



## (12) 发明专利

(10) 授权公告号 CN 102123178 B

(45) 授权公告日 2014. 04. 09

(21) 申请号 201110075213. 7

(51) Int. Cl.

(22) 申请日 2005. 12. 30

H04L 29/08 (2006. 01)

(30) 优先权数据

11/039, 946 2005. 01. 24 US

11/169, 002 2005. 06. 29 US

(56) 对比文件

WO 2004/001585 A1, 2003. 12. 31, 全文.

CN 1450766 A, 2003. 10. 22, 全文.

(62) 分案原申请数据

200580049254. 1 2005. 12. 30

审查员 孔令通

(73) 专利权人 茨特里克斯系统公司

地址 美国佛罗里达州

(72) 发明人 P. 赫马尼 P. 森达拉延 K. 凯拉什

A. 索尼 R. 辛哈 S. 安娜马莱萨米

K. R. 布尚

(74) 专利代理机构 中国专利代理(香港)有限公司

司 72001

代理人 李娜 高为

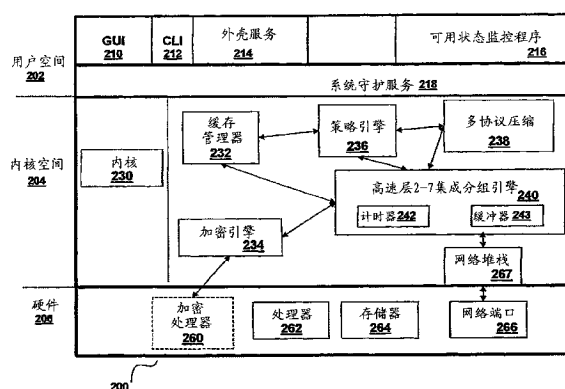
权利要求书2页 说明书42页 附图14页

## (54) 发明名称

在网络中对动态产生的对象执行缓存的系统和方法

## (57) 摘要

本发明的解决方案提高了缓冲存储器存储和提供动态产生的数据的能力。本发明也使缓存器能够有效处理各种不同的应用请求类型, 因此增加应用性能并消除了用于保持从缓存器提供的数据的新鲜度的管理复杂性。本发明除了结合以不复制最初产生对象的应用服务器所执行的处理的方式理解和处理数据的能力之外, 通过使用试探法提供了缓存动态内容的有效方法以有效地预测该应用服务器的性能。本发明的这些技术又增加了动态缓存的使用, 并因此有助于改进网络和潜在的应用基础设施的性能。



1. 一种用于响应来自多个客户端对于动态产生对象的请求的方法，该方法包括步骤：
  - (d) 由设备从始发服务器接收第一客户端对于动态产生对象的第一请求的响应，该响应包括动态产生的对象；
  - (e) 由设备请求分组处理引擎以传送响应到第一客户端，所述响应保持在缓冲器中同时等待传送；
  - (f) 在完成将所述响应传送到第一客户端之前，由设备从第二客户端接收对于动态产生对象的第二请求；以及
  - (g) 由设备请求分组处理引擎以将缓冲器中保持的对于第一客户端的响应的动态产生对象传送到第二客户端。
2. 如权利要求 1 的方法，包括由分组处理引擎将响应传送到第一客户端。
3. 如权利要求 1 的方法，包括响应第二请求，由分组处理引擎将动态产生的对象传送到第二客户端。
4. 如权利要求 1 的方法，包括在将响应传送到第一客户端或第二客户端之一之后，由分组处理引擎从缓冲器中移除响应。
5. 如权利要求 1 的方法，其中所述缓冲器包括 TCP / IP 缓冲器或设备的 TCP / IP 堆栈的一部分。
6. 如权利要求 1 的方法，其中所述设备包括单个 TCP/IP 堆栈用于向第一客户端和第二客户端传送，缓冲器与单个 TCP/IP 堆栈相关联。
7. 如权利要求 1 的方法，其中步骤 (c) 进一步包括由设备从多个客户端接收对于动态产生对象的多个请求，以及其中步骤 (d) 进一步包括由设备请求以将缓冲器中保持的对于第一客户端的响应的动态产生对象传送到多个客户端。
8. 如权利要求 1 的方法，包括根据与第一客户端到网络的连接相关的一个或多个特性由分组处理引擎保持对第一客户端的响应。
9. 如权利要求 1 的方法，包括由所述设备排队第二请求。
10. 如权利要求 1 的方法，包括在向第一客户端和第二客户端传送动态产生的对象之后从缓冲器移除动态产生的对象。
11. 如权利要求 1 的方法，其中所述动态产生的对象被识别为不可缓存的。
12. 如权利要求 1 的方法，包括由所述设备通过与第二请求相关的对象决定因素确定第二请求是用于第一请求的动态产生的对象。
13. 如权利要求 1 的方法，其中所述设备包括在第一客户端和始发服务器之间进行通信中的装置、网络设备或计算设备之一。
14. 如权利要求 1 的方法，其中所述设备包括与分组处理引擎进行通信的缓存器。
15. 如权利要求 14 的方法，包括由所述缓存器请求分组处理引擎以向第二客户端传送对第一客户端的响应或第一客户端的动态产生对象。
16. 一种用于对请求进行响应的系统，所述请求是来自多个客户端针对动态产生的对象，该系统包括：
  - 一个设备，用于该设备从始发服务器接收第一客户端对于动态产生对象的第一请求的响应，该响应包括动态产生的对象，该设备包括：
    - 用于保持等待从设备传送的一个或多个网络分组的缓存器；

从所述设备接收请求以向第一客户端传送响应的分组处理引擎,缓冲器保持包括响应的一个或多个网络分组;以及

其中在完成将响应传送到第一客户端之前,所述设备从第二客户端接收对于动态产生对象的第二请求,并请求分组处理引擎以将缓冲器中保持的响应的动态产生对象传送到第二客户端。

17. 如权利要求 16 的系统,其中所述分组处理引擎将响应从缓冲器传送到第一客户端。

18. 如权利要求 16 的系统,其中响应所述第二请求,分组处理引擎将动态产生的对象从缓冲器传送到第二客户端。

19. 如权利要求 16 的系统,其中在将响应传送到第一客户端或第二客户端之一后,分组处理引擎从缓冲器中移除响应。

20. 如权利要求 16 的系统,其中所述缓冲器包括 TCP / IP 缓冲器或所述设备的 TCP / IP 堆栈的一部分。

21. 如权利要求 16 的系统,其中所述设备包括单个 TCP / IP 堆栈用于向第一客户端和第二客户端传送,缓冲器与单个 TCP / IP 堆栈相关联。

22. 如权利要求 16 的系统,其中所述设备从多个客户端接收对于动态产生对象的多个请求,以及其中所述设备请求分组处理引擎以将缓冲器中保持的对于第一客户端的响应的动态产生对象传送到多个客户端。

23. 如权利要求 16 的系统,其中根据与第一客户端到网络的连接相关的一个或多个特性所述分组处理引擎保持对第一客户端的响应。

24. 如权利要求 16 的系统,其中所述设备排队第二请求。

25. 如权利要求 16 的系统,其中在向第一客户端和第二客户端传送动态产生的对象之后所述分组处理引擎从缓冲器移除动态产生的对象。

26. 如权利要求 16 的系统,其中所述动态产生的对象被识别为不可缓存的。

27. 如权利要求 16 的系统,包括由所述设备通过与第二请求相关的对象决定因素确定第二请求是用于第一请求的动态产生的对象。

28. 如权利要求 16 的系统,其中所述设备包括在第一客户端和始发服务器之间进行通信中的装置、网络设备或计算设备之一。

29. 如权利要求 16 的系统,其中所述设备包括与分组处理引擎进行通信的缓存器。

30. 如权利要求 29 的系统,其中所述缓存器请求分组处理引擎以向第二客户端传送对第一客户端的响应或第一客户端的动态产生对象。

## 在网络中对动态产生的对象执行缓存的系统和方法

[0001] 本申请是申请日为 2005 年 12 月 30 日、申请号为 200580049254.1、发明名称为“在网络中对动态产生的对象执行缓存的系统和方法”的专利申请的分案申请。

