核处理器或其任意组合。

[0085] 图2示出了可以用于实现各个方面示例性多核处理器的架构。多核处理器202可

以包括极为接近的(例如,在单个衬底、裸片、集成芯片上等)两个或更多个独立处理核心204、206、230、232。如果信号必须离开芯片,则处理核心204、206、230、232的接近使得存储器能够按照比可能高得多的频率/时钟速率来操作。此外,处理核心204、206、230、232的接近允许片上存储器和资源(例如,电压干线)的共享以及核心之间的更多协调合作。

[0086] 多核处理器202可以包括含有一级(L1)高速缓存212、214、238、240和二级(L2)高速缓存216、226、242的多级高速缓存。多核处理器202还可以包括总线/互连接口218、主存储器220和输入/输出模块222。L2高速缓存216、226、242可以比L1高速缓存212、214、238、240大(且慢),但是比主存储器220单元小(且快得多)。每个处理核心204、206、230、232可以包括处理单元208、210、234、236,其具有对L1高速缓存212、214、238、240的私有访问。处理核心204、206、230、232可以共享对L2高速缓存(例如,L2高速缓存242)的访问或者可以访问独立的L2高速缓存(例如,L2高速缓存216、226)。

[0087] L1高速缓存和L2高速缓存可以用于存储由处理单元频繁访问的数据,而主存储器220可以用于存储由处理核心204、206、230、232访问的较大的文件和数据单元。可以配置多核处理器202以使处理核心204、206、230、232按顺序地从存储器寻找数据,如果信息未被存储在高速缓存中,则首先查询L1高速缓存,然后是L2高速缓存,再然后是主存储器。如果信息未被存储在高速缓存或主存储器220中,则多核处理器202可以从外部存储器和/或硬盘存储器224寻找信息。

[0088] 处理核心204、206、230、232可以经由总线/互连接口218与彼此通信。每个处理核心204、206、230、232可以具有对一些资源独有的控制并且与其它核心共享其它资源。

[0089] 处理核心204、206、230、232可以彼此等同、异构和/或实现不同的专门功能。因此,从操作系统的角度(例如,可以执行不同的操作系统)或从硬件的角度(例如,可以实现不同的指令集/架构)来看,处理核心204、206、230、232不需要对称。

[0090] 多处理器硬件设计,例如上面参照图1和2所讨论的那些设计,可以包括往往位于相同硅片上的相同封装内的不同能力的多处理核心。对称的多处理硬件包括两个或更多个等同的处理器,所述处理器连接到由单个操作系统控制的单个共享的主存储器。非对称或“松耦合”的多处理硬件可以包括两个或更多个异构处理器/核心,所述异构处理器/核心中的每一个可以由独立的操作系统来控制并被连接到一个或多个共享的存储器/资源。

[0091] 图3A示出了一种加载并呈现HTML文档的示例性浏览器方法300。在方框302中,网络浏览器组件可以接收用于请求加载位于特定统一资源定位符(URL)处的HTML文档的用户输入。在方框304中,网络浏览器组件可以通过公知的超文本传输协议(HTTP)消息(该消息是经由互联网传送的)向网络服务器请求位于该URL处的HTML文档。在方框306中,网络浏览器组件可以从网络服务器接收位于该URL处的HTML文档。在方框308中,网络浏览器组件可以解析所接收的HTML文档以识别/发现在该HTML文件中引用的外部资源(图像、音频、CSS等)。

[0092] 在方框310中,网络浏览器组件可以向其中资源被维护的网络服务器请求所识别的外部资源,所述网络服务器可以包括提供该HTML文档的服务器或可经由互联网访问的任何其它服务器。在方框312中,网络浏览器组件可以从网络服务器接收所请求的外部资源。在确定框314中,网络浏览器组件可以确定所接收的资源中是否有任何一项引用其它外部资源。

[0093] 当网络浏览器组件确定所接收的资源引用其它外部资源时(即,确定框314=“是”),网络浏览器可以请求/接收由方框310-314中新接收资源所引用的那些其它/另外的外部资源。这些操作可以被反复执行,直到被引用的全部外部资源被下载完为止。

[0094] 当网络浏览器确定所接收的资源未引用任何另外的外部资源时(即,确定框314=“否”),则在方框316中,网络浏览器可以分析所接收的外部资源以确定为正确地呈现网页所需要的资源。在方框318中,网络浏览器可以使用所需要的下载资源来呈现网页。

[0095] 图3B示出了在示例性浏览器系统350中的示例性逻辑组件、信息流、操作和转换。浏览器系统350可以是软件应用/模块,其被配置为使得处理器执行各种操作以便从互联网检索信息和/或资源以及在计算设备(例如,移动设备)的电子显示器上呈现网页。

[0096] 浏览器系统350可以包括脚本组件362,其被配置为在各种阶段和/或在各种操作期间(例如,在页面加载操作期间和页面加载操作之后等)与网页交互以提供与外部模块380的交互性。外部模块380可以包括用户I/O模块(例如,鼠标、键盘等)和/或应用模块(例如,插件、GPS等)。在一个方面中,脚本362组件可以包括JavaScript®引擎,其被配置为编译和/或执行JavaScript®代码。

[0097] 在方框354中,浏览器系统350可以执行提取操作以从网络352中的服务器请求/接收编程指令356(例如,通过HTTP)。在方框358中,浏览器系统350可以翻译/解码所接收的编程指令356以生成HTML代码360。生成的HTML 360代码可以包括(即,嵌入或包括对其的引用)JavaScript®代码,该代码的执行可以生成另外的HTML代码以插入到包含HTML的页面(例如,其中包括JavaScript®的HTML代码)中。这种生成的HTML代码可以影响HTML页面的行为和/或表示。生成的HTML 360代码还可以包括样式表和/或CSS代码。

[0098] 在方框364中,浏览器系统350可以解析HTML 360代码(和被嵌入/被引用的JavaScript®代码)以生成HTML文档的文档对象模型(DOM)366。DOM 366可以表示HTML代码中的各种对象的内容、关系、样式和位置。浏览器“传递(pass)”和组件之间的通信可以经由DOM 366发生。“浏览器传递”可以是通过HTML文档的相关部分的、与单个迭代相关联的线程、过程或应用。在实施例中,浏览器传递可以是“工作项目”。

[0099] 如上面所提及的,JavaScript®代码可以被嵌入在HTML代码中,并且在同一时间,生成要被插入到包含HTML的页面的另外的HTML代码。为了实现代码的插入(并且确保正确的顺序),可能需要两个不同的过程来解释、解析和执行JavaScript®代码和包含HTML的代码。因此,在一个方面中,方框364的解析操作可以由多个过程或应用来执行。

[0100] 在方框368中,浏览器系统350可以执行样式化操作以例如通过将一个或多个样式表(例如,CSS)应用于HTML文档和/或所生成的DOM 366树,来生成修改后的DOM树370。

[0101] 在方框372中,浏览器系统350可以通过执行布局操作来“解决”页面布局374。在一个方面中,可以执行布局操作以使得当显示页面所需的内容变得可用时(例如,被下载、处理和/或添加到DOM),逐步地解决页面布局。

[0102] 在方框376中,浏览器系统350可以执行呈现操作以在计算设备的电子显示器上显示HTML文档的内容378。

[0103] 各个方面修改现有浏览器处理算法的潜在串行本质。各个方面可以包括动态和并

发浏览器系统,其支持高度并行性和/或并发性。各个方面可以在多个级别利用并发性。各个方面可以执行针对单个浏览器传递的并行算法以加速各种浏览器组件和/或操作的处理和/或执行时间。各个方面可以将浏览器传递交迭以加速总执行时间。

[0104] 图4和5根据各个方面示出了适合于在多个级别利用并发性的示例性浏览器系统500中的示例性组件、信息流和子系统。

[0105] 图4示出了浏览器系统500,其包括提取管理器组件502、DOM分配器组件504、HTML解析器组件506、HTML预扫描器组件508、图像解码器组件510、CSS引擎组件512、**JavaScript®**引擎组件514、布局和呈现引擎组件516、以及用户接口组件518。在一个方面中,浏览器系统500还可以包括沙盒(sandboxed) **JavaScript®**引擎组件530。这些组件502-530中的每个组件都可以是软件模块(例如,在处理器上运行的过程、执行的线程、线程池、程序等)。在各个方面中,组件502-530中的任何或全部组件都可以使用线程库(例如,Pthread等)或并行任务库(例如,英特尔线程构件,Cilk等)以支持并发性。

[0106] 在一个方面中,浏览器系统500、组件502-518、530可以被宽松地耦合并且被配置为支持并发性。

[0107] 提取管理器组件502可以被配置为从网络提取资源,对所提取的资源执行高速缓存管理,以及向其它浏览器组件提供来自网络的数据的通知。在一个方面中,提取管理器组件502可以被配置为按照资源在HTML文档出现的顺序来提取这些资源(即,不施加任何优先权)。在另一方面中,提取管理器组件502可以被配置为基于预先指派的优先权来指派优先权和/或提取资源。

[0108] DOM分配器组件504可以被配置为调度DOM更新、使对DOM树的访问串行化,以及管理各个浏览器组件之间的交互。其它子系统(即,其余的浏览器基础设施)可以分配工作项目(其还被称为“DOM分配器工作项目”)到并发的DOM分配器队列中。DOM分配器项目504可以被配置为从DOM分配器队列拉取工作项目,并且每次处理一个工作项目。在各个方面中,工作项目可以包括浏览器传递和/或事件(例如,计时器事件、来自用户接口的事件等)。

[0109] HTML解析器组件506可以被配置为接收HTML文档的传入(例如,部分的等)数据块(例如,经由DOM分配器工作项目等),以及通过执行HTML解析算法(例如,HTML5解析算法)来构建DOM树。HTML解析器组件506可以向提取管理器队列添加HTML文档中引用的外部资源,该提取管理器队列可被提取管理器组件502访问。HTML解析器组件506还可以通过在解析操作期间的合适时间调用**JavaScript®**引擎组件514来发起**JavaScript®**代码的执行。

[0110] HTML预扫描器组件508可以被配置为扫描HTML文档以快速确定HTML文档所请求/需要的外部资源。HTML预扫描组件508可以向提取管理器组件502分派任务(例如,通过通知、存储器写入操作等)以开始下载外部资源和/或基于该外部资源执行进一步处理。

[0111] 图像解码器组件510可以被配置为解码图像。例如,当提取管理器组件502接收到图像的完整数据时,其可以将该图像传递给图像解码器组件510,后者随后可以对该图像进行解码以供稍后使用。

[0112] CSS引擎组件512可以被配置为计算DOM元素的外观和感觉以供稍后的阶段(例如,布局和呈现阶段)使用。与上面讨论的图像解码操作564类似,提取管理器组件502可以将CSS样式表传递给CSS引擎以便解析并且发现要请求的新资源。

[0113] 在一个方面中,CSS引擎组件512可以包括CSS资源预取器组件520、CSS解析器组件522和DOM样式化器组件524。CSS资源预取器组件520可以执行CSS扫描和/或预取操作,其可以包括扫描CSS文档以快速确定CSS文档所请求/需要的外部资源。在另一方面中,CSS资源预取器组件520可以向提取管理器组件502分派任务以开始下载外部资源和/或基于该外部资源来执行进一步处理。

[0114] CSS解析器组件522可以被配置为读取CSS代码以及在存储器中创建数据结构(例如,CSS规则)的集合。DOM样式化器组件524可以被配置为使用由CSS解析器组件522创建的数据结构以确定DOM树中的节点的样式。对每个节点,CSS引擎组件512可以执行规则匹配操作以找到其选择器匹配节点的规则。这样的规则匹配操作对于每个节点可以返回许多(有时会冲突的)规则。在各个方面中,CSS引擎512可以被配置为使用级联操作来向规则指派权重并且选择具有最大权重的规则。

[0115] JavaScript®引擎组件514可以被配置为编译并执行JavaScript®代码。提取管理器502可以下载JavaScript®脚本并且将其发送给JavaScript®引擎组件514以被编译。HTML解析器506和/或DOM分配器504可以请求JavaScript®引擎组件514执行脚本。

[0116] JavaScript®引擎组件514可以包括用于编译任务/操作的线程池,并且可以被配置为并行编译多个脚本(JavaScript®代码)。在一个方面中,由于JavaScript®语义,可以在主引擎线程中顺序执行脚本的执行。在一个方面中,JavaScript®引擎组件514可以被配置为使得当HTML解析器506或者DOM分配器504(例如,对于用户接口事件)请求JavaScript®引擎组件514执行尚未被编译的脚本时,JavaScript®引擎组件514自动发起脚本的编译并且在尝试执行所请求的脚本之前等待编译的结果。

[0117] 在各个方面中,JavaScript®引擎组件514可以包括轻编译器(light compiler)526和全编译器(full compiler)528(例如,以支持自适应编译和JavaScript®代码的执行)。轻编译器526可以被配置为:针对不频繁重用的JavaScript®代码和/或对页面加载的优化,生成可执行代码。全编译器528可以被配置为:针对重重重用的JavaScript®代码和/或对交互性和网络应用优化,生成较高质量代码。在各个方面中,全编译器528的较慢的代码生成可以在重用代码的多个运行之间分摊。与轻编译器526相比,全编译器528可以实现针对迭代的网络应用的显著加速。例如,使用全编译器528,N体(N-body)模拟网络应用可以加快运行六个因子。

[0118] 沙盒JavaScript®引擎组件530可以是与主JavaScript®引擎组件514分离的隔离的JavaScript®引擎。沙盒JavaScript®引擎组件530可以包括全部组件、特征和功能JavaScript®引擎组件514。

[0119] 布局和呈现引擎组件516可以被配置为将经样式的DOM树转换为可视的网页。在一个方面中,布局和呈现引擎组件516可以被配置为将对DOM和CSS样式表的改变反映到移动设备的电子显示器上,从而用户可以查看并且与更新的HTML文档交互。对DOM和/或CSS的

改变可以是由于提取管理器组件502传送新资源、HTML解析器组件506更新DOM,由于JavaScript®引擎组件514计算等。

[0120] 在一个方面中,布局和呈现引擎516可以被配置为获取DOM信息的快照以及异步地执行布局和/或呈现操作。在另一方面中,布局和呈现引擎516可以被配置为同步地调用布局和/或呈现操作(例如,当JavaScript®使用查询布局信息的API时)。

[0121] 用户接口组件518可以被配置为管理浏览器系统500与移动设备用户之间的交互。用户接口组件518组件可以将用户交互(例如,触摸移动设备的电子显示器上的链接)翻译为功能/方法调用(例如,Java本地接口或者“JNI”方法调用),所述功能/方法调用创建工作项目以放置在DOM分配器队列中。

[0122] 在一个方面中,上面提及的全部组件502-518、530可以针对每个网页实例化一次。在另一方面中,提取管理器组件502以及布局和呈现引擎组件516可以是全局的,而其它组件(例如,504、506、508、510、512、514和518)可以针对每个网页或HTML文档实例化一次。

[0123] 图5示出了上面讨论的示例性浏览器系统500中的示例性子系统和信息流。具体地,图5示出了可以包括用户接口子系统552、资源管理器子系统554、每页DOM引擎子系统556、每页JavaScript®引擎子系统558和呈现引擎子系统560的浏览器系统500。

[0124] 子系统555-560中的每个子系统可以宽松耦合并且被配置为支持并发。子系统552-560可以被实现为软件模块(例如,在处理器上运行的过程、执行的线程、程序等)。子系统552-560的操作可以由参照图4讨论的组件中的一个或多个组件和/或任何单个或多处理器计算系统来执行。

[0125] 在一个方面中,资源管理器子系统554和呈现引擎子系统560可以被实例化一次(例如,可以是全局的),而每页DOM引擎子系统556和每页JavaScript®引擎子系统558可以针对每个网页或HTML文档被实例化一次。

[0126] 用户接口子系统552可以被配置为执行用于管理与浏览器系统550的用户交互的各种操作,包括:将用户交互(例如,触摸移动设备的电子显示器上的链接)翻译为功能/方法调用,所述功能/方法调用创建工作项目以放置在DOM分配器队列中;检测和/或发送事件以校正每页JavaScript®引擎子系统558的实例;和/或向资源管理器子系统554发送统一资源定位符(URL)/统一资源标识符(URI)信息(例如,通过存储器写入操作、功能调用等)。

[0127] 资源管理器子系统554可以被配置为执行预取操作562、HTML预扫描操作563、图像解码操作564、CSS扫描/预取操作566和JavaScript扫描/预取操作567。作为例子,这些操作可以由提取管理器502、HTML预扫描器508、图像解码器510、CSS引擎512和/或JavaScript引擎514、530组件或由参照图4所讨论的组件的任意组合来执行。

[0128] 预取操作562可以包括:从与URL/URI相对应的网络服务器请求/接收资源和/或编程指令,翻译或解码所接收的编程指令以生成HTML,以及发送所生成的HTML代码以校正每页JavaScript®引擎子系统558的实例(例如,通过存储器写入操作等)。

[0129] 生成的HTML代码可以嵌入和/或引用JavaScript®代码、CSS代码、图像和各种其它资源。HTML文档中最常引用的资源是图像、CSS样式表和JavaScript®资源。样式表和JavaScript®源还可以引用其它外部资源。在一个方面中,可以扫描所生成的HTML代码,从