[0002] 相关申请

[0003] 本申请要求于 2005 年 6 月 29 日提交的题目为“METHOD AND DEVICE FOR PERFORMING INTEGRATED CACHING IN A DATA COMMUNICATION NETWORK”、申请号为 11/169,002 的美国专利申请和于 2005 年 1 月 24 日提交的题目为“SYSTEM AND METHOD FOR ESTABLISHING A VIRTUAL PRIVATE NETWORK”、申请号为 11/039,946 的美国专利申请的优先权,本文在此引入这两篇申请以供参考。

### 技术领域

[0004] 本发明总体上涉及在网络中缓存数据。具体地,本发明涉及在网络中对动态产生的对象执行缓存的设备、系统和方法。

### 背景技术

[0005] 网络通信量的持续发展速度使得承载该通信量的基础设施变得紧张。已经提出了允许网络运营商处理该日益严重的问题的各种解决方案,其中包括缓存技术的开发。使用常规缓存,可以重新使用静态内容并将该静态内容提供给多个客户端而不会加重服务器基础设施的负荷。另外,缓冲存储器允许静态内容更接近终端用户来存储,因此改进了响应时间并同时减少服务器基础设施的负荷。降低的响应时间和减少的服务器基础设施的负荷减少了该基础设施的带宽和处理需求。

[0006] 然而,经过网络传送的日益增加的内容量是动态产生的,其中包括由企业计算解决方案和复杂因特网应用产生的较大百分比的网络通信量。动态产生的内容是当请求对象时服务器产生的内容,并通常是基于从客户端接收的输入。因此,它经常改变通过时间和对于产生系统进行的输入。动态内容的常规示例包括由客户端进行的股票行情表请求或数据库搜索。在每个实例中,在接收到特定客户请求后实时产生响应对象。

[0007] 对缓存动态产生的内容的挑战是多方面的。例如,缓存动态产生的内容没有通常公认的标准或规范。因为对于指定是否缓存动态产生的对象没有标准,所以该对象一般被看作是不可缓存的。另一挑战在于确定动态产生的对象的“新鲜”度的有效性,这是因为用于产生该对象的潜在数据的改变是无规律并不可预测的。

[0008] 除了上述困难之外,对动态产生的内容的请求一般也比对静态内容的请求更复杂。动态请求经常包含需要由目标应用处理或解析的信息字符串以识别将由缓存器使用的可应用参数用于识别与该请求有关的合适对象。然而,这些参数很少由客户端以逻辑或相容的顺序放在请求中。为了确定该请求识别大量动态产生的对象中的哪一个,每个该请求必须被标准化(即,以非任意顺序放置参数)。

[0009] 而且,利用动态产生的内容使一个请求与动态产生的对象匹配成为更复杂的任务,这是因为由应用所执行的特定处理需要被复制或另外通过有根据的推测来预料。该复

制或推测对于确定由缓存器所存储的对象是否适于提供给特定的即将到来的请求是很有必要的。该复杂性是由于应用本身复杂性而引起,也是因为响应内容可以是请求内容以及诸如用户身份(该用户身份在请求中可能出现或可能不出现)、时间、用户数据库的当前状态之类的特定其他外部变量,和无数其它因素的函数。

[0010] 总体上,缓存最初围绕缓存静态对象而开发。当因特网和应用变得越来越取决于传送动态产生的内容时,需要一种解决方案,该解决方案将缓存的优势扩展到动态内容,并且解决了这样的内容对于常规缓存技术所提出的各种挑战。

## 发明内容

[0011] 本发明的解决方案提高了缓冲存储器存储和提供动态产生的数据的能力。本发明也使缓存器能够有效处理各种不同的应用请求类型,因此增加应用性能并消除了用于保持从缓存器提供的数据的新鲜度的管理复杂性。本发明除了结合以不复制最初产生对象的应用服务器所执行的处理的方式理解和处理数据的能力之外,通过使用试探法提供了缓存动态内容的有效方法以有效地预测该应用服务器的性能。本发明的这些技术又增加了动态缓存的使用,并因此有助于改进网络和潜在的应用基础设施的性能。

[0012] 在一个实施例中,本发明涉及一种在缓存器中缓存和维持动态产生的对象的方法和系统。本发明的技术包括在缓存器接收无效命令以无效对象,诸如之前从始发服务器(originating server)提供并存储在缓存器中的动态产生的对象。从始发服务器中所述动态产生的对象没有被识别为可缓存的。缓存器接收的无效命令识别缓存的动态产生的对象。响应无效命令,缓存器将缓存的动态产生对象标记为无效,并从缓存器刷新该对象。

[0013] 在另一实施例中,本发明也提供了用于使用对象决定因素(object determinant)识别缓存的动态产生对象的技术。缓存器可以截取客户端和服务端之间的通信,并解析通信以识别对象决定因素。对象决定因素能识别之前提供并存储在缓存器中的对象。缓存器根据对象决定因素确定在始发服务器由对象决定因素所识别的对象中变化是否已发生或将发生。如果变化已发生或将发生,缓存器将由对象决定因素所识别的对象在缓存器中标记为无效。一旦将对象标记为无效,缓存器刷新无效对象并从始发服务器检索对象。

[0014] 本发明的另一些实施例应用上述方法至动态产生的对象组。例如,先前提提供的动态产生的对象组是在缓存器中形成的。对象组与至少一个对象决定因素有关。组的记录被保持在缓存器中并与对象决定因素有关。如果由缓冲器截取的通信所识别的对象决定因素表明在始发服务器上组的一个或多个对象中变化已发生或将发生,则将之前提供的对象的组标记为无效。

[0015] 一方面,本发明涉及一种对动态产生的对象进行缓存的方法,该动态产生的对象没有被识别为可缓存。该方法包括在缓存器中存储动态产生的对象,该动态产生的对象没有被识别为可缓存,从始发服务器提供所述动态产生的对象;由所述缓存器接收一个请求以无效所述缓存的动态产生的对象。响应所述请求,所述缓存器将所述缓存的动态产生的对象标记为无效。可以在始发服务器和客户端之间进行通信的任何设备(例如,装置、网络设备或计算设备)上操作该缓存器。

[0016] 在某些实施例中,本发明的方法包括由所述始发服务器请求无效所述缓存的动态产生的对象。在其他实施例中,响应所述始发服务器中动态产生对象的改变,所述始发服务

器自动请求无效所述缓存的动态产生的对象。在另一实施例中,客户端与始发服务器进行通信以接收动态产生的对象,并且客户端请求无效所述缓存的动态产生的对象。在又另一实施例中,外部管理控制器请求无效所述缓存的动态产生的对象。

[0017] 在一个实施例中,本发明的方法包括从所述缓存器刷新被标记为无效的所述缓存的动态产生的对象。在另一实施例中,在所述缓存动态产生对象的例如十毫秒或更少时间的很短时间内缓存器接收无效请求。在一些实施例,响应于很短的期限(expiry)届满,例如缓存对象的 10 毫秒或更少的期限,所述缓存器无效缓存的动态产生的对象。

[0018] 另一方面,本发明涉及一种对动态产生的对象组进行缓存的方法。在一些实施例中,该对象组具有没有被识别为可缓存的至少一个对象。该方法包括在缓存器中识别从始发服务器之前提供的动态产生的对象组。所述缓存器将所述组和对象决定因素相关联。本发明还包括由所述缓存器截取一个通信,所述通信识别组的对象决定因素以及表示在始发服务器上对所述组的对象之一的变化将要发生或已经发生。在一个实施例中,本发明的方法包括响应于截取所述通信或识别对象决定因素,由所述缓存器将所述动态产生的对象组标记为无效。