而可以提前提取(例如,作为预取操作562的一部分)由HTML文档识别的全部引用(包括被嵌入的或被引用的样式表和JavaScript®源)。

[0130] HTML预扫描操作563可以包括扫描生成的HTML代码以快速发现请求/需要的外部资源,以及告知提取管理器和/或预取器,其可以开始下载外部资源和/或基于所发现的外部资源执行进一步处理。在一个方面中,可以执行外部资源的下载作为上面讨论的预取562操作的一部分。在一个方面中,HTML预扫描器操作和预取操作562可以被并发地执行(例如,在单独的线程/过程中)。

[0131] 图像解码操作564操作可以包括通过呈现引擎子系统560来解码图像以供稍后使用。响应于确定图像的完整数据集已被下载(例如,通过作为预取562操作的一部分而执行的存储器写入操作等)和/或响应于接收通知(例如,从提取管理器组件502),可以执行图像解码操作564。在一个方面中,可以与HTML预扫描操作563和预取操作562并发地执行图像解码操作564。

[0132] CSS扫描/预取操作566可以包括扫描嵌入在所生成的HTML代码的(或由其引用的)中的CSS样式表以快速发现CSS样式表所请求的请求/需要的外部资源。在一个方面中,CSS扫描/预取操作566可以包括告知提取管理器和/或预取器,其可以开始下载所发现的外部资源。在一个方面中,CSS扫描/预取操作566可以包括发起对所发现的外部资源的下载。在一个方面中,响应于提取管理器组件502向CSS引擎组件512发送一个或多个CSS样式表,可以(例如,由CSS源预取器520)在CSS引擎组件512中执行CSS扫描/预取操作566。在一个方面中,CSS扫描/预取操作566可以与图像解码操作564、HTML预扫描操作563和预取操作562并发地被执行。

[0133] 每页DOM引擎子系统556可以被配置为执行HTML解析操作568、CSS解析操作570、计时器操作572、样式化操作574和用于管理事件的操作576。在一个方面中,每页DOM引擎子系统556的操作可以与其它子系统552、554、558、560的操作并发地被执行。

[0134] HTML解析操作568可以包括:解析所接收的HTML代码,将HTML标记标签与实质内容分离,和/或生成所接收的HTML代码的DOM。HTML解析操作568还可以包括识别HTML文档中引用的外部资源,从而所识别的外部资源可以被提取管理器502下载和/或作为预取操作562的一部分。HTML解析操作568还可以包括在HTML代码的解析期间(例如,当发现JavaScript®时等)(例如,通过调用执行操作578)发起JavaScript®代码的执行。

[0135] CSS解析操作570和样式化操作574可以包括将一个或多个CSS样式表应用于所生成的DOM树(或者基于CSS样式表生成修改后的DOM树)。在各个方面中,可以并发地执行HTML解析操作568、CSS解析操作570和样式化操作574中的任何一项或全部。

[0136] 计时器操作572可以包括管理或响应与计时器和/或计时器类(例如,系统计时器)相关的事件和/或状况。

[0137] 事件576操作可以包括管理各种事件,例如计时器事件和用户接口事件(例如,响应于用户触摸移动设备的电子显示器上的链接而生成的事件)。

[0138] 每页JavaScript®引擎子系统558可以被配置为执行JavaScript®执行操作578和JavaScript®编译操作580。

[0139] 在各个方面中,每页DOM引擎子系统556和/或资源管理器子系统554可以被配置为发送HTML代码中嵌入的(或引用的)JavaScript®代码以校正每页JavaScript®引擎558的实例以便进行编译和/或执行(即,通过执行578和编译580操作)。在这方面中,JavaScript®引擎558可以基于JavaScript®编译和/或执行操作578、580的结果来更新/修改生成的DOM树。

[0140] 呈现引擎子系统560可以被配置为执行布局操作582和呈现操作584。例如,呈现引擎子系统560可以从每页DOM引擎子系统556接收DOM树和/或布局树(例如,通过存储器写入、调用、通知等),解决每页布局(通过布局操作582),以及在计算设备的电子显示器上显示内容(通过呈现操作584)。在一个方面中,执行布局操作582可以包括:当另外的内容变得可用于呈现引擎子系统560时(例如,被下载、处理和/或添加到DOM树),逐步地解决页面布局。在各个方面中,可以并发地执行布局操作582和/或呈现操作584中的任何一项或全部。

[0141] 如参照图4和图5所讨论的,HTML解析器506和/或CSS解析522可以发现用于呈现HTML文档所请求/需要的外部资源(图像、音频、CSS、JavaScript®等)并且例如经由提取管理器502和/或作为预取操作的一部分来请求下载所发现的外部资源。

[0142] 移动设备在下载HTML和CSS代码/内容中发现的资源时可能经历高延迟时间。例如,由于HTML5规范中的特质,HTML解析器必须等待脚本元素(例如,<script>块)完成执行后才能够继续解析HTML文档的剩余部分。因此,如果网页引用脚本元素之后的外部资源,则提取该资源的操作不能与等待脚本元素完成执行的操作交迭。这往往增加了下载和显示网页所要求的时间。

[0143] 在各个方面中,浏览器系统500可以被配置为在脚本元素之前推测性地解析以发现新资源,而不等待脚本元素完成执行。在这些方面中,可以强制浏览器系统500丢弃推断解析的结果中的一些结果(例如,当JavaScript®通过文档写入API等将新内容插入到DOM树中时)。

[0144] 在一个方面中,浏览器系统500可以被配置为执行积极的资源预取操作以尽早地发现所请求/需要的资源,以及请求并行地提取/下载多个资源。以这种方式,各个方面可以防止浏览器系统500被强制丢弃推断解析的结果中的一些结果,并且可以屏蔽网络延迟,使用更多的可用带宽,以及减少用于等待资源到达所消耗的总时间。

[0145] 浏览器系统500可以被配置为执行积极的资源预取操作,其可以包括通过沙盒执行的推测性资源提取。在各个方面中,这些积极的资源预取操作可以作为HTML预扫描操作563、CSS预取操作566或二者的一部分被执行。

[0146] 参照图4-5,为促成积极的资源预取操作而执行的HTML预扫描操作563可以包括:获得HTML文档中的全部“id”、“类别”和/或“样式”属性,快速发现HTML文档中引用的外部资源,以及触发对来自网络的所发现资源的下载。HTML预扫描器508可以“大致解析”HTML以便于发现资源,而不执行向HTML解析器506要求的任何实质的或计算密集的处理(例如,构建DOM树)。通过放弃这些复杂解析操作,HTML预扫描操作563可以与HTML解析操作568并发执行(和在其之前运行),而不必等待脚本元素完成执行。

[0147] 在一个方面中,当网络分组到达时,其可以被独立地发送给HTML预扫描器508和

HTML解析器506。在一个方面中,通过与(非推测性的)HTML解析器570操作并行地执行HTML预扫描操作563可以进一步减少等待用于资源到达所消耗的时间。

[0148] 如上面所讨论的,网络浏览器系统500可以包括:CSS解析器522,其被配置为快速扫描CSS文档;以及CSS资源预取器520,其被配置为执行CSS预取操作。在一个方面中,CSS样式表可以被分配到负责并发解析CSS的线程池。如果CSS规则还包含其它外部资源,则CSS资源解析器可以基于其在HTML文档中实际被引用的可能性,做出关于是否发起针对所述其它外部资源的预取的决定。在一个方面中,CSS资源预取器520可以被配置为下载(或发起下载)特定范围/数量的引用资源(下载太少的资源可能意味着,当稍后对DOM树样式化时,更多的新资源将被DOM样式化器524发现,这可能导致额外延迟)。

[0149] 网站当中的习惯作法是,对于任何给定文档,通过例如使用站点常见的样式文件来引用比实际需要多得多的资源。下载全部包括的资源可能消耗过量带宽并且减慢页面加载。在各个方面中,CSS解析器522可以被配置为采用由HTML预扫描器508发现的“id”和“类别”属性来确定CSS规则是否可能被匹配。如果CSS规则选择器中引用的全部属性值已被HTML预扫描器508查看/评估,则可以确定规则可能匹配至少一个DOM树元素,并且浏览器系统500可以发起对与该CSS规则相对应的资源的下载。这种“CSS规则”启发法非常有效,并且错误的决定对浏览器系统500的操作没有显著的负面影响。缺失的资源可以(经由DOM样式化器组件524)在DOM样式阶段期间被发现,以下载该资源所要求的延迟作为代价。

[0150] 在一个方面中,HTML预扫描器508可以被配置为识别和/或发现不须执行JavaScript®就可以被发现的资源。

[0151] 如上面所讨论的,由于HTML5规范的特质,例如要求HTML解析器等待脚本元素(例如,<script>块)完成执行后才能继续解析,因此当下载HTML和CSS代码/内容中发现的资源时,移动设备可能经历较高的延迟时间。另外,现代网络文档(例如,HTML页面、HTML文档等)可能引用大量外部资源,并且每个外部资源可能包括对其它外部资源的引用。例如,HTML文档通常包括对各种外部资源的引用,例如图像、音频、级联样式表(CSS)和JavaScript®,并且被引用的资源(例如,CSS、JavaScript®)还可能包括对另外的外部资源(例如,图像、音频等)的引用。

[0152] 文档加载时间(即从请求文档直到其准备好在屏幕上显示的时间)由输入/输出成本(例如,所需资源的网络传送)决定。需要加载要求的全部资源的最小文档加载时间受资源存储单元与计算设备之间的连接带宽约束。另外,向显示设备传送文档资源招致延迟成本。各个方面可以被配置为尽早开始资源传送以更好地利用可用带宽、交迭传输延迟并且改进文档加载时间。

[0153] 如上面所提及的,由于并非需要(或者甚至使用)全部被引用的外部资源以呈现给定网页,因此递归地下载全部被引用的资源可能浪费大量的带宽和功率。另外,当任何资源不是立即可用时,在页面能够被正确呈现之前,浏览器必须等待直到其接收并且分析那些资源为止。这增加了加载和/或呈现网页所需要的时间(例如,文档加载时间)的量,并且降低了用户体验。

[0154] 传统解决方案尝试使用诸如将部分网页高速缓存在存储器中以减少下次页面访问时必须下载的信息之类的技术来加速呈现网页。然而,使用传统解决方案,网络浏览器无

法在未先分析整个文档(即,网页),请求并且接收文档和子文档中引用的大多数资源(如果不是全部资源)并且分析所接收资源的情况下,在第一时间识别为呈现网页所要求的外部资源。因此,使用传统解决方案,直到整个文档已经被完全分析之后,才能确定文档所要求的资源的精确集合。

[0155] 为了克服现有解决方案的这些限制,各个方面可以使用推测/预测技术,以便在整个文档被分析之前识别为呈现网页或文档所需要的资源。

[0156] 通常,(基于不完整的信息集合)推测性地预测资源是否是需要的导致以下四种可能结果中的一种:真阳性、真阴性、假阳性和假阴性。真阳性结果是资源被推测性地下载并且是实际上稍后需要的。真阴性结果是资源未被推测性地下载并且是不需要的。假阳性结果是不需要的资源被推测性地下载(其浪费带宽和能量),以及假阴性结果是资源没有被推测性地下载但是是需要的(因此相对于来自推测性预处理的资源,没有收益)。

[0157] 真阳性结果和真阴性结果是有益的并且是被期望的,这是因为这样的决定通过减少页面加载时间来改进用户体验。然而,假阳性结果和假阴性结果是不利的。例如,假阴性可能导致在文档(例如,HTML文档)呈现期间请求资源,这可能延长文档加载时间,直到资源可用为止。由于对浏览器来说,为正确地呈现文档不需要该资源,因此浪费了计算和网络资源(带宽、处理等)。

[0158] 各个方面包括网络浏览器系统,其被配置为基于启发法来执行推测性的资源下载操作,以使真阳性和真阴性的数量最大化,同时使假阳性和假阴性的下载决定最小化。

[0159] 图6示出了处理HTML文档以发现为正确呈现网页所需要的外部资源(图像、音频、CSS、JavaScript®等)以及在页面加载/呈现操作之前预取所发现的资源的示例性浏览器方法600。方法600的操作可以由执行被适当配置的网络浏览器的具有单个处理器或多处理器计算系统的处理器来执行。

[0160] 参照图6,在方框602中,网络浏览器可以发起或调用扫描操作(例如,经由HTML预扫描器508、CSS引擎512等)以扫描HTML文档和/或CSS文档的结构信息和/或发现资源。在一个方面中,扫描操作可以作为HTML预扫描操作563的一部分被执行。在一个方面中,扫描操作可以作为CSS扫描操作566的一部分被执行。在各个方面中,扫描操作可以与HTML和CSS解析操作568、570并发地被执行或者独立于其被执行。在各个方面中,扫描操作可以由过程、线程、应用、工作项目和/或浏览器传递来执行。

[0161] 在方框604中,扫描操作(例如,HTML和/或CSS扫描操作)可以确定(即,预测、推测)哪些被发现的资源可能是需要的。在方框606中,扫描操作可以向浏览器提取组件(例如,向提取管理器502)发出资源请求(例如,通过存储器写入操作等)以开始下载被确定为很可能需要的资源。在一个方面中,作为方框606的一部分,两个或更多个资源请求可能被并行或并发地发出(或发送)。在一个方面中,每个资源请求可能产生新过程和/或被不同的执行线程处理。在方框608中,扫描操作可以继续扫描HTML文档和/或CSS文档以发现另外的需要的资源。方框604-608中的操作可以重复,直到全部外部资源被发现和/或整个HTML文档被扫描为止。

[0162] 在方框610中,网络浏览器可以发起或者调用提取操作(例如,经由提取管理器502)以下载资源请求(例如,方框606中的扫描操作发出的资源请求)标识的一个或多个资源。

[0163] 在方框612中,网络浏览器可以扫描被下载的资源以发现对外部资源的另外的引用。作为方框612的一部分,网络浏览器可以发起或调用新过程或执行线程以执行扫描操作。在一个方面中,作为方框612的一部分,网络浏览器可以发起或者调用CSS扫描操作566。在一个方面中,作为方框612的一部分,网络浏览器可以发起或者调用HTML扫描操作或者HTML预扫描563操作。

[0164] 在方框614中,网络浏览器可以基于扫描被下载的资源来确定(即预测、推测)可能需要的被发现的资源。在方框616中,网络浏览器可以向浏览器提取组件(例如,向提取管理器502)发出另外的资源请求(例如,通过存储器写入操作等)以下载被确定为很可能需要的资源。在一个方面中,这些另外的资源请求中的每个请求可能产生其它过程和/或可能由不同的过程或执行线程处理。可以重复方框610–616中的操作直到全部外部资源被发现和/或被下载为止。在一个方面中,方框602–608的操作可以与方框610–616中的操作并行地被执行。

[0165] 与传统的HTML解析器不同,上面参照图6讨论的扫描操作不执行对被扫描的HTML文档的错误校正或执行遇到的JavaScript®代码。这使得能够快速执行扫描操作。另外,与传统的HTML解析器不同,上面讨论的扫描操作可以被并行地或并发地执行(例如,在独立的线程或过程中等),这使得各个方面能够更加充分地利用现代计算设备中普遍存在的多处理器架构。此外,上面讨论的扫描过程可以扫描HTML文档中引用的资源(例如,CSS文档),这也不在传统的HTML解析器中执行。

[0166] 通常,如果扫描操作(例如,HTML预扫描操作563、CSS扫描操作566等)仅扫描HTML文档的结构,则可以正确地推测关于需要的资源(即仅产生真阳性),除非例如在文档中存在结构错误(因为扫描器不执行错误校正)或者当其被解析时,文档的替代文档中存在嵌入的JavaScript®代码(因为扫描器不执行JavaScript®)。

[0167] 在一个方面中,为了使真阳性和真阴性的数量最大化,扫描操作(例如,HTML预扫描操作563、CSS扫描操作566等)可以使用在HTML文档的初始扫描期间获得的信息来识别可能需要的资源。

[0168] 图7A示出了使用推测技术和启发法来发现文档资源以便进行推测性下载的示例性浏览器方法700。文档资源可以包括图像、CSS文件、JavaScript®脚本等。浏览器方法700使得HTML文档扫描器和多个CSS文档扫描器能够并行执行,智能识别可能需要的资源,减少根据推测资源请求和/或预取操作得出的假阴性的数量。在一个方面中,浏览器方法700可以使用启发法(例如,“CSS规则”启发法)以使假阳性的数量最小化。

[0169] 在浏览器方法700的方框702中,HTML文档扫描器(例如,HTML预扫描器508)可以开始扫描HTML文档以发现资源并且获得与HTML文档中包括的HTML元素相关联的(或其提及的)全部URL/URI和HTML“id”、“类别”和/或“样式”属性。HTML文档扫描器可以独立于HTML解析器和/或与其并行执行。

[0170] 在方框704中,HTML文档扫描器可能遇到由HTML文档中包括的URL/URI和/或HTML元素引用的外部资源。在方框706中,HTML文档扫描器可以发出请求(例如,向提取管理器)以下载HTML文档中引用的遇到的资源。在一个方面中,HTML文档扫描器可以被配置为调用对每个遇到的外部CSS资源的下载和/或解析(例如,当外部资源被扫描器遇到时等等)。在