[0019] 而且,在本发明的某些实施例中,缓存器从所述缓存器刷新被标记为无效的对象组。在其他实施例中,所述动态产生的对象组是预先指定的。在其他实施例中,本发明自动识别所述动态产生的对象组并根据一个规则使所述对象决定因素和所述组相关联。

[0020] 一方面,本发明涉及用于维持动态产生的对象的缓存器的另一种方法。本发明的方法包括由缓存器截取客户端和始发服务器之间的通信,例如客户端向始发服务器请求动态产生的对象。动态产生的对象之前已由始发服务器提供并存储在缓存器中。该方法在表示所述始发服务器上动态产生对象的变化已发生或将发生的通信中由缓存器识别对象决定因素,并由所述缓存器将所述缓存的动态产生的对象标记为无效。然后,该方法从始发服务器获得所请求的动态产生的对象。

[0021] 在本发明的一个实施例中,缓存器刷新无效的动态产生的对象。在另一实施例中,缓存器使动态产生的对象和从始发服务器之前提供的动态产生的对象组相关联,将该组和对象决定因素相关联,并响应请求将所述动态产生的对象组标记为无效。在某些实施例中,所述动态产生的对象没有被识别为可缓存的。在其他实施例中,缓存器刷新被标记为无效的所述动态产生的对象组。动态产生的对象组可以是预指定的或另外根据一个规则通过对对象决定因素而被自动识别的。

[0022] 在本方法的某些实施例中,客户端在通信中嵌入对象决定因素作为预定字符串。在其他实施例中,缓存器根据与动态产生对象相关的预定启发式规则识别对象决定因素。在一个实施例中,缓存器从通信的如下元素之一中选择对象决定因素:1) USERID, 2) IP 地址, 3) TCP 端口, 4) HTTP 报头, 5) 自定义 HTTP 报头, 6) 客户 URL, 7) cookie 报头, 8) URL 查询字符串和 9) POST 主体。缓存器也根据用户配置的无效策略从通信中提取对象决定因素。

[0023] 在另一实施例中,本发明的方法包括维持缓存器中的一个表以使对象决定因素和缓存器中存储的一个或多个对象或对象组相关联。在某些实施例中,本方法包括由智能统计引擎检查来自客户端的通信以识别动态产生的对象集以与缓存器中的组相关联。缓存器可以将存储在缓存器中的对象与内容组相关联。该内容组由具有作为索引的具体数(incarnation number)的散列表来表示。在其他实施例中,缓存器对通过请求所识别的动

态产生的对象执行散列算法以确定动态产生对象的变化。

[0024] 在某些方面,本发明涉及一种对动态产生的对象进行缓存的系统,该动态产生的对象没有被识别为可缓存,该系统包括用于在缓存器中存储动态产生的对象的装置,该动态产生的对象没有被识别为可缓存,从始发服务器提供所述动态产生的对象;以及用于由所述缓存器接收一个请求以无效所述缓存的动态产生的对象的装置。该系统也包括用于响应所述请求,由所述缓存器将所述缓存的动态产生的对象标记为无效的装置。

[0025] 在其他方面,本发明涉及一种对诸如动态产生对象之类的对象组进行缓存的系统。在某些实施例中,该动态产生的对象具有没有被识别为可缓存的至少一个对象。该系统包括用于在缓存器中识别从始发服务器之前提供的动态产生的对象组的装置,所述动态产生的对象中的至少一个没有被识别为可缓存;以及用于由缓存器将组和对象决定因素相关联的装置。该系统也包括用于由所述缓存器截取一个通信的装置,所述通信识别组的对象决定因素以及表示在始发服务器上对所述组的对象之一的变化将要发生或已经发生。

[0026] 一方面,本发明涉及一种用于维持动态产生的对象的缓存器的系统。该系统包括用于由缓存器截取客户端和始发服务器之间的通信的装置。该系统也包括用于在表示所述始发服务器上动态产生对象的变化已发生或将发生的通信中由缓存器识别对象决定因素的装置;以及用于由缓存器将缓存的动态产生对象标记为无效的装置。在某些实施例中,系统也包括用于从始发服务器获得动态产生的对象的装置。

[0027] 另一方面,本发明涉及一种用于提供缓存器中存储的动态产生对象的颗粒度无效的方法和系统。本发明的技术结合了将缓存器所存储的对象的届满(expiration)时间配置为微小的颗粒度时间间隔的能力,诸如由分组处理引擎的分组处理计时器提供的时间间隔的颗粒度(granularity)。因此,本发明可以缓存期限时间小到非常小时间间隔的对象。该特性被称为“无效颗粒度”(invalidation granularity)。通过在期限时间中提供该微小的颗粒度,本发明的缓存器可以缓存并提供频繁变化的对象,经常甚至是一秒内改变数次的对象。一种技术是调节(leverage)本发明的设备所用的分组处理计时器,这样能够以毫秒级的时间增量操作以允许无效或期限颗粒度低至10ms。当本发明的集成缓存器响应分组处理引擎的分组处理计时器而执行缓存操作时,对象的无效可以在以毫秒级的时间段内发生,并具有以分组处理计时器所提供的时间间隔级别的期限(诸如10ms)的对象。

[0028] 一方面,本发明涉及一种响应分组处理计时器的信号而无效缓存器中缓存的动态产生对象的方法。该方法可以在设备的分组处理引擎中执行,其中分组处理引擎处理具有至少一个动态产生对象的网络分组并在缓存器中存储动态产生的对象。该方法包括从分组处理计时器接收第一信号以处理网络分组;响应于所述信号处理网络分组;以及向缓存器传送第二信号,所述第二信号表示所缓存的动态产生对象是无效的。在一个实施例中,第二信号提供无效命令以无效缓存的动态产生对象。在另一实施例中,第二信号提供缓存的动态产生对象的期限届满。

[0029] 在某些实施例中,在处理网络分组期间传送第二信号,而在其他实施例中,一旦完成网络分组的处理就传送第二信号。在本发明的一个实施例中,分组处理计时器以一或多微秒的时间增量操作。分组处理计时器以一个时间增量操作以在10毫秒或更短的期限时间无效缓存器中所存储的对象。该方法包括由缓存器将缓存的动态产生对象标记为无效。在一个实施例中,该方法包括从始发服务器传送包括动态产生对象的网络分组。

[0030] 具有分组处理引擎的设备包括任何如下设备：1) 电桥、2) 路由器、3) 转换器，以及 4) SSL VPN 设备。在某些实施例中，缓存器、分组处理引擎或分组处理计时器在设备的内核空间内操作。在某些实施例中，该方法包括通过分组处理计时器为缓存的动态产生对象的期限提供时间间隔为等于或小于一个持续时间的时间以从始发服务器动态产生并提供动态产生的对象。当该动态产生的对象从始发服务器提供时，该动态产生的对象被识别为不可缓存的。

[0031] 另一方面，本发明涉及一种响应分组处理计时器的信号而无效缓存的动态产生对象的设备。该设备处理具有至少一个动态产生对象的网络分组并在缓存器中存储该动态产生的对象。该设备包括产生第一信号的分组处理计时器，以及与分组处理计时器进行通信的分组处理引擎。分组处理引擎响应接收到分组处理计时器的第一信号而处理网络分组。该设备也包括与分组处理引擎进行通信并具有在存储元件中缓存的动态产生对象的缓存器。该缓存器从分组处理引擎接收第二信号，该第二信号表示缓存的动态产生对象是无效的。在一个实施例中，第二信号包括无效命令以无效缓存的动态产生对象。在另一实施例中，第二信号包括缓存的动态产生对象的期限届满。

[0032] 在本发明的设备的一个实施例中，在由分组处理引擎处理网络分组期间分组处理引擎向缓存器传送第二信号。在另一实施例中，一旦完成网络分组的处理，分组处理引擎就向缓存器传送第二信号。在某些实施例中，分组处理计时器以一或多个毫秒的时间间隔操作。在一个实施例中，分组处理计时器以一个时间间隔操作以向缓存器触发第二信号用于在 10 毫秒或更少的第一时间间隔无效缓存的对象。

[0033] 本发明的设备包括如下设备之一：1) 电桥、2) 路由器、3) 转换器，以及 4) SSL VPN 设备。缓存器、分组处理引擎或分组处理计时器可以在设备的内核空间内操作。在某些实施例中，分组处理计时器为缓存的动态产生对象的期限提供时间间隔小于或等于一个持续时间以从始发服务器动态产生并提供动态产生的对象。从始发服务器中，该动态产生的对象没有被识别为可缓存的。始发服务器可以传送具有动态产生的对象的网络分组。

[0034] 另一方面，本发明涉及一种提供称为闪速缓存的技术的方法和系统，用于响应来自多个客户端对于诸如动态产生对象之类的对象的请求。本发明的该技术使用缓冲器中存储的动态产生对象以向客户端传送，例如响应来自客户端的请求，也响应来自其他客户端对动态产生对象的另外请求并将对象保存在缓存器中。使用该技术，本发明能够增加对于极快改变对象的缓存命中率，诸如不能被另外缓存的动态产生的对象。

[0035] 一方面，本发明涉及一种用于响应来自多个客户端对于动态产生对象的请求的方法。该方法包括由设备从始发服务器接收第一客户端对于动态产生对象的第一请求的响应。该响应包括动态产生的对象。该方法也包括由设备请求分组处理引擎以传送响应到第一客户端。该响应保持在缓冲器中同时等待传送。该方法进一步包括在完成将所述响应传送到第一客户端之前，由设备从第二客户端接收对于动态产生对象的第二请求，以及由设备请求分组处理引擎以将缓冲器中保持的对于第一客户端的响应的动态产生对象传送到第二客户端。

[0036] 在一个实施例中，本发明的方法包括由分组处理引擎将响应传送到第一客户端。在另一实施例中，该方法包括响应第二请求，由分组处理引擎将动态产生的对象传送到第二客户端。在某些实施例中，该方法包括在将响应传送到第一客户端或第二客户端之一后，



由分组处理引擎从缓冲器中移除响应。在某些实施例中，本发明所用的缓冲器包括 TCP/IP 缓冲器或所述设备的 TCP/IP 堆栈的一部分。该设备包括单个 TCP/IP 堆栈用于向第一客户端和第二客户端传送，并且缓冲器与单个 TCP/IP 堆栈相关联。

[0037] 在本发明的某些实施例中，该方法包括由设备从多个客户端接收对于动态产生对象的多个请求，以及由设备请求以将缓冲器中保持的对于第一客户端的响应的动态产生对象传送到多个客户端。在一个实施例中，分组处理引擎根据与第一客户端到网络的连接相关的一个或多个特性而保持对第一客户端的响应。在另一实施例中，设备排队第二请求，而不是将第二请求传送到始发服务器。在一个实施例中，该设备使用缓冲器中保持的第一请求的响应而响应第二请求。

[0038] 在某些实施例中，本发明的方法在向第一客户端和第二客户端传送动态产生的对象之后从缓冲器移除动态产生的对象。在一个实施例中，例如当从始发服务器提供该动态产生的对象时，该动态产生的对象被识别为不可缓存的。在该方法的其他实施例中，所述设备通过与第二请求相关的对象决定因素确定第二请求是用于第一请求的动态产生的对象。

[0039] 在某些实施例中，设备可以是在第一客户端和始发服务器之间进行通信的装置、网络设备或计算设备之一。在其他实施例中，设备包括与分组处理引擎进行通信的缓存器。在一个实施例中，该方法包括由缓存器请求分组处理引擎以向第二客户端传送对第一客户端的响应或第一客户端的动态产生对象。

[0040] 另一方面，本发明涉及一种用于对请求进行响应的系统，所述请求是来自多个客户端针对动态产生的对象。该系统包括一个设备，用于从始发服务器接收第一客户端对于动态产生对象的第一请求的响应，该响应包括动态产生的对象。该设备包括用于保持等待从该设备传送的一个或多个网络分组的缓存器。该设备还包括从所述设备接收请求以向第一客户端传送响应的分组处理引擎，缓冲器保持包括响应的一个或多个网络分组。在完成将响应传送到第一客户端之前，所述设备从第二客户端接收对于动态产生对象的第二请求，并请求分组处理引擎以将缓冲器中保持的响应的动态产生对象传送到第二客户端。

[0041] 在本发明系统的一个实施例中，分组处理引擎将响应从缓冲器传送到第一客户端。在另一实施例中，其中响应第二请求，分组处理引擎将动态产生的对象从缓冲器传送到第二客户端。在本系统的某些实施例中，在将响应传送到第一客户端或第二客户端之一后，分组处理引擎从缓冲器中移除响应。另外，在其他实施例中，缓冲器是 TCP/IP 缓冲器或另外是设备的 TCP/IP 堆栈的一部分。在某些实施例中，设备包括单个 TCP/IP 堆栈用于向第一客户端和第二客户端传送，缓冲器与单个 TCP/IP 堆栈相关联。

[0042] 在本发明的另一实施例中，所述设备从多个客户端接收对于动态产生对象的多个请求，以及其中设备请求分组处理引擎以将缓冲器中保持的对于第一客户端的响应的动态产生对象传送到多个客户端。在一个实施例中，分组处理引擎根据与第一客户端到网络的连接相关的一个或多个特性而保持对第一客户端的响应。在另一实施例中，设备排队第二请求，而不是将第二请求传送到始发服务器。在某些实施例中，设备排队第二请求并使用缓冲器中保持的对于第一客户端的响应来响应第二请求。在其他实施例中，在向第一客户端和第二客户端传送动态产生的对象之后分组处理引擎从缓冲器移除动态产生的对象。在一个实施例中，动态产生的对象例如通过始发服务器被识别为不可缓存的。

[0043] 在本发明的另一实施例中，所述设备通过与第二请求相关的对象决定因素确定第

二请求是用于第一请求的动态产生的对象。在某些实施例中,所述设备是在第一客户端和始发服务器之间进行通信的装置、网络设备或计算设备。所述设备也具有与分组处理引擎进行通信的缓存器。另外,缓存器请求分组处理引擎以向第二客户端传送对第一客户端的响应或第一客户端的动态产生对象。

[0044] 另一方面,本发明涉及一种处理如下情况的“闪速群”技术,其中在该情况中在服务器正处理并返回对第一请求者的响应对象期间缓存器接收对于相同对象的另外请求,例如几乎同时的请求。一旦由缓冲器响应所有该几乎同时的请求,则从缓冲器刷新对象,而不需要另外期限时间或无效操作。本发明的该技术能在极短的时间量内缓存和提供对于对象的数据,所述对象另外被认为是不可缓存的。该方法在向大量并发用户提供快速变化的数据的应用中产生巨大的改进,所述应用诸如实时股票报价或快速变化的新闻报道。

[0045] 一方面,本发明涉及一种在网络设备中用于对请求进行响应的方法,所述请求是来自多个客户端针对动态产生的对象教导一种响应来自多个客户端对于动态产生对象的请求的方法。该方法包括从第一客户端接收对于来自始发服务器的动态产生对象的第一请求;以及向始发服务器传送第一请求。在响应第一客户端的第一请求之前,该方法包括从第二客户端接收对于动态产生的对象的第二请求。该方法进一步包括从始发服务器接收对于第一请求的响应,该响应具有动态产生的对象,并响应第一请求向第一客户端传送动态产生的对象,以及响应第二请求向第二客户端传送动态产生的对象。