一个方面中,外部CSS资源的下载可以使得CSS文档扫描器(例如,CSS引擎512等)开始扫描CSS文档。

[0171] 在方框708中,HTML文档扫描器可以遇到和/或收集HTML id、类别和样式属性。在方框710中,HTML文档扫描器可以向CSS文档扫描器发送所遇到/所收集的信息(即,与收集id、类别和样式属性有关的信息)。在一个方面中,发送所收集的信息可以包括向CSS文档扫描器发送每个遇到和/或识别的HTML id、类别和样式属性。

[0172] 在方框712中,HTML文档扫描器可以继续扫描HTML文档以发现另外的资源。在确定框714中,HTML文档扫描器可以确定其是否已完成对HTML文档的扫描。当HTML文档扫描器确定其已完成对HTML文档的扫描时(即,确定框714=“是”),则在方框716中,HTML文档扫描器可以通知CSS文档扫描器(例如,CSS引擎512、执行CSS扫描操作566的过程等),其已完成对HTML文档的扫描(例如,通过存储器写入操作、方法调用、通知等)。当HTML文档扫描器确定其还未完成对HTML文档的扫描时(即,确定框714=“否”),则在方框702中,HTML文档扫描器可以继续扫描HTML文档以发现另外的资源。

[0173] 在浏览器方法700的方框719中,CSS文档扫描器可以为了外部资源而开始扫描CSS文档。可以由提取管理器获得的CSS文档的可用性来触发方框719中CSS文档的初始化(例如,响应于作为方框706的一部分而执行的操作等)。在一个方面中,CSS文档的扫描可以与HTML文档的扫描(例如,方框702-716中的操作)并行地被执行。因此,CSS文档扫描器可以扫描所接收的CSS文档以识别那些文档中引用的外部资源,同时HTML文档扫描器继续扫描HTML文档(例如,识别另外的CSS文档以下载等)。此外,可以存在并行执行的多个CSS文档扫描器(例,如当多个CSS文档被下载时)。

[0174] 在方框720中,CSS文档扫描器可以从HTML文档扫描器接收与HTML id、类别和/或样式属性有关的信息。在方框721中,CSS文档扫描器可以确定所接收的信息是否将(与所接收的HTML id、类别和/或样式属性相关联的)CSS规则和/或外部资源标记或标识为HTML文档可能需要的和/或使用的。在方面中,作为方框721的一部分,CSS文档扫描器可以确定与CSS规则相关联的每个HTML id、类别和/或样式属性是否已经被HTML文档扫描器遇到。

[0175] 在确定框722中,CSS文档扫描器可以确定(与所接收的HTML id、类别和/或样式属性相关联的)CSS规则和/或外部资源是否可能是HTML文档所需要的和/或使用的。在一个方面中,作为确定框722的一部分,CSS文档扫描器可以确定HTML文档提及的每个URL/URI和HTML id、类别和/或样式属性是否已经被遇到。

[0176] 当CSS文档扫描器确定CSS规则和/或外部资源可能是HTML文档所需要的和/或使用的时(即,确定框722=“是”),则在方框724中,CSS文档扫描器可以立即请求下载该CSS规则引用的资源,例如通过执行存储器写入操作和/或通知提取管理器502。

[0177] 在一个方面中,当确定HTML文档所提及的每个URL/URI和HTML id、类别和/或样式属性已经被遇到时,CSS文档扫描器可以确定CSS规则和/或外部资源可能是被需要的。

[0178] 当CSS文档扫描器确定CSS规则和/或外资资源不可能是HTML文档所需要的和/或使用的时(即确定框722=“否”),则在方框723中,CSS文档扫描器可以在存储器存储与资源引用的列表中的CSS规则(例如,接收的HTML id、类别和/或样式属性)有关的信息。在方框725中,如果需要的话(例如,当存在要被扫描/处理的另外的元素等时),CSS文档扫描器可以继续扫描CSS文档。

[0179] 在方框726中,CSS文档扫描器可以从HTML文档扫描器接收通知,其指示HTML文档扫描器已经完成对HTML文档的扫描。在方框727中,CSS文档扫描器可以从存储器中存储的资源引用的列表中取回与CSS规则有关的信息并且评估所取回的信息。

[0180] 在确定框728中,CSS文档扫描器可以确定所取回的信息是否标记/标识CSS规则和/或外部资源是HTML文档所需要的(或者可能需要的)。在方面中,作为确定框728的一部分,CSS文档扫描器可以确定与所取回的CSS规则相关联的每个HTML id、类别和/或样式属性是否已经被HTML文档扫描器遇到和/或处理。

[0181] 当CSS文档扫描器确定所取回的信息标记/标识CSS规则/外部资源可能是HTML文档所需要的和/或使用的时(即,确定框728=“是”),则在方框729中,CSS文档扫描器可以请求下载与该CSS规则相对应的资源。以这种方式,由同时扫描HTML文档和CSS文档导致的假阴性的数量可以被最小化。另外,各个方面可以在具有很少或不增加数据传送成本以及由于处理器和网络接口/无线电的使用减少而获得的较少的功耗的情况下,减少文档加载时间(并因此提高响应能力)。

[0182] 返回图7A,当CSS文档扫描器确定所取回的信息没有将外部资源标记或标识为是HTML文档所需要的(或可能需要的)时(即,确定框728=“否”),则在方框721中,CSS文档扫描器可以从存储器取回下一规则。可以重复方框720-722的操作,直到HTML文档扫描器在存储器中存储的全部CSS规则都已被评估为止。

[0183] 在各个方面中,HTML文档扫描器和/或CSS文档扫描器可以使用比上述CSS规则更精确的启发法以改进性能。例如,在一个方面中,HTML文档扫描器可以被配置为扫描嵌入式JavaScript®代码以获得URL和/或可能修改HTML文档的命令。类似地,在一个方面中,CSS文档扫描器可以被配置为记录与HTML标签有关的分层信息,所述HTML标签与每个遇到的ID相关联,这可以使得CSS文档扫描器能够识别并且拒绝更多的潜在假阳性。

[0184] 在传统浏览器中,HTML解析器通常负责识别全部外部资源并且通过网络向服务器请求它们。如上面所讨论的,当这些资源在HTML文档中被明确规定时,各个方面可以预取这些资源并且在页面加载中比传统浏览器早得多地发出请求。另外,各个方面可以并行地预取和/或处理资源。

[0185] 软件开发者越来越多地使用脚本(例如,JavaScript®代码®)来动态地确定特定的应用设备组合(例如,网络浏览器—移动设备组合)将需要的资源。例如,脚本可以对与客户端(例如,浏览器)和计算设备有关的各种因素进行评估以识别要下载的资源。这样的脚本可以基于评估因素实质上动态地针对资源(例如,图像、CSS、其它JavaScript®等)构建URL。因此,HTML文档可能需要HTML文档中没有明确标识的资源,并且所述资源只可能通过执行HTML文档中包括的JavaScript®代码来确定。

[0186] 由于这种JavaScript®代码可以改变包含HTML(和HTML代码本身)的状态、行为和/或表示,因此要求HTML解析器顺序地和/或按照HTML规范中定义的以下排序规则来执行遇到JavaScript®代码(或脚本)。例如,当HTML解析器遇到脚本标签(即,用于定义客户端侧脚本(例如,JavaScript®脚本)的<script>标签)时,HTML解析器必须等待该脚本被下载

和执行后,才可以继续解析HTML文档的剩余部分。因此,可以在JavaScript®脚本(即,JavaScript®代码内的<script>标签)的执行期间使全部资源请求串行化(即,要求一个接一个地执行)。另外,对于HTML文档扫描操作(例如,HTML预扫描操作563等)来说,静态地预测为正确地呈现网页而需要的资源可能是更加困难的。

[0187] 各个方面可以通过在沙盒JavaScript®引擎530中推测性地预取资源来克服这些和其它限制,这使得浏览器系统500能够发现并且下载HTML文档中没有明确请求的资源,前述操作与其它浏览器操作(例如,HTML解析)和其它资源请求并行发生。这些方面还可以使得浏览器系统500能够并行执行多个JavaScript®脚本,而不会无意地修改浏览器状态。

[0188] 各个方面可以在脚本(例如,JavaScript®代码)一被发现时就执行它们,前述操作与其它浏览器操作(例如,HTML预扫描563、HTML解析568等)和/或其它脚本并行执行。为了避免干扰网页的正常处理,可以在沙盒JavaScript®引擎530中执行脚本,该引擎是与其它浏览器组件隔离的和/或分离的(例如,以不影响主JavaScript®引擎的操作)。在沙盒JavaScript®引擎530中执行脚本防止系统在脚本的并行执行期间无意地修改浏览器状态。在一个方面中,可以在沙盒JavaScript®引擎530的单独实例(例如,线程)中执行各脚本。

[0189] 各个方面可以修改在浏览器客户端与JavaScript®引擎530之间的API。

[0190] 通常,脚本引擎(例如,JavaScript®引擎514、530、558)向浏览器API(即,使得脚本调用浏览器操作的接口)提供绑定(即,用于映射语言的API)以调用浏览器操作(例如,操纵DOM、访问网络等)。

[0191] 在一个方面中,JavaScript®引擎530可以监视向网络请求资源的浏览器API。JavaScript®引擎530可以修改绑定(或者向脚本引擎提供单独的绑定集合)以使得资源请求被重定向到不同的浏览器组件,例如预取器组件。以这种方式,资源请求和/或收集的信息可以被直接传递到预取器组件以便进一步处理。

[0192] 沙盒JavaScript®引擎可以扫描JavaScript®代码并且仅执行与发现外部资源最相关的代码的选择部分和/或选择操作。由于扫描操作仅涉及发现脚本可能请求的资源,因此扫描操作不受到HTML规范规则的约束,并且不必运行/执行全部遇到的代码。通过不完全执行全部遇到的代码,沙盒JavaScript®引擎可以快速地执行JavaScript®扫描操作。

[0193] 沙盒JavaScript®引擎可以应用启发法以进一步加速JavaScript®扫描操作。作为例子,这样的启发法可以包括限制:总执行时间(例如,每个脚本或操作最多消耗10ms等)、循环迭代的数量(例如,仅处理循环中的前10个迭代等)、递归深度、支持的特征、抽象解释等。

[0194] 各个方面可以限制对象和数据结构(例如,哈希表、阵列等)的大小以进一步加速JavaScript®扫描操作,这是因为这样的结构通常不影响资源的相关性。

[0195] 软件开发者往往在其代码中使用常见的模式、框架和/或服务(本文统称为“模式”)。各个方面可以在代码中检测普遍性/模式(例如,在解析、分析、编译期间等)并且仅执行与发现资源有关的模式(或者模式识别的 JavaScript® 代码的一部分)。在一个方面中,不同于完全遵循和保守的代码生成,沙盒 JavaScript® 引擎可以被配置为针对最常见的模式(例如,通过积极的编译器优化)。可以使用各种已知的模式识别技术来检测模式,例如在代码中检测关键字(其是相对简单的操作)和/或分析页面和/或脚本的结构(其是相对复杂的操作)。

[0196] 图7B示出了通过沙盒 JavaScript® 引擎中的脚本的并行处理来并行地推测性地预取资源的示例例方法730。方法730的操作可以与本文讨论的其它浏览器操作并行地被执行。

[0197] 在方法730的方框732中,HTML文档扫描器(例如,HTML预扫描器508)可以开始扫描HTML文档以获得结构信息和/或发现资源。在方框734中,HTML文档扫描器可以遇到JavaScript®脚本,以及向沙盒 JavaScript® 引擎发送所遇到的脚本(例如,通过存储器写入操作、重定向的资源请求、修改后的绑定等)以立即执行所遇到的脚本。在方框732中,HTML文档扫描器可以继续扫描HTML文档以获得结构信息和/或发现资源。在一个方面中,响应于遇到脚本,HTML文档扫描器可以生成(或者产生)沙盒 JavaScript® 引擎。

[0198] 在方框735中,沙盒 JavaScript® 引擎可以开始扫描脚本以发现资源。在方框736中,沙盒 JavaScript® 引擎可以推测性地执行脚本(或者脚本中包括的 JavaScript® 代码的一部分)。脚本的推测性执行可以包括仅执行与发现外部资源最可能相关的操作和/或代码的一部分。在各个方面中,推测性的执行操作可以与其它浏览器操作(例如,HTML预扫描563、HTML解析568等)和/或其它脚本的执行(是否是推测性的)并行地被执行。

[0199] 在一个方面中,脚本的推测性执行可以包括仅执行对应于与发现资源有关的模式的 JavaScript® 代码的一部分。

[0200] 在一个方面中,作为方框736的一部分,沙盒 JavaScript® 引擎可以基于启发法来进行 JavaScript® 代码的推测性执行(例如,以减少执行时间)。这样的启发法方法可以包括限制:总执行时间、循环迭代的数量、递归深度、支持的特征和/或代码的抽象解释。

[0201] 在一个方面中,作为方框736的一部分,沙盒 JavaScript® 引擎可以限制从脚本的推测性执行生成的数据结构(例如,哈希表、阵列等)的大小。完整的数据结构不会使得识别用于下载的其它资源,所以用于完全生成/构成大数据结构所需的处理时间可以被绕过。

[0202] 在方框738中,沙盒 JavaScript® 引擎可以发现为了呈现HTML文档所需要的但在HTML文档中没有明确请求的资源。在方框740中,沙盒 JavaScript® 引擎可以告知(或者产生)预取器取回所发现的资源。在方框742中,沙盒 JavaScript® 引擎可以丢弃方框736中执行的处理的结果。

[0203] 在方框744中,预取器可以对在方框738中沙盒 JavaScript® 引擎发现的资源进行

定位。在方框746中,预取器可以下载所定位的资源。在方框748中,预取器可以将下载的资源保存到存储器。

[0204] 如上面所讨论的,HTML代码可以嵌入 JavaScript® 代码(其被称为“内联脚本”)并且包括对 JavaScript® 代码的链接(其被称为“外部脚本”)二者。为了正确地处理HTML文档,必须按照HTML标准定义的特定顺序来执行内联脚本和外部脚本二者。

[0205] 当多个脚本被并行地下载、解析、分析以及编译时,脚本变成准备好执行的顺序可能不同于HTML标准定义的特定执行顺序。如果脚本未准备好执行,但是按HTML标准定义的特定执行顺序是下一脚本,则可以在执行HTML文档的任何另外的处理之前要求浏览器等待,直到该脚本变成准备好执行。各个方面使用该等待时间来准备用于执行的其它脚本或资源(这未被HTML标准管理)。可以并行地和/或在其它脚本的执行期间准备多个脚本和资源。

[0206] 另外,并非HTML文档中包括的(即,嵌入或链接到的)全部脚本都被实际执行,但是提前准备用于执行的全部脚本可能浪费功率和处理资源。各个方面可以智能地选择要准备的用于执行的脚本。

[0207] 作为例子,HTML预取器可以发现并且(无序地)下载全部引用的脚本,并且HTML解析器可以稍后以正确顺序且在处理HTML文档的时间中的正确点处安排其执行。

[0208] 通常必须维持脚本的最终执行顺序。然而,可以并行地和/或无序地执行与下载、解析、分析和编译脚本相关联的全部操作。

[0209] 在一个方面中,可以准备HTML文档中包括的脚本以便并行地(即,相对于彼此)和/或无序地(即,相对于HTML标准定义的特定执行顺序)执行。这可以通过生成和/或将唯一标识符和/或签名与每个脚本相关联来实现。签名可以是基于脚本的内容。适合用于各个方面的签名和签名过程的例子包括:文件偏移(用于内联脚本)、消息摘要算法(例如,MD5)、安全哈希算法(SHA)、脚本的URL、脚本的URI、浏览器高速缓存密钥和/或各种已知的签名过程中的任何一种。

[0210] 图7C示出了智能地准备HTML文档中包括的脚本以并行执行的示例性浏览器方法750。方法750的操作可以由处理器与其它浏览器操作并行执行。

[0211] 在方框752中,HTML扫描器/预取器可以扫描HTML文档以获取结构信息和/或发现资源(图像、CSS、脚本等)。在方框754中,HTML扫描器/预取器可以发现HTML文档中的一个或多个脚本,并且告知HTML解析器(其与HTML扫描器并行执行)所发现的脚本。在方框756中,HTML扫描器/预取器可以发起外部脚本的下载。

[0212] 在方框758中,HTML解析器可以为每个发现的脚本(内联脚本和外部脚本二者)生成标识符(或签名)和/或将每个发现的脚本与标识符相关联。在一个方面中,HTML解析器可以将发现的脚本的文本设置为其标识符。在一个方面中,HTML解析器可以将外部脚本的URL/URI与外部脚本相关联(即,可以将它们的URL/URI设置为它们的签名),并且执行摘要和/或哈希算法以计算内联脚本的签名。如果脚本的URL/URI不可用、不唯一和/或不能唯一标识脚本,则作为方框758的一部分,HTML解析器可以生成并且使用签名来标识该脚本。

[0213] 在方框760中,HTML解析器可以向 JavaScript® 引擎发送脚本和其关联的标识符或URL/URI,该 JavaScript® 引擎与HTML解析器并行执行(例如,在单独的线程中)。在方框

762中,HTML解析器可以执行各种HTML解析器操作,例如解析HTML以发现其它脚本。

[0214] 在方框772中,JavaScript®引擎可以从HTML解析器接收脚本和相关联的标识符、签名或URL/URI。在方框774中,JavaScript®引擎可以准备(例如,解析、分析和/或编译)所接收的脚本以便执行。可以跨越全部接受的脚本无序地和/或并行地执行准备操作(即,可以一次准备多个脚本)。在一个方面中,作为方框774的一部分,JavaScript®引擎可以采用启发法(例如,通过抽象解释)来检测调用图而不执行代码,基于调用图来识别最可能被执行的脚本(或功能),以及为仅执行被确定为可能要被执行的脚本而进行准备。在方框776中,JavaScript®引擎可以将脚本准备期间生成的信息(例如,经编译的代码等)与该脚本标识符、签名或URL/URI相关联。

[0215] 在方框764中,HTML解析器可以识别要被执行的下一脚本(例如,基于HTML标准定义的执行顺序)。在方框766中,HTML解析器可以向JavaScript®引擎发送要被执行的下一脚本的标识符、签名或URL/URI。在方框768中,HTML解析器可以等待执行的结果或者脚本已经被执行的通知。在方框770中,HTML解析器可以继续执行HTML解析器操作。

[0216] 在方框778中,JavaScript®引擎可以从HTML解析器接收标识符、签名或URL/URI。在方框780中,JavaScript®引擎可以基于所接收的标识符、签名或URL/URI来识别合适的脚本。在确定框782中,JavaScript®引擎可以通过例如确定针对所识别的脚本的解析、分析和编译操作中的全部均已被执行来确定该脚本是否准备好立即执行。如果JavaScript®引擎确定脚本准备好立即执行(即,确定框782=“是”),则在方框786中,JavaScript®引擎可以告知HTML解析器执行的结果或者执行完成。

[0217] 当确定脚本还未准备好立即执行时(即,确定框782=“否”)时,则在方框784中,JavaScript®引擎可以使用传统解决方案来准备脚本以便执行。在方框786中,JavaScript®引擎可以根据HTML标准定义的特定执行顺序来执行脚本。以这种方式,方法750准备HTML文档中包括的脚本以用便并行地(即,相对于彼此)和无序地(即,相对于HTML标准定义的特定执行顺序)执行,并且以标准定义的顺序来执行脚本。

[0218] 图8示出了处理已预取资源的示例性浏览器方法800。在方框802中,网络浏览器组件(例如,经由提取管理器502)可以发起发现的资源(例如,图像)的下载,所述发现的资源可以与其它浏览器操作(例如,HTML解析等)的执行并发地(或并行地)被下载/被提取。当与发现的资源相关联的全部数据被下载和/或接收时,在方框804中,所下载的数据(例如,图像数据)可以被发送给线程池以便解码。在一个方面中,解码操作可以与其它浏览器操作并行地被执行。

[0219] 在方框806中,所下载的数据(例如,图像数据)可以被解码。在方框808中,可以将解码后的数据添加到DOM分配器队列。在方框810中,DOM分配器组件504可以使对DOM树和相应的树节点(例如,在图像数据的情况下“img”树节点)的更新串行化。在方框812中,可以从处理列表(例如,未处理的图像的列表)移除资源(例如,图像)。

[0220] 图9示出了适合于与各个方面结合使用的CSS引擎512中的示例性组件。CSS引擎512可以被配置为执行三种主要类型的操作:CSS资源预取操作902、CSS解析操作904和DOM

样式化操作906。

[0221] CSS解析操作904可以包括读取CSS节点并且在存储器中创建数据结构的集合(例如,CSS规则)。CSS节点可以被嵌入HTML中或者作为单独的文件被链接,并且可以被存储在不同的服务器上。传统CSS引擎(例如,WebKit或Firefox中的一个)可以在主浏览器线程中顺序解析CSS。因此,如果页面使用嵌入式CSS,则HTML解析不能解析HTML文档中的剩余部分,直到CSS引擎已经解析文档报头中的样式元素。如果页面使用若干个CSS文件,则其将都被顺序解析,即使可能存在未充分使用的CPU核心。如果站点使用大CSS文档,则这样的CSS解析串行化(即,CSS文档的串行处理)可能导致严重减速。各个方面可以使用异步任务以避免CSS解析串行化。

[0222] 参照图9,HTML解析器506可以被配置为在页面加载操作期间为DOM树中的每个样式元素产生CSS解析570任务。类似地,每当新CSS文件到达时,提取管理器502可以产生CSS解析570任务。因此,多个CSS解析570任务可以与HTML解析器506和/或HTML解析操作568并发地执行。

[0223] 因为样式表(CSS)和规则(CSS规则)的全序可能是样式化操作574的关键部分,所以浏览器系统500可以被配置为确保全序是相同的,如同全部样式表(CSS)已经按照程序员想要的顺序被解析了。

[0224] 在各个方面中,解析任务或解析操作568、570中的每一个都可以接收唯一的、顺序的解析器ID。浏览器系统500随后可以使用该ID来重新创建文档中的样式表的顺序。

[0225] DOM格式操作906可以使得CSS引擎512能够使用CSS解析器522创建的数据结构来确定DOM树中的节点的样式。对于每个节点,CSS引擎512可以执行规则匹配操作来找到其选择器匹配该节点的全部规则。对于每个节点,规则匹配通常返回许多(有时冲突的)规则。使用级联,CSS引擎可以向规则指派权重并且选择具有最大权重的规则。

[0226] 样式化节点中的最后一步可以包括DOM样式化操作906,其通过使用通过级联算法选择的规则来创建样式数据结构并且将其附加到DOM。只要实施特定的相关性,就可以在若干个节点上并行地执行规则匹配和级联操作。

[0227] 各个方面可以在多个浏览器操作和/或传递的并发执行(或交迭)期间遵守/实施现有的HTML和JavaScript®语义。DOM树可以是全部浏览器传递所使用的主数据结构。在各个方面中,可以使对DOM树(其可以由HTML5解析器构建)的访问串行化以符合HTML5规范。另外,为了允许更大的并行性,每个传递可以访问私有的并发数据结构(即,除了DOM树之外)。在一个方面中,这种另外的数据结构可以是布局树。

[0228] 图10示出了示例性并行DOM样式化方法1000,其中,在若干个节点上并行执行规则匹配和级联操作。在方框1002中,CSS引擎512可以遍历DOM树并且对于每个DOM节点产生两个不同任务:匹配任务和节点样式化任务。在方框1004中,匹配任务可以执行针对DOM节点的规则匹配和级联操作。在方框1006中,样式化任务可以创建描述DOM节点的样式数据结构。在方框1008中,样式化任务可以将样式数据结构附加到DOM树。

[0229] 图11A示出了适合用于各个方面的示例性DOM树。图11B示出了对应于图11A中示出的示例性DOM树的示例性任务有向无环图(DAG)。具体地,图11B示出了匹配任务(其被表示为三角形)如何可以完全独立于彼此和样式化任务(其被表示为矩形),而样式化任务取决于彼此和匹配任务。通常,匹配任务的并行执行仅受到计算系统中的处理核心的数量的限

制。

[0230] 如上面所提及的,样式化任务可以取决于彼此和/或匹配任务。可以要求每个样式化任务在其能够执行前满足两种相关性。第一,样式化任务只可以在作用于相同节点的匹配任务完成执行之后执行。这是因为样式化任务使用由匹配任务选择的规则来构建样式数据结构。第二,作用于节点的样式化任务只可以在作用于父节点的样式化任务完成执行之后执行。这是因为节点的样式属性中的一些属性可以从其父节点继承。例如,CSS代码p {color: inherit} 指示浏览器使用与其父节点相同的前景颜色来呈现

节点。

[0231] 就计算、功率、延迟等而言,由匹配任务执行的规则匹配操作可能是高代价的。例如,如果CSS引擎512需要确定规则“h1 p div {color:red}”是否应用于

元素E,则匹配算法可能需要找到E的先辈中是否有

元素,以及

的先辈中是否有

# 元素。这可能要求沿着DOM树的全部道路走到根部,这可能是高代价的操作。另外,典型网站可能需要多于400000个的、这样的DOM树行走。

[0232] 为了减少DOM树行走的数量,各个方面可以包括布隆过滤器,其存储与DOM节点的先辈有关的信息。布隆过滤器可以减少90%的到根部(A)的DOM树行走的数量,使样式化算法中消耗的时间减少一半。

[0233] 布隆过滤器可以是大的数据结构,并且CSS引擎512可以被要求针对每个样式化任务对其进行复制。由于复制成本可能远高于性能获益,因此各个方面可以使用比布隆过滤器更小的结构。这可以通过减少复制操作的数量和/或减少被复制的元素的大小来改进浏览器的性能。

[0234] 如上所述,各个方面可以使用元素id和类别属性来预测CSS文件中引用的图像是否应当被预取。在一个方面中,这些元素和属性可以被存储在数据库中,该数据库记录了他们中的每一个在文档中出现的次数。HTML解析器还可以向这个数据库添加信息。

[0235] 在规则匹配算法开始之前,CSS引擎512可以根据数据库中的项目的频率对其进行排序。浏览器系统500随后可以向位图数据结构(其被称为“匹配的位图”)中的每个项目指派比特。如果id和类别的数量大于位图大小,则可以向多个项目指派单个比特。由于这些位图较小,因此其可以被复制很多次而不会显著地影响计算设备的性能。

[0236] 在规则匹配操作期间,每个样式化任务可以从其父辈接收匹配的位图。匹配的位图可以记录其先辈的id、类别和标签。样式化任务可以使用匹配的位图来过滤出从未匹配的规则。之后,样式化任务可以向其添加它们的节点的id、类别和标签,并且向它们的后代发送副本。平均而言,这样的匹配的位图避免了到DOM树的根部的行走的90%,仅具有0.024%的假阳性。

[0237] 假阳性可能发生,这是因为匹配的位图不记录遇到标记和id的顺序。例如,为了确定规则“h2 h1 p {color:red}”是否应用于某节点

,且匹配的位图指示

# 和二者都是 的先辈,可能要求浏览器系统500走遍DOM树以检验是否是的先辈。如果不是这种情况,则是假阳性的情形。这样的假阳性可能不会导致页面错误地呈现,但是可能会浪费CPU周期。

[0238] 在一个方面中,诸如由呈现引擎子系统560执行的布局和呈现操作可以包括执行以下计算:将经样式的DOM转换成位图图像以在屏幕上显示。应用于位图图像的DOM和CSS样式可以被组合以形成新的树结构(其被称为布局树),其中,每个节点代表网页上的可视

35

元素。每个DOM节点可以被翻译为零、一或许多布局树节点。呈现引擎子系统560可以将布局树作为输入并且计算每个元素占据的页面的区域。每个元素的样式可以被视为对布局的约束(例如,内联/块显示、浮点、宽度、高度等)。

[0239] 呈现引擎子系统560可以遍历布局树并且解决约束(例如,作为布局操作582的一部分)以确定每个元素的最终宽度、高度和位置。呈现引擎子系统560还可以走遍(例如,作为呈现操作584的一部分)布局树(其可以利用布局引擎计算的结果来注释)并且根据CSS的规则在屏幕上将其画出。

[0240] 由于布局操作582和呈现操作紧密相关并且以流水线方式一起操作,因此在一个方面中,可以由诸如布局和呈现引擎516之类的单个组件来执行它们。

[0241] 在各个方面中,呈现引擎子系统560可以被配置为执行布局操作582,从而在布局树上的4类传递中执行CSS布局算法。在每类传递中,与传统方法相比,信息可以以更受控的方式流过该树,发掘出布局过程中的并行性的潜力。

[0242] 在一个方面中,呈现引擎子系统560可以在布局树上执行4类传递:最小的或优选的宽度计算传递、宽度计算传递、块级格式化上下文(block formatting context)流传递和绝对位置计算。

[0243] 第一传递(即,最小的或优选的宽度计算传递)可以是自下而上的传递,该传递将宽度沿着树传播,以向每个元素指派最小宽度和优选宽度。作为例子,对于包含文本的段落的div元素,最小宽度可以是当在每个字之后放置换行符时的宽度,而优选宽度可以是没有换行符的宽度。

[0244] 第二传递(即,宽度计算传递)可以是自上而下的传递,该传递计算每个元素的最终宽度。取决于元素的样式,最终宽度可以从其父宽度或者最小/优选宽度得到。

[0245] 在第三传递(即,块级格式化上下文流传递)期间,每个元素具有已知的宽度,并且其内容可以用于计算其高度。作为例子,对于包含文本的段落的div元素,在宽度被确定之后,可以将文本放置在其内部,并且可以对每个线条的高度求和以得到该div的总高度。传播的方向可能是复杂的。其内容用于计算该div的高度的元素可以被称为块级格式化上下文(BFC)。元素是否是块级格式化上下文可以由其CSS样式来确定。

[0246] DOM树中的块级格式化上下文元素可以形成逻辑树,该逻辑树可以被覆盖在DOM上。可以自下而上地走块级格式化上下文覆盖树,并且到浏览器系统到达DOM树的根部的时候,其将已经布满整个网页。在这个阶段的结束时,浏览器系统500将被告知全部元素的高度,以及它们在包含它们的块级格式化上下文内的相对位置。

[0247] 第四传递(即,绝对位置计算传递)可以是自上而下的传递,该传递使用来自先前传递的、在每个块级格式化上下文内的相对位置来计算页面上的每个元素的绝对位置。

[0248] 在一个方面中,可以通过走布局树来实现呈现,从而在前景元素之前访问背景元素。各个方面可以与其样式一致的方式将每个元素画入图形缓冲器中,并且在屏幕上(例如,经由GUI)显示缓冲器的内容。由于组合步骤使用的存储器带宽,这些呈现操作在计算上可能是高代价的。各个方面可以被配置为通过各种组件/子系统的并行或并发执行来减少每个组合步骤需要的存储器带宽。

[0249] 通常,布局和呈现操作的性能是重要的,这是由于它们对从页面加载时间到用户接口的响应能力的所有方面的影响。另外,布局和呈现操作与比如执行JavaScript®之类

的其它重要任务竞争CPU周期。

[0250] 连同顺序优化,各个方面可以包括粗糙和精细粒度的并行性二者以改进布局和呈现引擎的性能。这两种方法可以是互补的。在粗糙级别,示例性浏览器可以将尽可能多的工作从关键路径移出并且移入工作线程。在精细级别,示例性浏览器可以并行化布局和呈现算法/方法。

[0251] 在传统网络浏览器中,操作DOM的任务(例如,解析或JavaScript®)从不与布局和呈现任务同时执行,这确保二者彼此不干扰。相反地,各个方面交迭这两个类型的任务。因此,在各个方面中,每次DOM改变时,可以不更新布局树。

[0252] 各个方面可以分离(或保持分离)布局树和DOM。对布局树的更新可以作为分批操作在布局和呈现操作将正常发生时被执行,这往往是在解析或JavaScript®执行任务完成之后。以这种方式对更新进行分组可能意味着要求浏览器系统500维护额外的状态信息以识别已经改变的DOM的部分,但是能避免执行不必要的工作,这是因为对于DOM的每个中间状态,没有更新布局树。

[0253] 当布局树准备好利用结果做有用的工作时,各个方面可以对其进行更新。布局树可以是与DOM分离的实体。可以进行全部DOM改变而不影响布局树。相反地,一旦布局树被更新,呈现引擎子系统560不需要以任何方式访问DOM。这实现了并行性,并且还意味着布局树必须复制传统地仅被存储在DOM中的某些信息。特别地,布局树可以包含对文本、图像、CSS样式和HTML canvas元素的直接引用。

[0254] 文本和图像可以是不可变的,并且安全地与DOM共享。CSS样式逻辑上可以是不可变的,但是CSS样式对象中的数据量可能太大(和/或其可能被更新得太频繁)以至于不能每次复制整个对象。因此,在一个方面中,每个样式对象可以在内部被划分为许多较小的子样式对象。可以使用写入时复制的方法来更新共享的子样式。未共享的子样式可以被就地更新。因此,复制样式对象可能仅需要创建共享相同子样式的新的样式对象,这可能是代价较低的。另外,可以对子样式进行分组,从而被一起更新的CSS属性具有相同的子样式,这可以在更新发生时使子样式副本最小化。这种安排使得DOM、布局和呈现组件能够引用相同CSS类型而不会使得在一个位置/组件中所做的改变对其它位置/组件可见。类似的写入时复制的方法可以用于HTML canvas元素。

[0255] 布局树与DOM树的分离使得能够在呈现引擎子系统560中实现粗糙粒度的并行性。当网页准备好对用户第一次显示时,浏览器系统500可以创建工作项目,该工作项目初始化布局树并且将其传递给呈现引擎子系统560以便处理。将布局和呈现操作分离到不同线程使得浏览器系统500的剩余部分能够向前进行,例如JavaScript®可以被执行,用户接口(UI)事件可以被处理,以及CSS样式化可以被计算等。

[0256] 当呈现引擎子系统560完成其任务并且在屏幕上显示页面时,其可以提交“LR工作项目”以更新布局树,并且再次开始全部过程。仅“LR工作项目”需要排它地访问DOM,而一旦树被更新,则其它操作可以被并行地和/或异步地执行。