[0046] 在本发明的一个实施例中,该方法包括从多个客户端接收对于动态产生的对象的多个请求,以及响应每个客户端的请求,向多个客户端传送动态产生的对象。在某些实施例中,该方法包括在完成将动态产生的对象传送到第二客户端之前,从第三客户端接收对于动态产生对象的第三请求,以及响应第三请求,向第三客户端传送动态产生的对象。

[0047] 在另一实施例中,本发明的方法几乎同时接收第一请求和第二请求。在一个实施例中,该方法排队第二请求而不是将第二请求传送到始发服务器。在某些实施例中,该方法在向第一客户端和第二客户端传送动态产生的对象之后,从缓存器中刷新该动态产生的对象。在一个实施例中,例如当从始发服务器提供该动态产生的对象时,该动态产生的对象被识别为不可缓存的。该方法包括通过与第二请求相关的对象决定因素确定第二请求是用于第一请求的动态产生的对象。在另一实施例中,该方法修改响应以包括实体标签报头或缓存控制报头并将修改的响应传送到第一客户端或第二客户端。

[0048] 另一方面,本发明涉及一种用于对请求进行响应的网络设备,所述请求是来自多个客户端针对动态产生的对象。该网络设备包括用于从第一客户端接收对于来自始发服务器的动态产生对象的第一请求的装置;用于向始发服务器传送第一请求的装置。该网络设备还包括一个装置,用于在响应第一客户端的第一请求之前,从第二客户端接收对于动态产生对象的第二请求;以及用于从始发服务器接收对于第一请求的响应,该响应包括动态产生的对象。该系统还包括用于响应第一请求向第一客户端传送动态产生的对象,并响应第二请求向第二客户端传送动态产生的对象的装置。。

[0049] 在本发明的一个实施例中,该网络设备进一步包括用于从多个客户端接收对于动态产生对象的多个请求,以及响应每个客户端的请求,向多个客户端传送动态产生对象的装置。在另一实施例中,该网络设备包括用于在完成将动态产生的对象传送到第二客户端之前,从第三客户端接收对于动态产生对象的第三请求,以及用于响应第三请求,向第三客

户端传送动态产生的对象的装置。在其他实施例中,该网络设备具有几乎同时接收第一请求和第二请求的装置。在某些实施例中,例如该网络设备排队第二请求,而不是将第二请求传送到服务器。

[0050] 在一个实施例中,本发明的网络设备在向第一客户端和第二客户端传送动态产生的对象之后,从缓存器中刷新该动态产生的对象。在某些实施例中,该动态产生的对象被识别为不可缓存的。在其他实施例中,该网络设备包括用于通过与第二请求相关的对象决定因素确定第二请求是用于第一请求的动态产生的对象的装置。在另一实施例中,该网络设备包括用于修改响应以包括实体标签报头或缓存控制报头之一并将修改的响应传送到第一客户端或第二客户端之一的装置。该网络设备可以是在客户端和始发服务器之间进行通信的装置或计算设备。

[0051] 另一方面,本发明涉及一种由缓存器修改来自服务器的响应以将响应中到客户端的动态产生对象识别为可缓存的方法和系统,该服务器没有将动态产生的对象识别为可缓存的。在某些实施例中,诸如处理对对象的 HTTP 请求和响应的实施例,本发明的技术在响应中插入实体标签或“etag”以对对象提供缓存控制,假定没有来自始发服务器的实体标签和 / 或缓存控制信息。本发明的该技术通过在向客户端的响应中插入诸如对象的实体标签和缓存控制信息之类的信息以响应于客户端使缓存器能够检查在随后请求中的命中而增加缓存器命中率。

[0052] 一方面,本发明涉及一种使用实体标签来缓存没被标识为可缓存的动态产生对象的方法,该方法包括由缓存器接收客户端对于来自始发服务器的动态产生对象的请求的响应。该动态产生对象被识别为不可缓存的。该方法包括由缓存器对于缓存器中保存的动态产生对象产生实体标签并修改响应以包括产生的实体标签。该方法进一步包括由缓存器向客户端传送识别动态产生对象为可缓存的经过修改的响应。

[0053] 在本发明的方法的一个实施例中,从始发服务器接收的响应不包括实体标签。在某些实施例中,缓存器修改响应以识别客户端缓存动态产生对象的持续期间。在其他实施例中,缓存器修改响应以向客户端指出存储响应并在检查动态产生对象是否改变之后向缓存器或始发服务器之一提供响应。在一个实施例中,来自始发服务器的响应不具有动态产生对象的缓存控制信息,并且缓存器修改传送到客户端的响应以包括缓存控制信息。

[0054] 在某些实施例中,本发明的方法也包括由缓存器为实体标签产生数字。缓存器可以将所产生的实体标签和动态产生的对象一起存储。在另一实施例中,缓存器可以将缓存控制信息和动态产生的对象一起存储。在一个实施例中,缓存器提供具有所产生的实体标签和缓存控制信息的缓存的动态产生对象。

[0055] 在本发明的某些实施例中,缓存器在装置、网络设备或计算设备上操作。在其他实施例中,请求或响应包括超文本标记语言(HTML)。在另一实施例中,缓存器修改响应以包括识别所产生的实体的 etag 报头,而在另一实施例中,缓存器修改响应以包括缓存控制报头。在某些实施例中,该方法包括由缓存器从客户端接收具有 If-None-Match 报头的第二请求,并响应第二请求,将与 If-None-Match 报头相关的 etag 值与在缓存器中和动态产生对象一起保存的实体标签相比较。在另一实施例中,如果 etag 值匹配实体标签,则缓存器向客户端传送对于第二请求来说没有修改的响应;而在另一实施例中,如果 etag 值不匹配实体标签,则缓存器向客户端传送具有第二实体标签的更新的动态产生对象。

[0056] 在下文的附图和说明中阐述了本发明的各个实施例的细节。

## 附图说明

[0057] 这里包含的并且形成说明书的一部分的相应附图图解了本发明，并且与说明书一起，进一步用来解释本发明的原理并且能使相关领域的技术人员实现并且使用本发明。

[0058] 图 1 是示出了其中可以实施本发明的实施例的示例网络环境的框图；

[0059] 图 2 是示出根据本发明实施例执行集成缓存的装置的示例结构的框图；

[0060] 图 3A 是用于使设备操作与分组处理和分组处理计时器相集成本发明方法实施例中采取的步骤的流程图；

[0061] 图 3B 是用于根据图 3A 实施无效颗粒度技术的本发明的方法实施例中采用的步骤的流程图；

[0062] 图 4A 是在使用无效命令以无效陈旧对象(stale object)的本发明的方法实施例中采用的步骤的流程图；

[0063] 图 4B 是在包括无效对象组的本发明的方法实施例中采用的步骤的流程图；

[0064] 图 4C 是在其中为了对象决定因素解析客户端请求的本发明的方法实施例中采用的步骤的流程图；

[0065] 图 4D 是在包括使用对象决定因素而无效对象组的本发明的方法实施例中采用的步骤的流程图；

[0066] 图 5 是在提供闪速缓存技术的本发明的方法实施例中采用的步骤的流程图；

[0067] 图 6 是在提供闪速群控制技术的本发明的方法实施例中采用的步骤的流程图；

[0068] 图 7A 和 7B 是在为对象提供实体标签和缓存控制的本发明的方法实施例中采用的步骤的流程图；

[0069] 图 8A 和 8B 是用于实施本发明的示例性实施例的计算设备的实施例的框图。

[0070] 从下文结合附图所陈述的详细描述中本发明的特征和优势将变得更加明显，其中等同的参考特征表示对应的元件。在附图中，相似的附图标记一般表示等同的、功能类似的和 / 或结构类似的元件。