[0257] 某些JavaScript®DOM API(例如,getStyle和offsetTop)可能需要与布局算法计算的结果有关的信息。呈现引擎子系统560可能被要求暂停直到结果可用。如果呈现引擎子系统560在主线程中执行布局,则其可能复制在LR工作项目(或LR线程)中执行的

计算,这可能浪费时间和能量。

[0258] 在一个方面中,呈现引擎子系统560可以被配置为记住布局树是否有最新布局信息。如果是,则可以立即返回同步布局请求。如果不是,则在LR线程中正常地执行布局操作,并且可以请求呈现引擎子系统560通知主线程何时完成布局过程。当妨碍复制工作时,这尽快传送了所需结果。

[0259] 除了并行性之外,分离布局树和DOM的另一优点是可以将呈现引擎子系统560当做网页之间共享的服务。由于布局树并不返回参考利用其来构建布局树的DOM,因此相同的呈现引擎子系统560可以管理全部布局树而不管它们的源。这意味着高代价的、与有限呈现相关的资源(比如图形缓冲器)在整个浏览器系统500仅需要一个实例。

[0260] 布局树提供的另一优点是,当用户与快速改变的页面交互时,在确定用户意图方面具有添加的灵活性。例如,如果用户点击屏幕上通过JavaScript®移动的按钮,JavaScript®改变DOM和结果出现在屏幕上之间存在延时,这是因为布局和呈现操作花费时间。到用户的点击被注册的时候,可能已经更新DOM,并且从浏览器的角度来看,框的位置可能已经改变。尽管用户的鼠标指针完全地在框上,但对点击的尝试可能不成功。然而,因为布局树与DOM分离,所以浏览器系统500可以访问当前的工作树和在屏幕上显示的最后的树。这使得浏览器系统500可以基于当用户点击时所看到的来确定用户意图点击的对象,而不是基于DOM的当前状态,故得到改进的可察觉的响应能力和更好的用户体验。

[0261] 可以在各种移动计算设备上实现各个方面,图12中示出了其一个例子。具体地,图12是适合于与任何方面结合使用的智能电话/蜂窝电话1200的形式的移动收发机设备的系统框图。蜂窝电话1200可以包括耦合到内部存储器1202的处理器1201、显示器1203和扬声器1208。另外,蜂窝电话1200可以包括用于发送和接收电磁辐射的天线1204,所述天线1204可以被连接到无线数据链路、蜂窝电话收发机和/或耦合到处理器1201的无线收发机1205。蜂窝电话1200通常还包括用于接收用户输入的菜单选择按钮或摇杆开关1206。

[0262] 典型的蜂窝电话1200还包括声音编码/解码(CODEC)电路1213,其将从麦克风接收的声音数字化为适合无线传输的数据分组,以及解码所接收的声音数据分组以生成提供给扬声器1208的模拟信号以生成声音。另外,处理器1201、无线收发机1205和CODEC电路1213中的一个或多个可以包括数字信号处理器(DSP)电路(未单独示出)。蜂窝电话1200还可以包括ZigBee®收发机(即,IEEE 802.15.4收发机)1213,其用于无线设备或其它类似通信电路(例如,实现蓝牙®或WiFi协议的电路等)之间的低功率短距离通信。

[0263] 可以在各种商业可用的服务器设备中的任何一种(例如,图13中示出的服务器1300)上实现各个方面。这样的服务器1300通常包括耦合到易失性存储器1302和大容量非易失性存储器(例如,磁盘驱动器1303)的处理器1301。服务器1300还可以包括耦合到处理器1301的软盘驱动器、压缩盘(CD)或DVD盘驱动器1311。服务器1300还可以包括耦合到处理器1301的网络接入端口1306,其用于建立与网络1305的数据连接,网络1305例如耦合到其它通信系统计算机和服务器的局域网。

[0264] 计算设备的其它形式也可以从各个方面受益。这样的计算设备通常包括图14中示出的组件,图14示出了一种示例性个人膝上型计算机1400。这样的个人计算机1400通常包括耦合到易失性存储器1402和大容量非易失性存储器(例如,磁盘驱动器1403)的处理器

1401。计算机1400还可以包括耦合到处理器1401的压缩盘(CD)和/或DVD驱动器1404。计算设备1400还可以包括耦合到处理器1401的一些连接器端口，其用于建立数据连接或接收外部存储器设备，例如用于将处理器1401耦合到网络的网络连接电路1405。如计算机领域所公知的，计算机1400还可以耦合到键盘1408、诸如鼠标1410之类的定点设备和显示器1409。

[0265] 处理器1201、1301、1401可以是任何可编程的微处理器、微计算机或者多处理器芯片或芯片集，它们可以由软件指令(应用)来配置以执行各种功能，包括上面所描述的各个方面功能。在一些移动设备中，可以提供多个处理器1301，例如，专用于无线通信功能的一个处理器和专用于运行其它应用的一个处理器。通常，软件应用在被访问并被加载到处理器1201、1301、1401之前，可以被存储在内部存储器1202、1302、1303、1402中。处理器1201、1301、1401可以包括足够用于存储应用软件指令的内部存储器。

[0266] 提供前述的方法描述和过程流程图仅仅作为说明性的例子，而并非旨在要求或暗示必须按照给出的顺序执行各个方面的步骤。本领域的技术人员应当意识到，可以按照任何顺序执行前述方面中方框的顺序。诸如“随后”、“然后”、“接下来”等词汇并非旨在限制方框的顺序；这些词汇仅用于引导读者阅读对方法的描述。此外，以单数形式(例如使用冠词“一个”、“一”或“该”)对权利要求元素进行的任何引用都不应被解释为将元素限制为单数。

[0267] 结合本文公开的方面描述的各种说明性逻辑框、模块、电路和算法步骤可以被实现为电子硬件、计算机软件或二者的组合。为了清楚地说明硬件和软件的这种可交换性，以上各种说明性组件、方框、模块、电路和方框均围绕它们的功能来概括性描述。这样的功能被实现为硬件还是软件取决于具体应用和施加在整个系统上的设计约束。技术人员可以针对各个具体应用以变通方式来实现所描述的功能，但是这种实现决策不应当被解释为使得脱离本发明的范围。

[0268] 可以利用被设计用于执行本文描述的功能的通用处理器、数字信号处理器(DSP)、专用集成电路(ASIC)、现场可编程门阵列(FPGA)或其它可编程逻辑器件、分立门或晶体管逻辑器件、分立硬件组件或其任意组合来实现或执行用于实现结合本文公开的方面描述的各种说明性逻辑单元、逻辑框、模块和电路的硬件。通用处理器可以是微处理器，或者，处理器可以是任何传统的处理器、控制器、微控制器或状态机。处理器还可以被实现为计算设备的组合，例如DSP和微处理器的组合、多个微处理器、一个或多个微处理器与DSP内核的结合或者任何其它此种结构。或者，一些模块或方法可以由特定于给定功能的电路来执行。

[0269] 在一个或多个示例性方面中，可以用硬件、软件、固件或其任意组合来实现描述的功能。如果用软件实现，则这些功能可以作为一个或多个指令或代码被存储在非暂时性计算机可读介质或非暂时性处理器可读介质上。本文公开的方法或算法的步骤可以体现在处理器可执行的软件模块中，其可以位于非暂时性计算机可读或处理器可读存储介质上。非暂时性计算机可读或处理器可读存储介质可以是可以由计算机或处理器存取的任何存储介质。作为例子而非限制，这样的非暂时性计算机可读或处理器可读介质可以包括RAM、ROM、EEPROM、FLASH存储器、CD-ROM或其它光盘存储、磁盘存储或其它磁盘存储设备、或可以用于存储具有指令或数据结构形式的期望的程序代码且可以被计算机存取的任何其它介质。如本文使用的，磁盘或光盘包括压缩盘(CD)、激光盘、光盘、数字多功能盘(DVD)、软盘和蓝光光盘，其中磁盘通常磁性地复制数据，而光盘则用激光来光学地复制数据。上面的组合也包括在非暂时性计算机可读和处理器可读介质的范围内。另外，方法或算法的操作可以

作为代码和/或指令中的一项或任意组合或集合位于非暂时性处理器可读介质和/或计算机可读介质上，后者可以被并入计算机程序产品中。

[0270] 提供前面对公开方面的描述以使本领域任何技术人员能够实施或使用本发明。对本领域技术人员而言，对这些方面的各种修改将是显而易见的，并且可以将本文所定义的一般性原理应用于其它方面，而不脱离本发明的精神或范围。因此，本发明并不旨在要受限于本文示出的方面，而是要符合与所附权利要求以及本文所公开的原理和新颖性特征相一致的最广泛的范围。