## 具体实施方式

[0071] A. 示例网络环境

[0072] 图 1 示出了其中可以实施本发明的实施例的示例网络环境 100。如图 1 所示，示例网络环境 100 包括多个客户端 102a-102n，多个服务器 106a-106n，以及也被称为缓存装置、设备或缓存器的装置 104。服务器 106a-106n 创始并管理诸如对象数据库或关系数据库之类的数据库，该数据库向客户端 102a-102n 提供请求内容。为此，服务器 106a-106n 在此经常被称为“始发服务器”，这是因为它们通常，但不必然，始发形成请求内容的对象。每个客户端 102a-102n 和服务器 106a-106n 可以是任何类型和形式的计算设备，诸如下文结合图 8A 和 8B 更详细描述的计算设备 800。例如，任何客户端 102a-102n 可以是移动计算设备，诸如电信设备，例如蜂窝电话或个人数字助理，或除了任何类型的桌上型计算机之外的膝上型计算机或笔记本式计算机。

[0073] 每个客户端 102a-102n 经由公用数据通信网络 108 而可通信地耦合到装置 104，而

装置 104 经由专用数据通信网络 110 而可通信地耦合到服务器 106a-106n。在一个实施例中,公用数据通信网络 108 包括因特网以及专用数据通信网络 110 包括企业网。公用数据通信网络 108 和专用数据通信网络 110 可以是任何类型和形式的网络、公用的、专用的或其他的,以及在某些情况下,可以是相同网络。

[0074] 虽然图 1 示出了客户端 102a-102n 和服务器 106a-106n 之间的网络 108 和 110,但客户端 102a-102n 和服务器 106a-106n 可以在相同网络 108 或 110 上。网络 108 和 110 可以是相同类型的网络或不同类型的网络。网络 108 和 / 或网络 110 可以是诸如公司内部网或城域网(MAN)之类的局域网(LAN)或诸如因特网或万维网之类的广域网(WAN)。网络 108 和 / 或 110 可以是任何类型和 / 或形式的网络并且可以包括任何如下网络:点对点网络、广播网、广域网、局域网、电信网、数据通信网、计算机网、ATM(异步传输模式)网、SONET(同步光纤网)网、SDH(同步数字系列)网、无线网和有线网。网络 108 和 / 或 110 的拓扑结构可以是总线、星形或环形网络拓扑结构。网络 108 和 / 或 110 和网络拓扑结构可以是本领域的普通技术人员已知的能够支持本文描述的本发明操作的任何网络或网络拓扑结构。

[0075] 如图 1 所示,装置 104 被示为在公用数据通信网 108 和专用数据通信网 110 之间。在其他实施例中,装置 104 可以位于公用数据通信网 108 或专用数据通信网 110 之上。在其他实施例中,装置 104 可以是在与客户端 102a-102n 相同或不同的网络 108、110 上任何单个客户端 102a-102n 或任何单个服务器 106a-106n 的组成部分。因此,装置 104 可以位于在客户端 102a-102n 和服务器 106a-106n 之间的网络或网络通信路径中的任何点上。

[0076] 根据本发明实施例,装置 104 包括缓存管理逻辑并且还包含或访问它用于实施缓冲存储器的存储介质。利用这些特性,装置 104 监视客户端 102a-102n 对任何服务器 106a-106n 所做出的对象请求。响应于这些对象请求,从服务器 106a-106n 返回的对象被装置 104 存储在缓冲存储器中。从任何客户端 102a-102n 对相同对象的随后请求由装置 104 截取,该装置 104 试图从缓存器发送对象而不是将请求传递到服务器 106a-106n。这提供了双重优势,即减少响应来自客户端 102a-102n 的请求所需时间以及减少了支持服务器 106a-106n 的基础设施上的负荷。

[0077] 总之,图 1 所示的网络环境 100 只通过示例形式示出而不是限制性的。根据这里提供的教导,本领域技术人员将容易理解的是,本发明可以在其中一个或多个网络的节点之间传送对象请求和响应的任何网络环境中实施。

#### [0078] B. 示例装置或设备结构

[0079] 如这里更详细的描述,在本发明的实施例中,装置 104 使操作系统的内核层上的缓存功能与一个或多个其他处理任务相集成,该其他处理任务包括(但不限于)解密、解压缩、或认证和 / 或授权。这样的实施在于 2005 年 6 月 29 日提交的题目为“Method and Device for Performing Integrated Caching in a Data Communication Network”的申请号为 11/169,002 的共同拥有的同时待审美国专利申请中示出,其在此被引入以供参考。这里根据图 2 描述该示例结构,但本发明并不限于此,并且其他结构也可以用于实施这里所述的本发明的操作。

[0080] 图 2 示出了装置 104 的示例结构 200。如上所述,结构 200 仅通过示例的形式示出而不是限制性的。如图 2 所示,示例结构 200 包括硬件层 206 和划分为用户空间 202 和内核空间 204 的软件层。

[0081] 硬件层 206 提供硬件元件,在该硬件元件上执行在内核空间 204 和用户空间 202 内的程序和服务。硬件层 206 也提供允许内核空间 204 和用户空间 202 内的程序和服务相对于装置 104 内部地和外部地传送数据的结构和元件。如图 2 所示,硬件层 206 包括用于执行软件程序和服务的处理单元 262、用于存储软件 and 数据的存储器 264、用于经过网络传送和接收数据的网络端口 266、以及用于执行与处理经过网络传送和接收的数据的安全套接层相关的功能的加密处理器 260。在某些实施例中,中央处理单元 262 可以在单个处理器中执行加密处理器 260 的功能。另外,硬件层 206 可以包括对于每个处理单元 262 的多个处理器和加密处理器 260。虽然装置 104 的硬件层 206 一般使用加密处理器 260 来示出,但是处理器 260 可以是用于执行与诸如安全套接层(SSL)或传输层安全(TLS)协议之类的任何加密协议相关的功能的处理器。在某些实施例中,处理器 260 可以是通用处理器(GPP),而在其他实施例中,处理器 260 可以具有可执行指令以执行任何安全相关协议的处理。

[0082] 虽然使用图 2 中的特定元件示出了装置 104 的硬件层 206,但是装置 104 的硬件部分或元件可以包括计算设备的任何类型或形式的元件、硬件或软件,诸如这里进一步结合图 8A 和 8B 示出和讨论的计算设备 800。在某些实施例中,装置 104 可以包括服务器、网关、路由器、转换器、电桥或其他类型的计算或网络设备,并且具有与之相关的任何硬件和/或软件元件。

[0083] 装置 104 的操作系统分配、管理可用系统存储器或另外将可用系统存储器隔离为内核空间 204 和用户空间 204。在示例软件结构 200 中,操作系统可以是任何类型和/或形式的 Unix 操作系统,但本发明并不限制于此。因此,装置 104 可以运行任何操作系统,诸如任何版本的 Microsoft® Windows 操作系统、不同版本的 Unix 和 Linux 操作系统、任何版本的 Macintosh 计算机的 Mac OS®,任何嵌入式操作系统、任何网络操作系统、任何实时操作系统、任何开源操作系统、任何私有操作系统、用于移动计算设备或网络设备的任何操作系统、或者能够在装置 104 上运行并执行这里所述操作的任何其他操作系统。

[0084] 保留内核空间 204 用于运行内核 230,包括任何设备驱动器、内核扩展或其他涉及内核的软件。如本领域技术人员所知,内核 230 是操作系统的核心,并且提供对资源和装置 104 的相关硬件元件的访问、控制和管理。根据本发明实施例,内核空间 204 也包括协同缓存管理器 232 工作的多个网络服务或处理,经常被称为集成缓存器,它的优势在这里将进一步详细的描述。另外,内核 230 的实施例将取决于设备 104 安装、配置或另外使用的操作系统的实施方式。