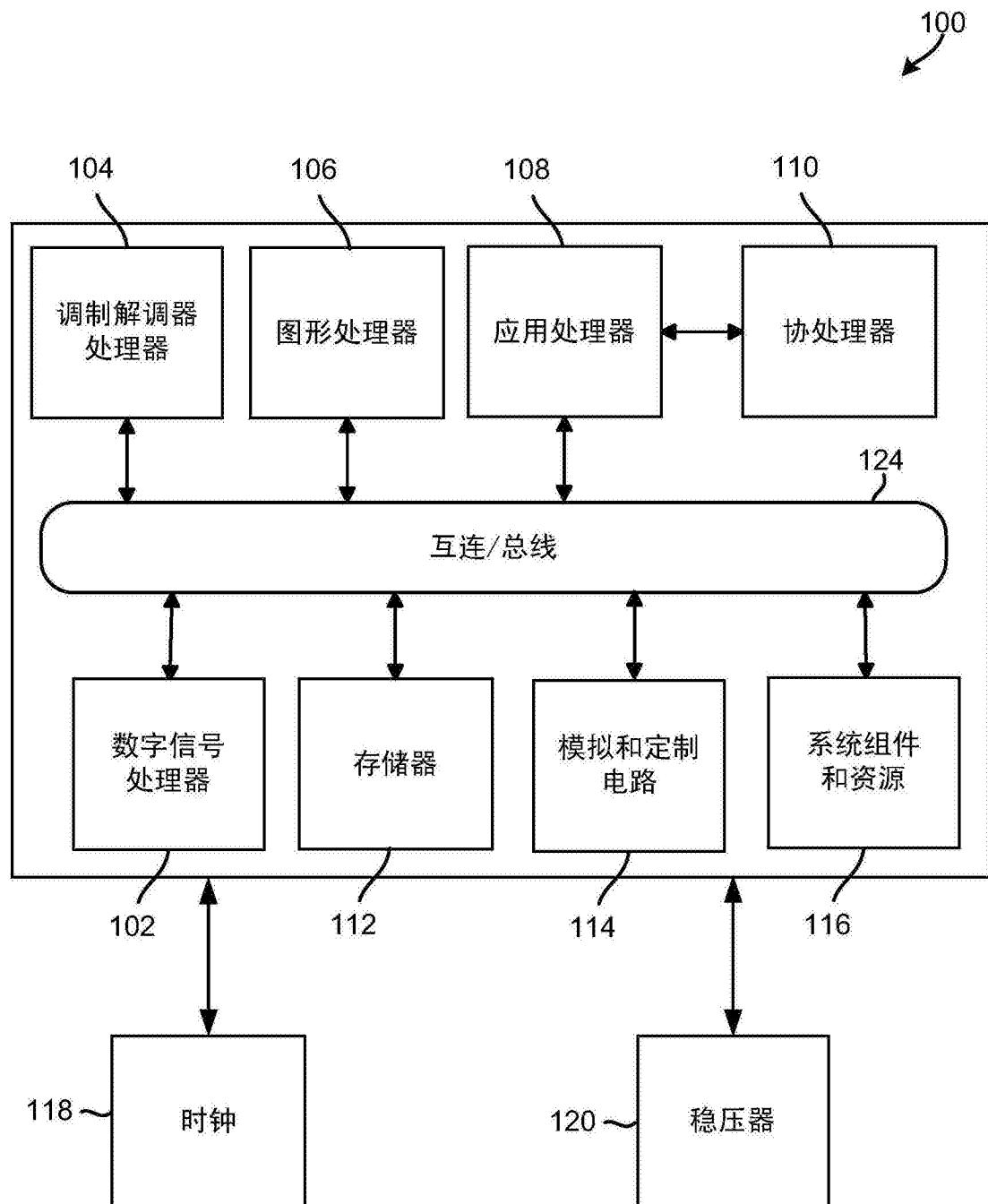


图1

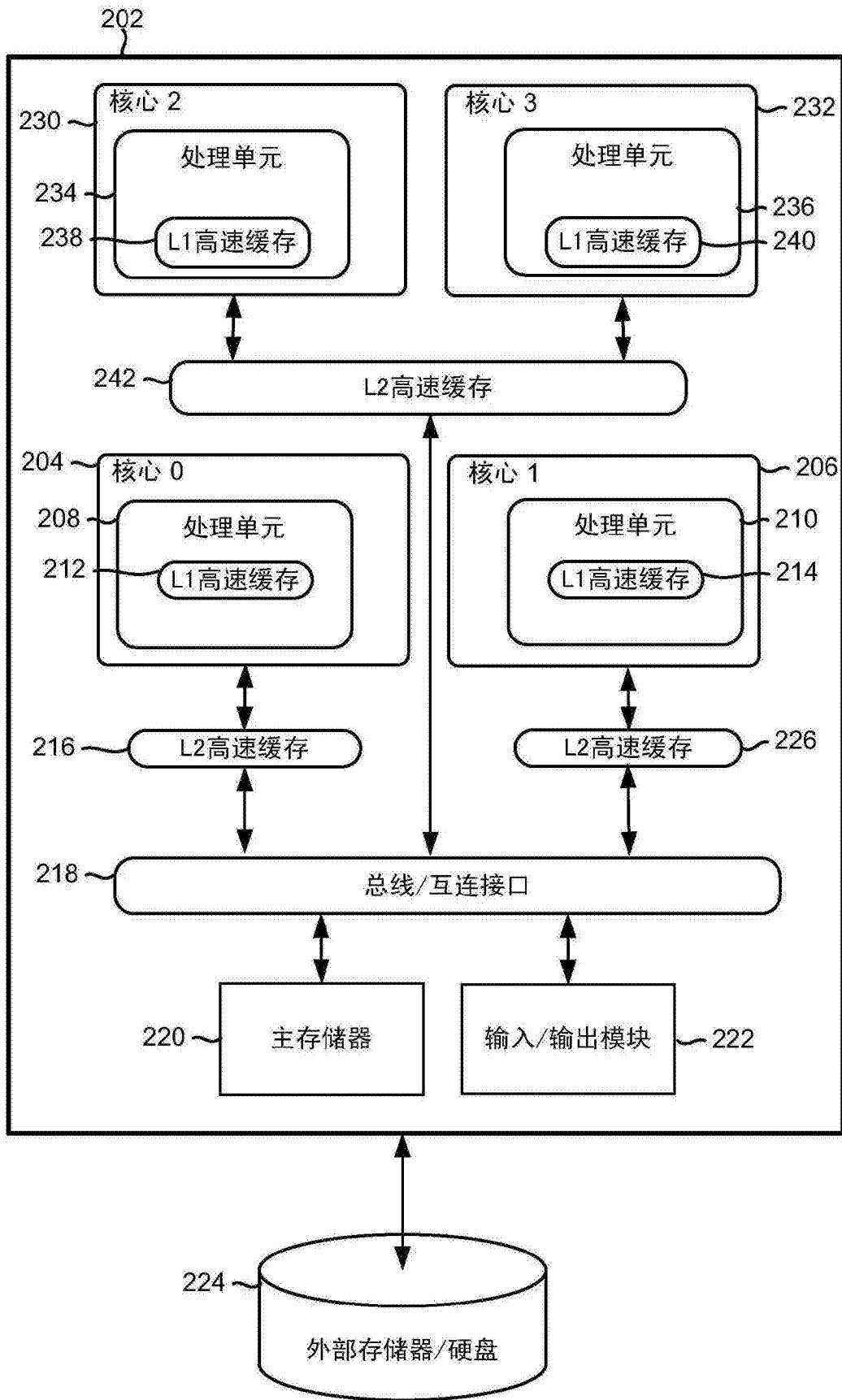


图2

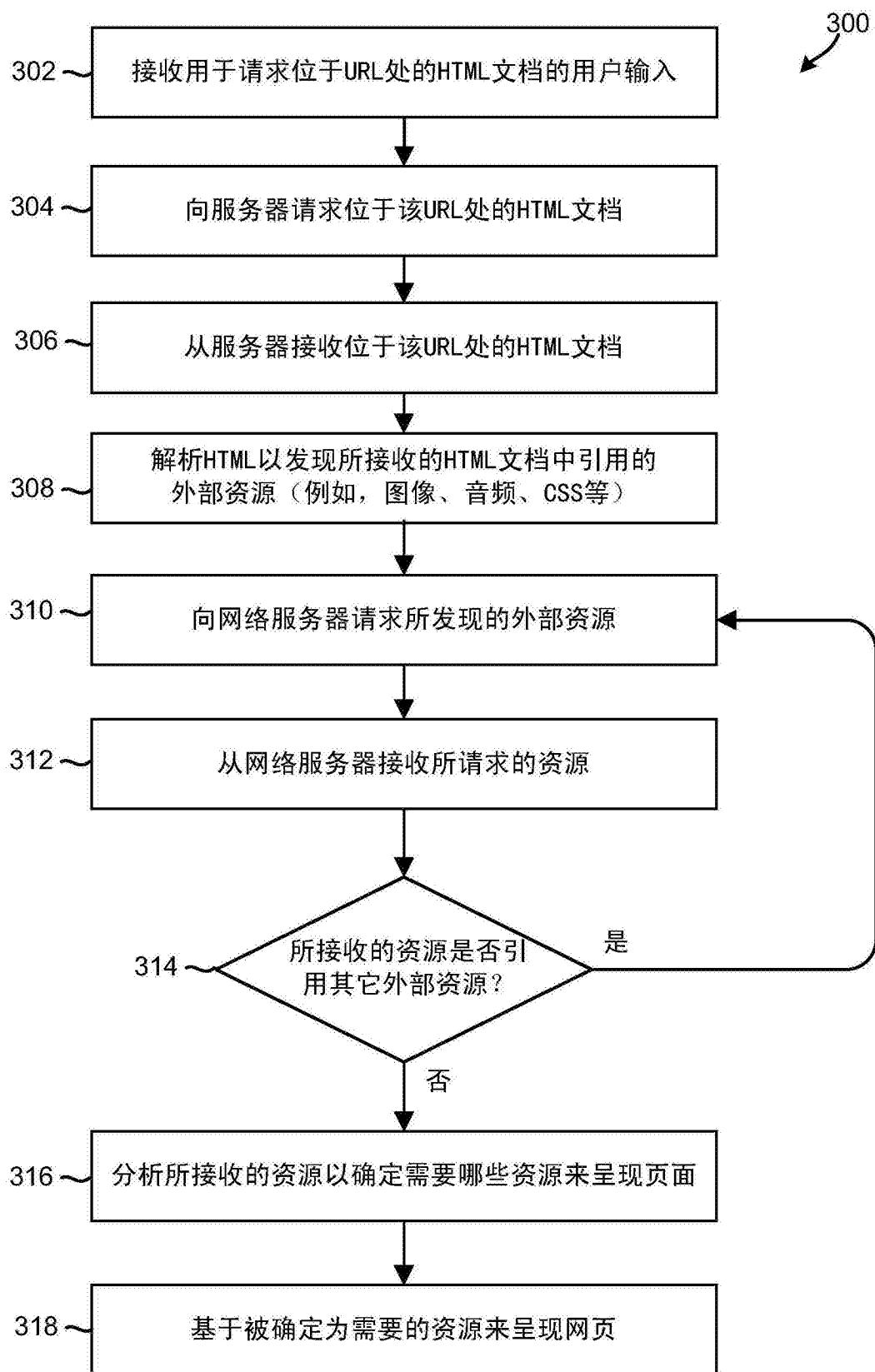


图3A

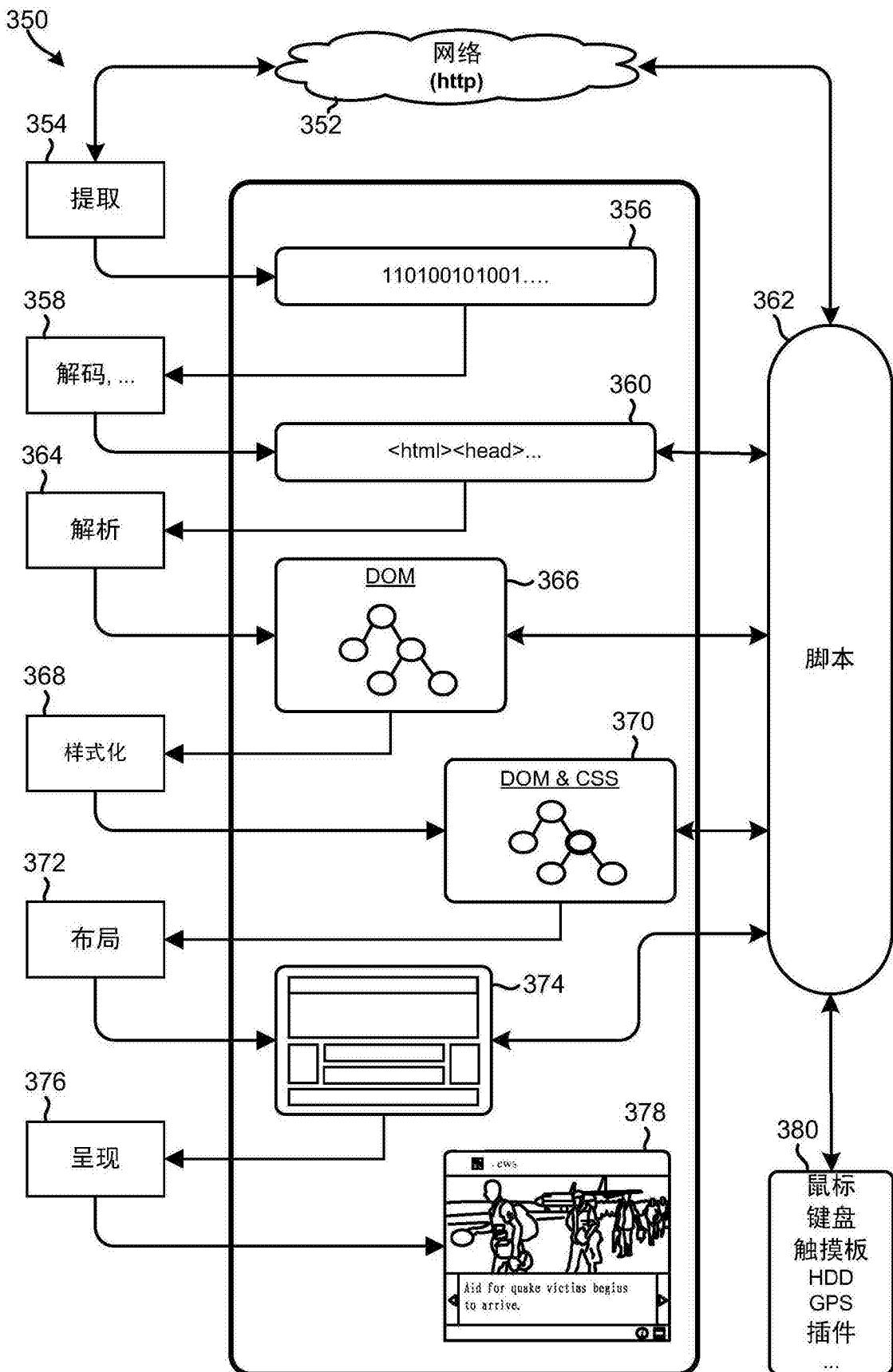


图3B

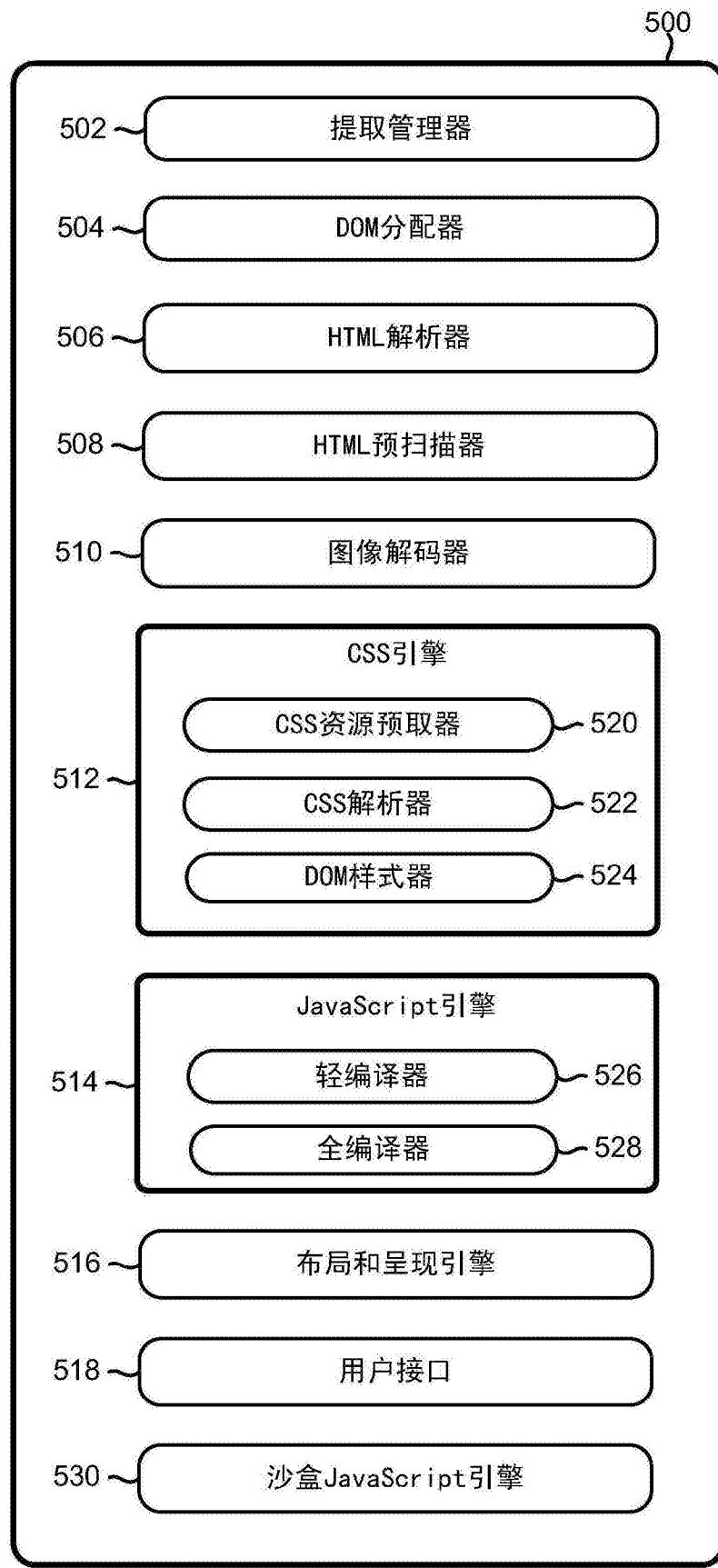


图4

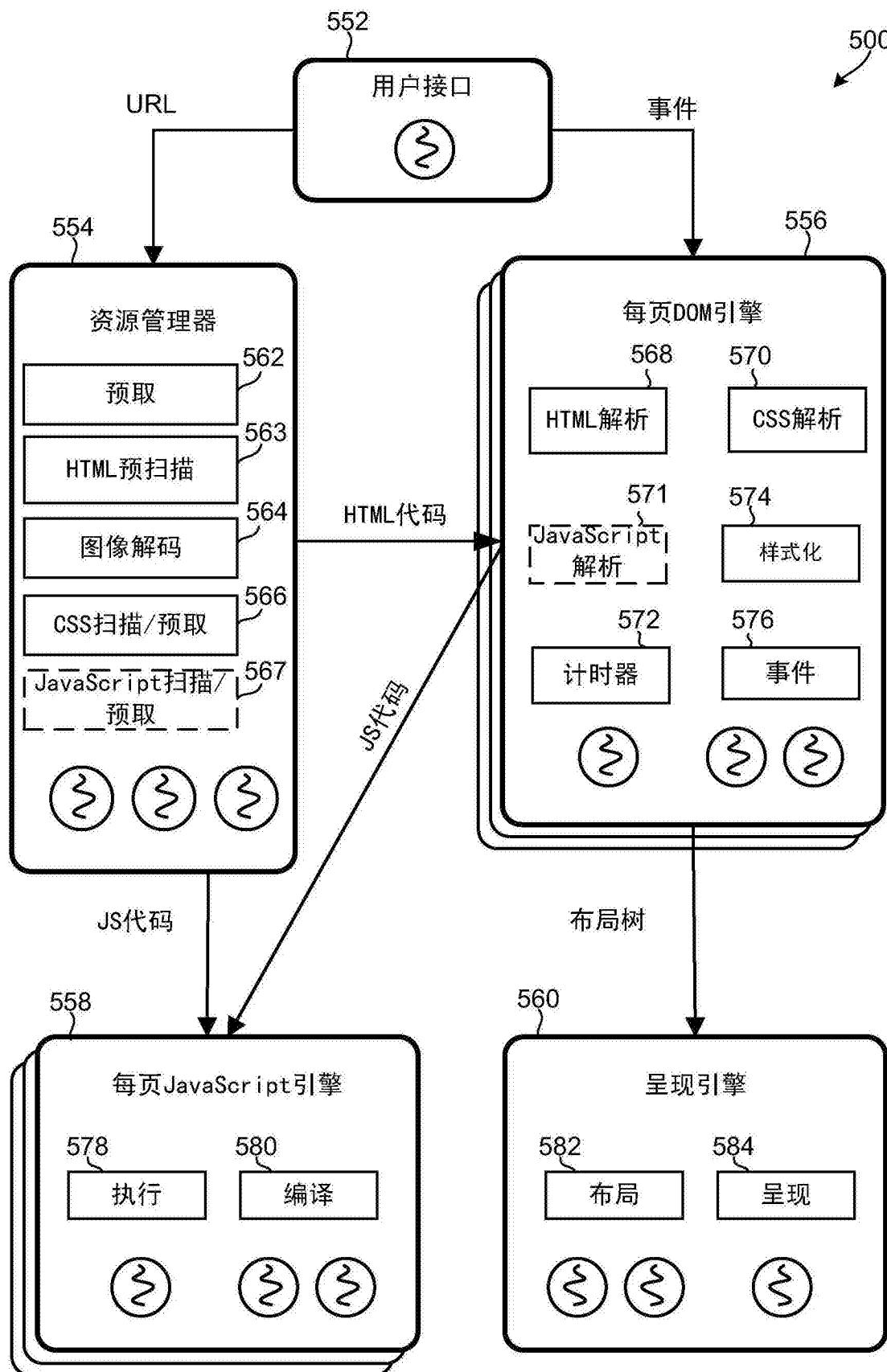


图5