[0085] 在一个实施例中,设备 104 包括一个网络堆栈 267,诸如基于 TCP/IP 的堆栈,用于与客户端 102a-102b 和/或服务器 106a-106n 进行通信。在一个实施例中,网络堆栈 267 用于与诸如网络 108 的第一网络和第二网络 110 进行通信。在某些实施例中,设备 104 终止诸如客户端 102a-102n 的 TCP 连接的第一传输层连接,并建立由客户端 102a-102n 使用的到服务器 106a-106n 的第二传输层连接,例如在装置 104 和服务器 106a-106n 上终止第二传输层连接。可以借助于单个网络堆栈 267 建立第一和第二传输层连接。在其他实施例中,设备 104 包括多个网络堆栈,例如 267 和 267',以及在一个网络堆栈 267 上建立或终止第一传输层连接,以及在第二网络堆栈 267' 上建立或终止第二传输层连接。例如,一个网络堆栈可以用于接收和传送第一网络上的网络分组,以及另一网络堆栈用于接收和传送第二网络上的网络分组。在一个实施例中,网络堆栈 267 包括用于排队一个或多个网络分组

以由装置 104 传送的缓冲器 243。

[0086] 如图 2 所示,内核空间 204 包括缓存管理器 232、高速层 2-7 集成分组引擎 240、加密引擎 234、策略引擎 236 和多协议压缩逻辑 238。在内核空间 204 或内核模式中而不是用户空间 202 内运行这些元件或处理 232、240、234、236 和 238 单独或结合地改进这些元件中的每个的性能。内核操作表示这些元件或处理 232、240、234、236 和 238 在设备 104 的操作系统的内核地址空间内运行。例如,通过将加密和解密操作移至内核而以内核模式运行加密引擎 234 改进加密性能,因此减少以内核模式的存储器空间或内核线程和以用户模式的存储器空间或线程之间转换的次数。例如,以内核模式获取的数据不需要传递或拷贝到以用户模式运行的处理或线程,诸如从内核级数据结构到用户级数据结构。另一方面,也减少了内核模式和用户模式之间的上下文切换(context switch)的数量。另外,可以在内核空间 204 内更有效地执行任何元件或处理 232、240、235、236 和 238 之间的同步或通信。

[0087] 在某些实施例中,元件 232、240、234、236 和 238 的任何部分可以在内核空间 204 内运行或操作,而这些元件 232、240、234、236 和 238 的其他部分可以在用户空间 202 中运行或操作。在一个实施例中,本发明使用内核级数据结构,该内核级数据结构提供访问一个或多个网络分组的任何部分,例如,包括来自客户端 102a-102n 的请求或来自服务器 106a-106n 的响应的网络分组。在某些实施例中,由分组引擎 240 经由到网络堆栈 267 的传输层驱动器接口或滤波器而获取内核级数据结构。内核级数据结构包括经由涉及网络堆栈 267 的内核空间 204、由网络堆栈 267 接收或传送的网络业务或分组可访问的任何接口和/或数据。在其他实施例中,可以由元件或处理 232、240、234、236 和 238 中的任何一个使用该内核级数据结构以执行元件或处理的所需操作。在一个实施例中,当使用内核级数据结构时元件 232、240、234、236 和 238 以内核模式 204 运行,而在另一实施例中,当使用内核级数据结构时元件 232、240、234、236 和 238 以用户模式运行。在某些实施例中,内核级数据结构能被拷贝或传递到第二内核级数据结构或任何所需的用户级数据结构。

[0088] 缓存管理器 232 包括软件、硬件或软件和硬件的任何组合以提供对任何类型和形式的内容的缓存访问、控制和管理,诸如由始发服务器 106a-106n 提供的对象或动态产生的对象。由缓存管理器 232 处理并存储的数据、对象或内容包括任何格式的数据,诸如标记语言,或者借助于任何协议传送的数据。在某些实施例中,缓存管理器 232 复制其他地方存储的原始数据或之前计算、产生或传送的数据,其中相对于读取缓冲存储器元件而言原始数据需要更长的访问时间以提取、计算或另外获取。一旦在缓冲存储器元件中存储数据,则通过访问缓存的拷贝而不是再提取或再计算原始数据而获得将来的使用,因此减少了访问时间。在某些实施例中,缓冲存储器元件可以包括设备 104 的存储器 264 中的数据对象。在其他实施例中,缓冲存储器元件可以包括具有比存储器 264 更快访问时间的存储器。在另一实施例中,缓冲存储器元件包括任何类型和形式的设备 104 的存储元件,诸如硬盘的一部分。在某些实施例中,处理单元 262 可以提供缓冲存储器以供本发明的缓存管理器 232 使用。在其他实施例中,缓存管理器 232 使用存储器、寄存器或处理单元的任何部分或组合用于缓存数据、对象和其他内容。

[0089] 而且,本发明的缓存管理器 232 包括任何逻辑、功能、规则或操作以执行这里所述本发明的技术的任何实施例。例如,缓存管理器 232 包括逻辑或功能以根据无效时间段的届满或一旦从客户端 102a-102n 或服务器 106a-106n 接收无效命令就无效对象。在某些实

施例中,缓存管理器 232 可以像在内核空间 204 中执行的程序、服务、处理或任务一样操作,而在其他实施例中,可以像用户空间 202 中执行的程序、服务、处理或任务一样操作。在一个实施例中,缓存管理器 232 的第一部分在用户空间 202 内执行而第二部分在内核空间 204 内执行。在某些实施例中,缓存管理器 232 可以包括任何类型的通用处理器(GPP)、或诸如现场可编程门阵列(FPGA)、可编程逻辑设备(PLD)或特定用途集成电路(ASIC)之类的任何其他类型的集成电路。

[0090] 策略引擎 236 包括例如智能统计引擎或一个(或多个)其他可编程应用。在一个实施例中,策略引擎 236 提供配置机制以允许用户识别、指定、定义或配置缓存策略。在某些实施例中,策略引擎 236 也访问存储器以支持诸如查询表或散列表之类的数据结构以启用用户选择的缓存策略决定。在其他实施例中,策略引擎 236 包括任何逻辑、规则、功能或操作以确定并且除了对安全、网络通信量、网络访问、压缩或由装置 104 执行的任何其他功能或操作的访问、控制和管理之外提供对由装置 104 缓存的对象、数据或内容的访问、控制和管理。这里将进一步描述特定缓存策略的其他示例。

[0091] 加密引擎 234 包括任何逻辑、商业规则、功能或操作,用于管理诸如 SSL 或 TLS 之类的任何安全相关协议以及与其相关的任何功能的处理。例如,加密引擎 234 加密并解密经由装置 104 传送的网络分组、或其任何部分。加密引擎 234 代表客户端 102a-102n、服务器 106a-106n 或装置 104 还布置或建立 SSL 或 TLS 连接。因此,加密引擎 234 提供 SSL 处理的卸载和加速。在一个实施例中,加密引擎 234 使用隧道协议以提供客户端 102a-102n 和服务器 106a-106n 之间的虚拟专用网络。在某些实施例中,加密引擎 234 与加密处理器 260 进行通信。在其他实施例中,加密引擎 234 包括在加密处理器 260 上运行的可执行指令。