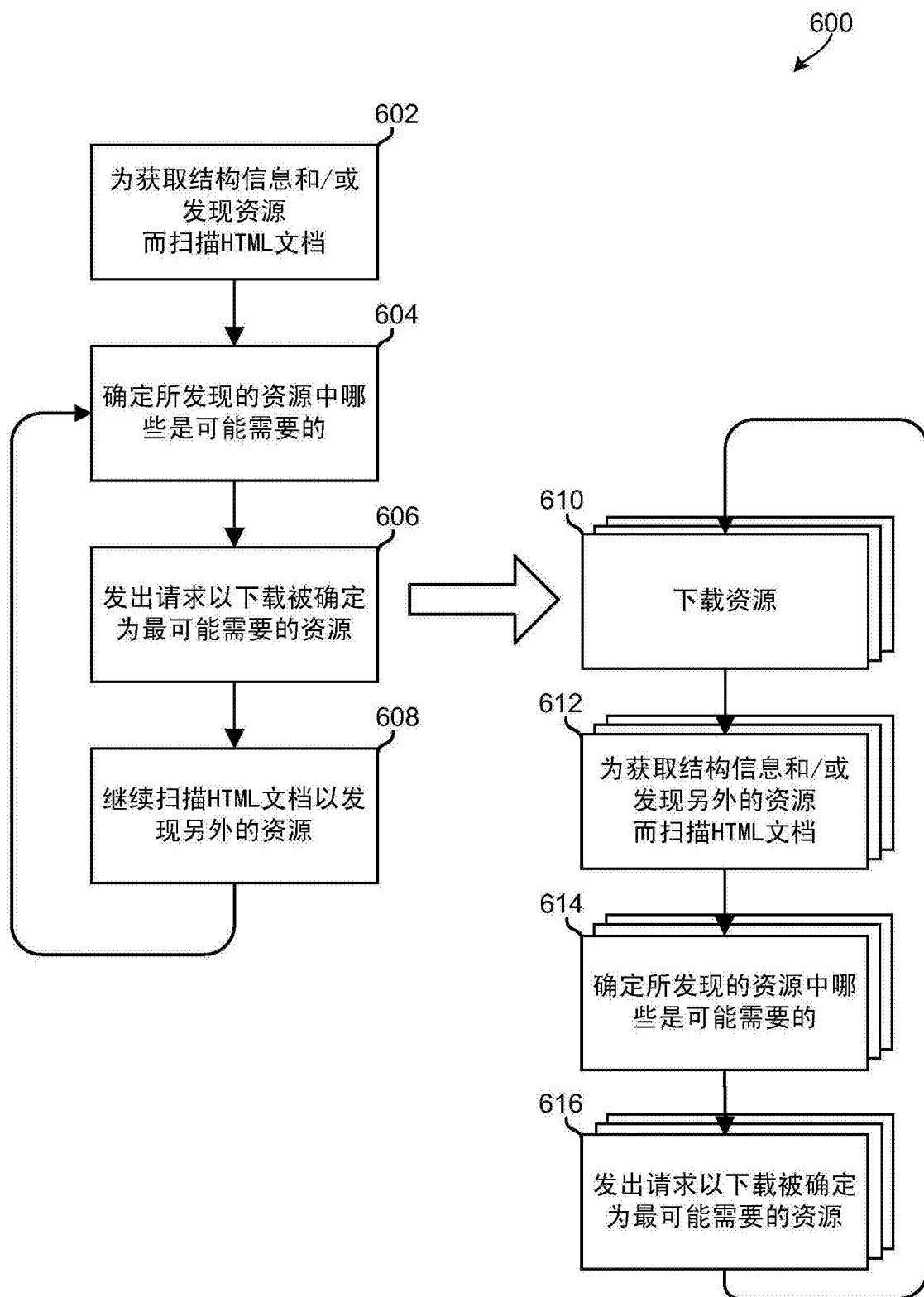


图6

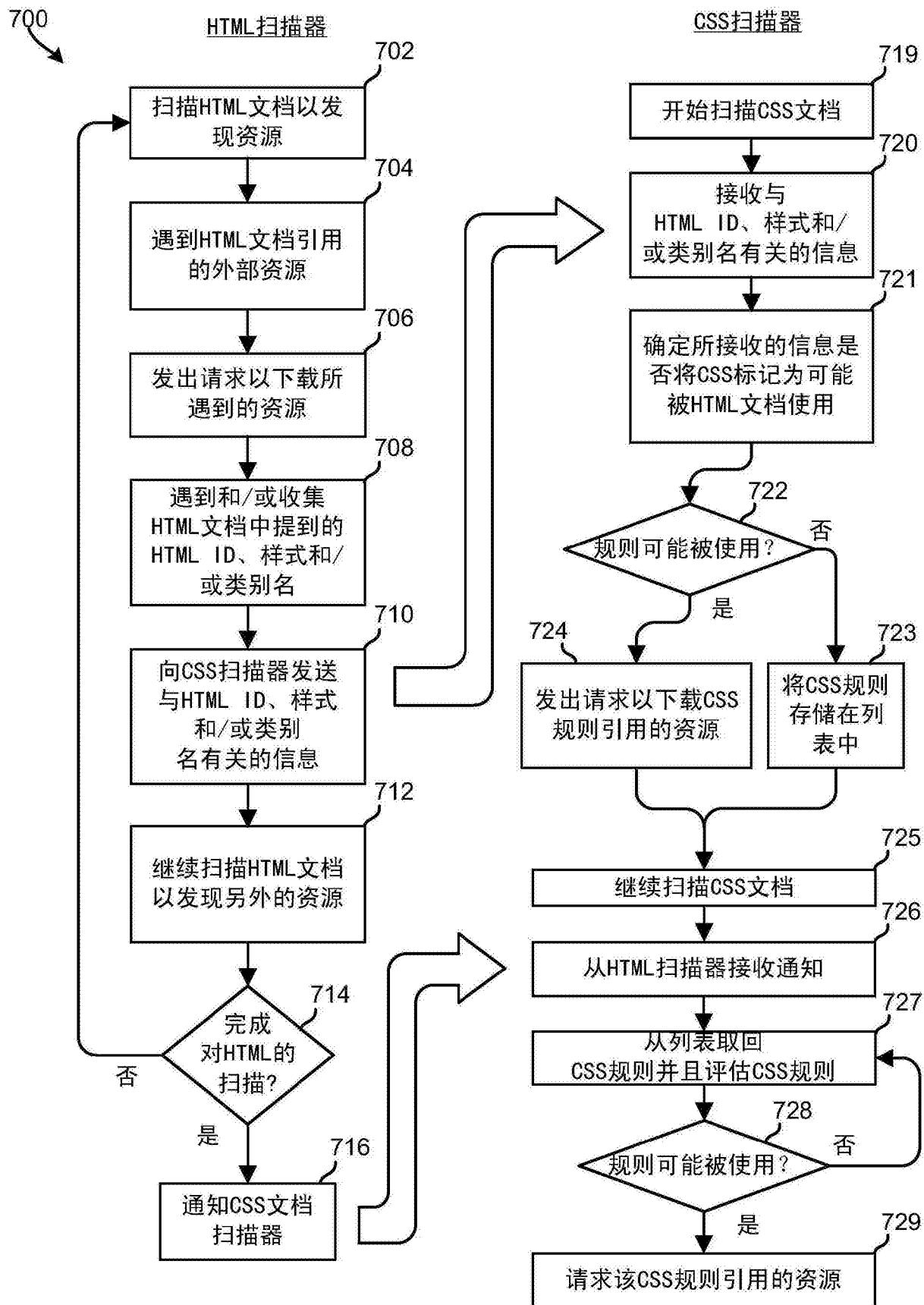


图7A

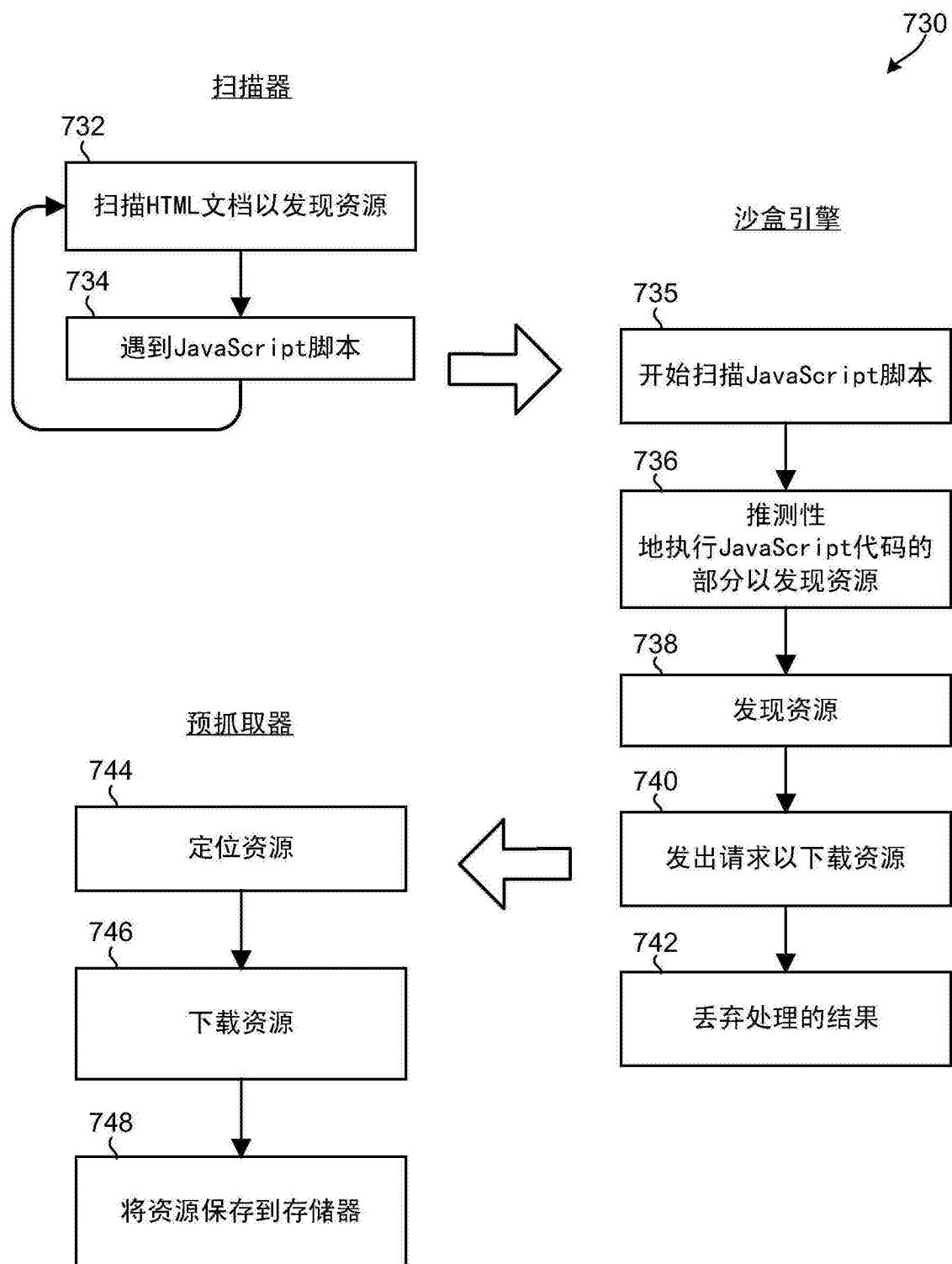


图7B

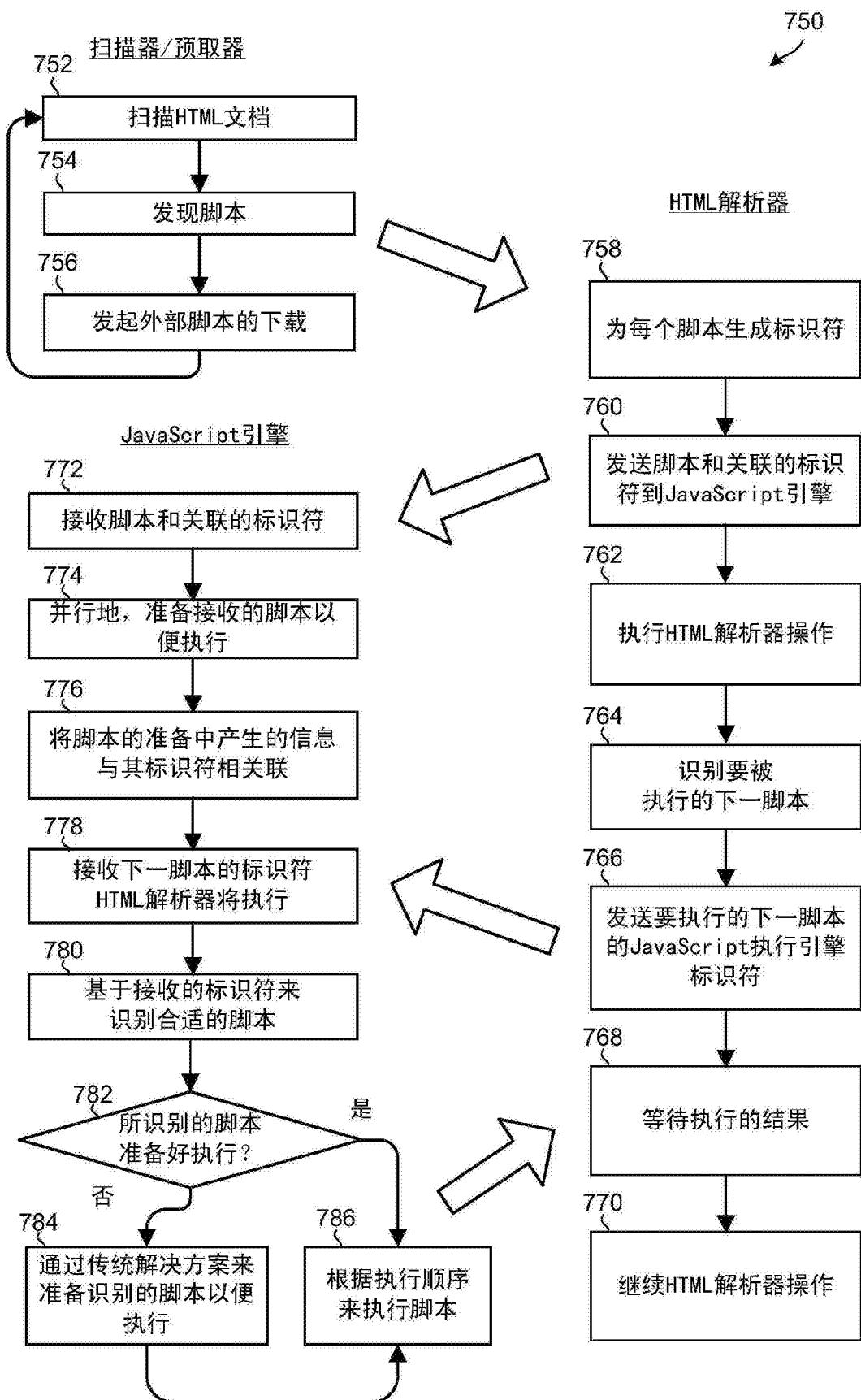


图7C

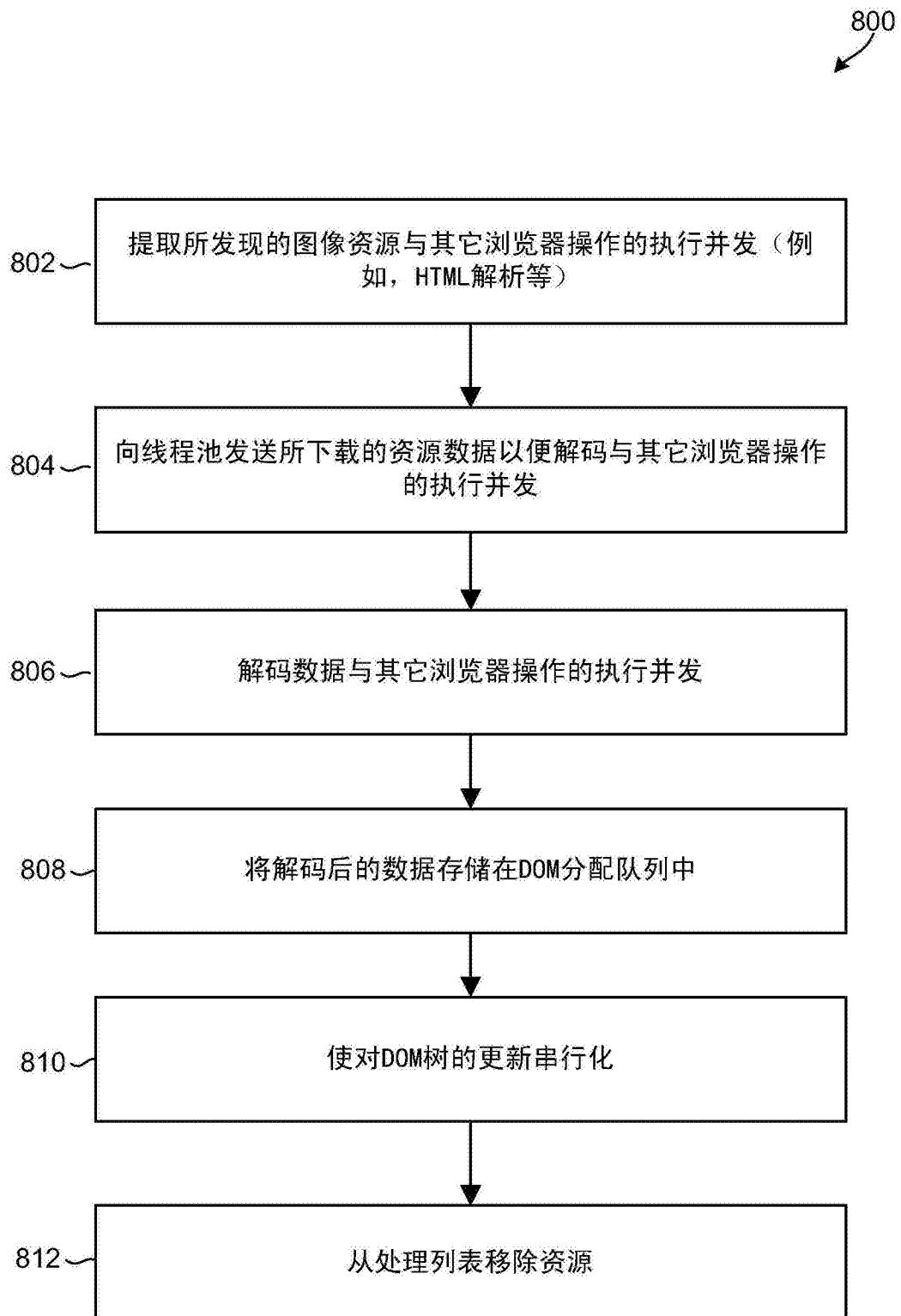


图8

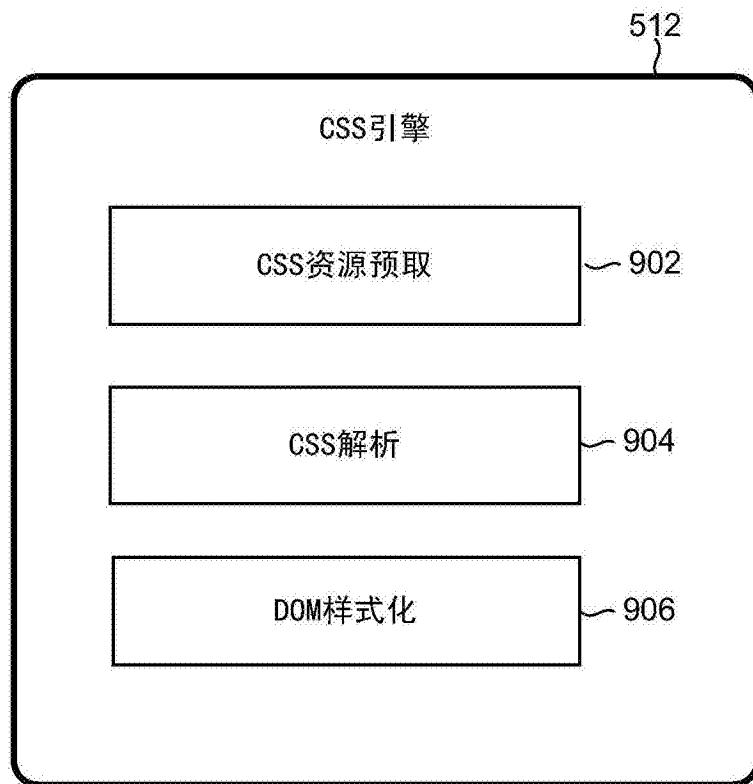


图9

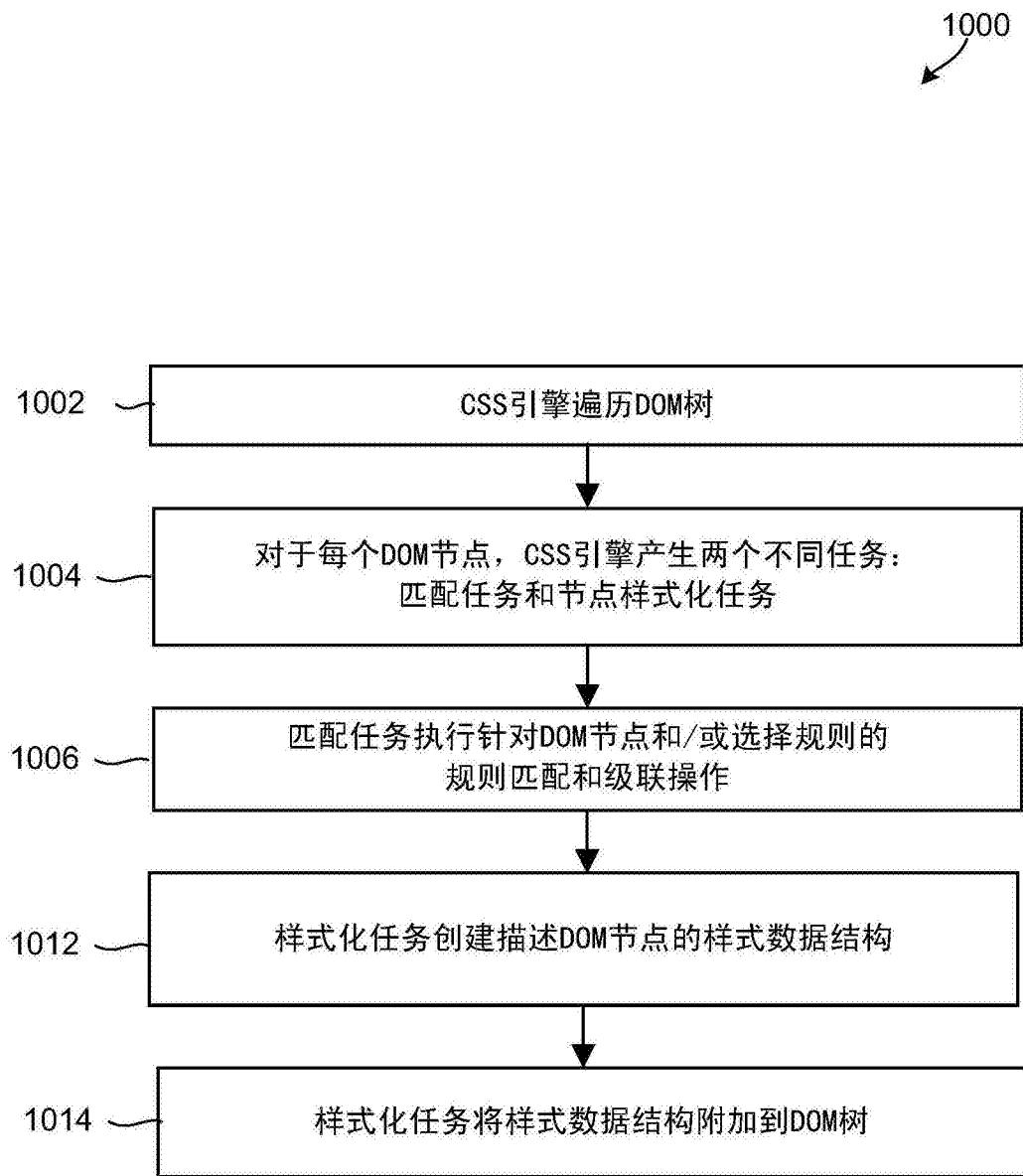


图10

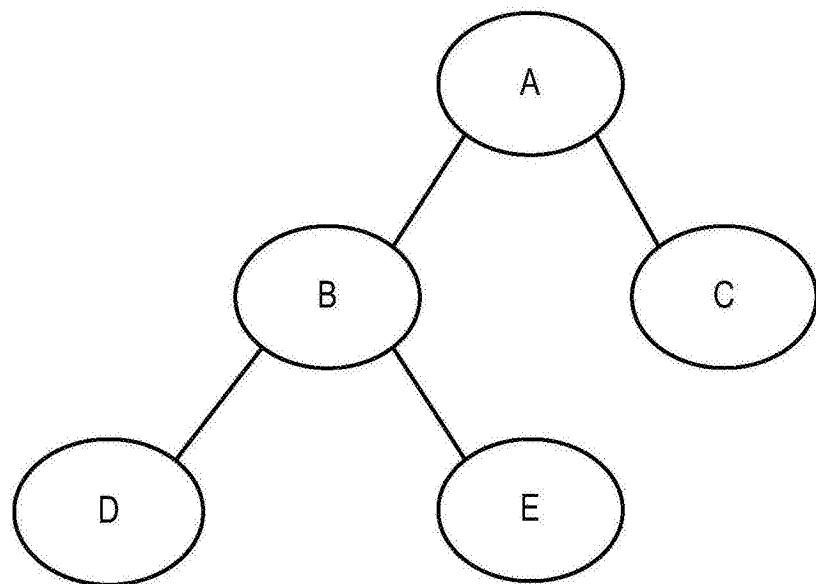


图11A

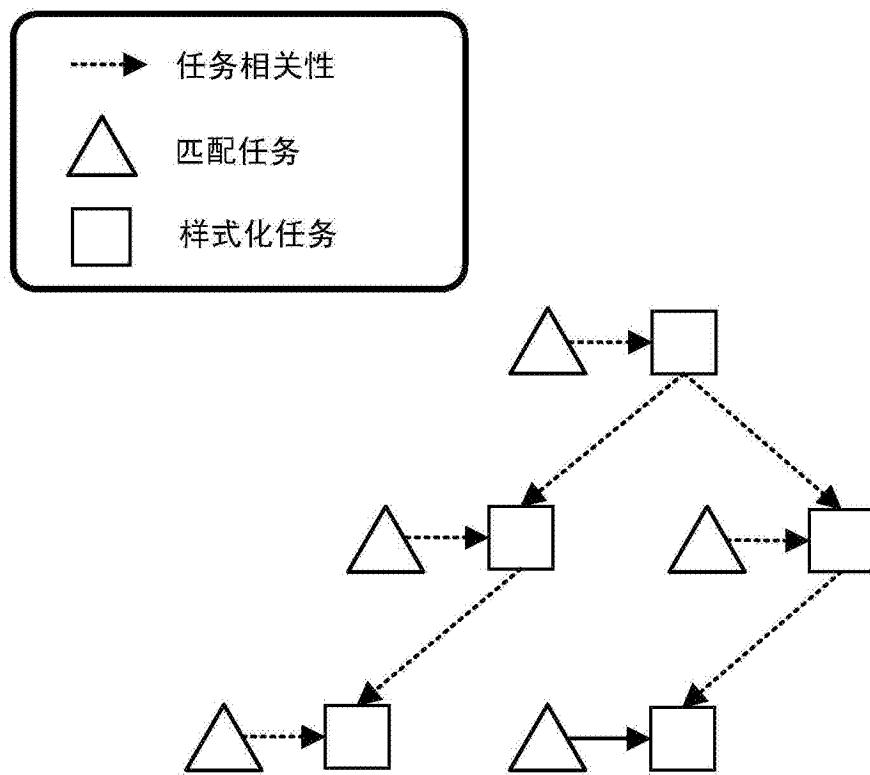


图11B

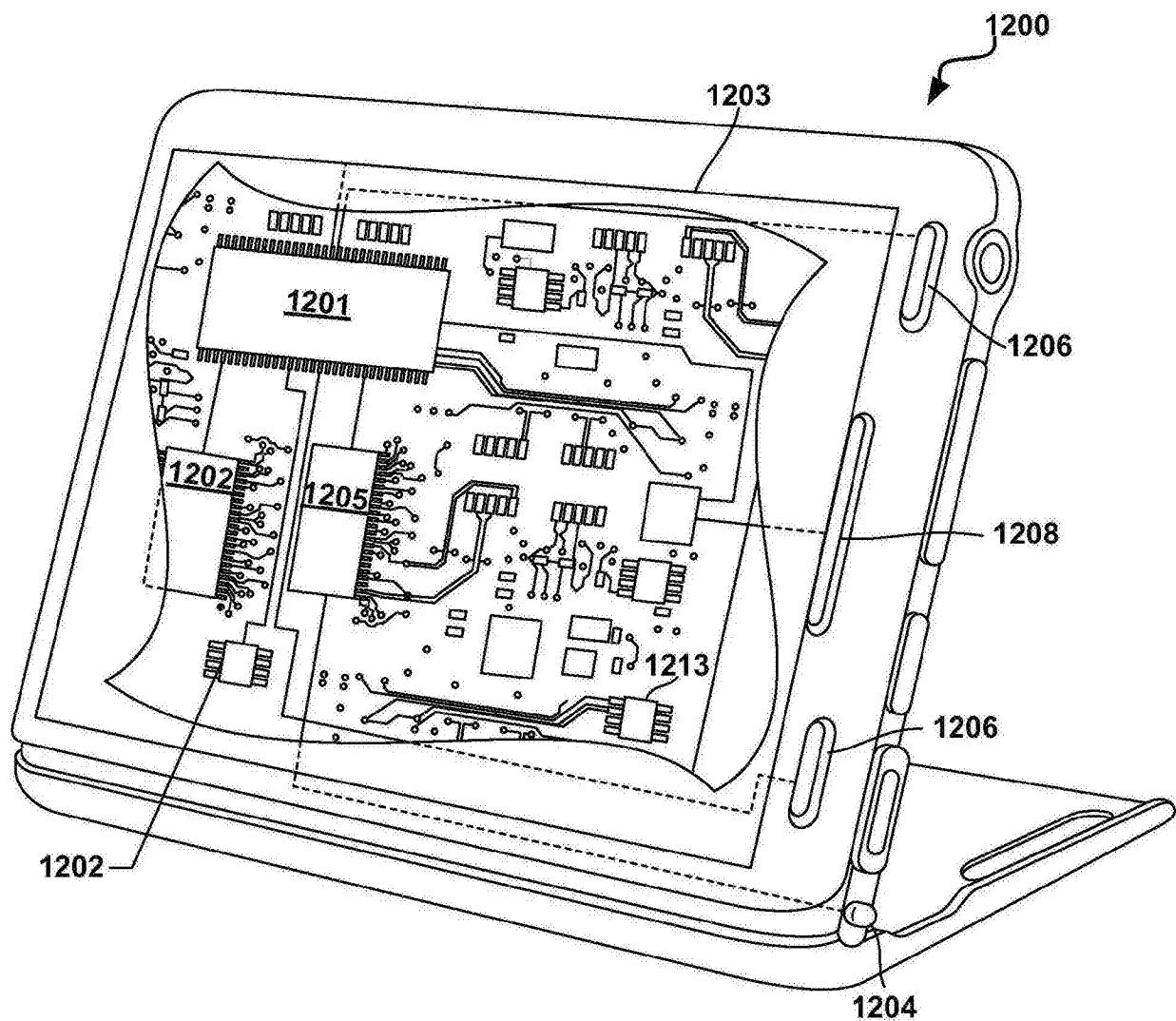


图12

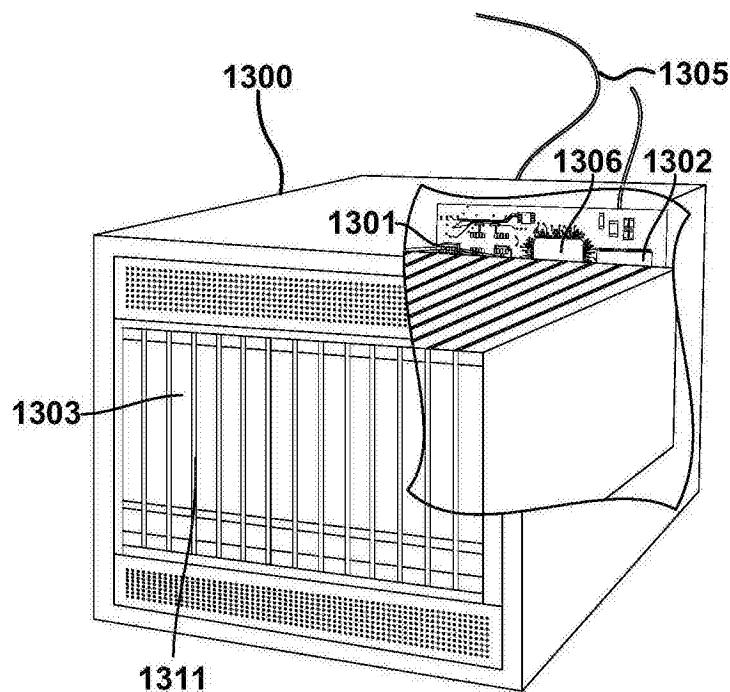


图13

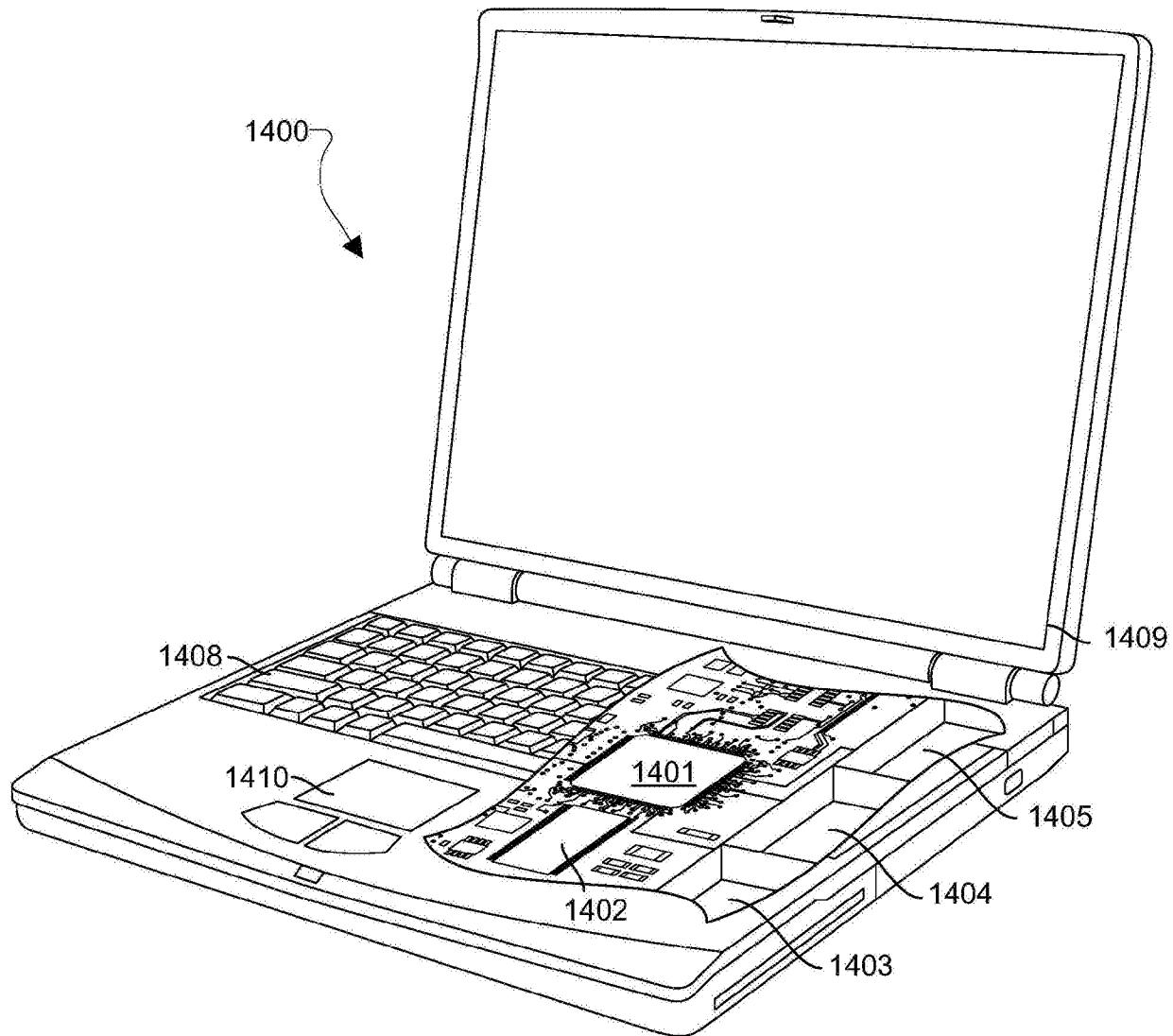


图14