[0092] 多协议压缩引擎 238 包括任何逻辑、商业规则、功能或操作,用于压缩网络分组的一个或多个协议,诸如由设备 104 的网络堆栈 267 所用的任何协议。在一个实施例中,多协议压缩引擎 238 在客户端 102a-102n 和服务器 106a-106n 之间双向地压缩任何基于 TCP/IP 的协议,包括消息应用程序编程接口(MAPI)(电子邮件)、文件传输协议(FTP)、超文本传输协议(HTTP)、公共因特网文件系统(CIFS)协议(文件传输)、独立计算体系结构(ICA)协议、远程桌上协议(RDP)、无线应用协议(WAP)、移动 IP 协议以及基于 IP 的语音(VoIP)协议。在其他实施例中,多协议压缩引擎 238 提供基于超文本标记语言(HTML)协议的压缩并在某些实施例中,提供诸如可扩展标记语言(XML)的任何标记语言的压缩。在一个实施例中,多协议压缩引擎 238 提供任何高性能协议的压缩,诸如为装置 104 设计用于装置 104 通信的任何协议。在另一实施例中,多协议压缩引擎 238 使用修改的传输控制协议压缩任何负载或任何通信,所述协议诸如交易 TCP (T/TCP)、具有选择确认的 TCP (TCP-SACK)、具有大窗口的 TCP (TCP-LW)、诸如 TCP-Vegas 协议等的拥塞预测协议以及 TCP 欺骗协议。

[0093] 因此,本发明的多协议压缩引擎 238 借助于例如 Microsoft Outlook 的桌面客户端和非 Web 瘦客户端,诸如像 Oracle SAP 和 Siebel 等的流行企业应用发布的任何客户端以及诸如 Pocket PC 的移动客户端而加速用户访问应用的性能。在某些实施例中,通过以内核模式 204 执行的并与分组处理引擎 240 集成来访问网络堆栈 267 的多协议压缩引擎 238 能够压缩诸如 TCP/IP 协议所执行的任何协议,诸如任何应用层协议。

[0094] 通常也被称为分组处理引擎或分组引擎的高速层 2-7 集成分组引擎 240 负责管理借助于网络端口 266 由装置 104 接收并传送的分组的内核级处理。高速层 2-7 集成分组引



擎 240 包括缓冲器,用于在诸如接收网络分组或传送网络分组的处理期间排队一个或多个网络分组。另外,高速层 2-7 集成分组引擎 240 与一个或多个网络堆栈 267 进行通信以借助于网络端口 266 发送并接收网络分组。高速层 2-7 集成分组引擎 240 结合加密引擎 234、缓存管理器 232、策略引擎 236 和多协议压缩逻辑 238 一起工作。具体地,加密引擎 234 被配置以执行分组的 SSL 处理,策略引擎 236 被配置以执行涉及业务管理的功能,诸如请求级内容切换和请求级缓存重定向,以及多协议压缩逻辑 238 被配置以执行涉及压缩和解压缩数据的功能。

[0095] 高速层 2-7 集成分组引擎 240 包括分组处理计时器 242。在一个实施例中,分组处理计时器 242 提供一个或多个时间间隔以触发输入的(即接收的)或输出的(即传送的)网络分组的处理。在某些实施例中,高速层 2-7 集成分组引擎 240 响应计时器 242 而处理网络分组。分组处理计时器 242 向分组引擎 240 提供任何类型和形式的信号以通知、触发或传送涉及时间的事件、间隔或发生的事情(occurrence)。在某些实施例中,分组处理计时器 242 以毫秒级操作,例如 100ms、50ms 或 25ms。例如,在某些实施例中,分组处理计时器 242 提供时间间隔或按其他方式使得网络分组以 10ms 时间间隔由高速层 2-7 集成分组引擎 240 进行处理,而在其他实施例中,以 5ms 时间间隔,并在另外的其他实施例中,缩短至 3、2 或 1ms 时间间隔。高速层 2-7 集成分组引擎 240 在操作期间与加密引擎 234、缓存管理器 232、策略引擎 236 和多协议压缩引擎 238 进行接口、集成或通信。因此,响应于分组处理计时器 242 和 / 或分组引擎 240 而执行加密引擎 234、缓存管理器 232、策略引擎 236 和多协议压缩逻辑 238 的任何逻辑、功能或操作。因此,可以在通过分组处理计时器 242 提供的时间间隔的颗粒度(granularity)(例如,在小于或等于 10ms 的时间间隔)上执行加密引擎 234、缓存管理器 232、策略引擎 236 和多协议压缩逻辑 238 的任何逻辑、功能或操作。例如,在一个实施例中,缓存管理器 232 响应于高速层 2-7 集成分组引擎 240 和 / 或分组处理计时器 242 而执行任何缓存对象的无效。在另一实施例中,可以将缓存对象的期限时间或无效时间设置为与分组处理计时器 242 的时间间隔相同的颗粒度级,诸如在每 10ms。

[0096] 与内核空间 204 相反,用户空间 202 是用户模式应用或另外以用户模式运行的程序所用的存储器区域或部分操作系统。用户模式应用不能直接访问内核空间 204 并使用服务调用以便访问内核服务。如图 2 所示,应用 104 的用户空间 202 包括图形用户界面(GUI) 210、命令行界面(CLI) 212、外壳服务(shell service) 214、可用状态监控(health monitoring)程序 216 和守护(daemon)服务 218。GUI210 和 CLI212 提供一个装置,通过该装置系统管理者或其他用户可以交互并控制装置 104 的操作,诸如借助于装置 104 的操作系统并或是用户空间 202 或内核空间 204。GUI210 可以是任何类型和形式的图形用户界面并且可以借助文本、图形或另外通过任何类型的程序或应用(例如,浏览器)表现。CLI212 可以是任何类型和形式的命令行或基于文本的界面,诸如由操作系统提供的命令行等。例如,CLI212 包括一个外壳,其是使用户能够与操作系统进行交互的工具。在某些实施例中,借助 bash、csh、tsh 或 ksh 类型外壳提供 CLI212。外壳服务 214 包括程序、服务、任务、处理或可执行指令以支持用户借助 GUI210 和 / 或 CLI212 与装置 104 或操作系统进行交互。

[0097] 可用状态监控程序 216 用于监控、检查、报告并确保网络系统正适当地运行以及用户通过网络正在接收请求的内容。可用状态监控程序 216 包括一个或多个程序、服务、任务、处理或可执行指令以提供逻辑、规则、功能或操作以用于监控装置 104 的任何活动。在

某些实施例中,可用状态监控程序 216 截取并检查经过装置 104 传递的任何网络通信量。在其他实施例中,可用状态监控程序 216 通过任何合适的装置和 / 或机制与一个或多个如下装置对接:加密引擎 234、缓存管理器 232、策略引擎 236、多协议压缩逻辑 238、分组引擎 240、守护服务 218、以及外壳服务 214。因此,可用状态监控程序 216 能调用任何应用程序接口(API)以确定装置 104 的任何部分的状态、状况或健康。例如,可用状态监控程序 216 在周期性的基础上 ping 或发送状况查询以检测程序、处理、服务或任务是否有效并正在运行。在另一示例中,可用状态监控程序 216 检查程序、处理、服务或任务所提供的任何状况、错误或历史日志以确定装置 104 的任何部分的任何情况、状况或错误。

[0098] 守护服务 218 是持续运行或在后台中的并且处理装置 104 所接收的周期服务请求的程序。在某些实施例中,守护服务向其他程序或处理转送请求,诸如合适时的另一守护服务 218。如本领域技术人员所知晓的,守护服务 218 自动地运行以持续地或周期地执行诸如网络控制的系统广泛功能,或执行任何所需任务。在某些实施例中,一个或多个守护服务 218 在用户空间 202 内运行,而在其他实施例中,一个或多个守护服务 218 在内核空间内运行。

#### [0099] C. 缓存动态产生的对象

[0100] 诸如一个或多个动态产生的对象之类的动态内容可以由服务器和 / 或后端数据库(未示出)产生,所述服务器被称作应用或始发服务器 106a-106n,该后端数据库本地或远程地处理来自一个或多个客户端 102a-102n 的对象请求,如图 1 所示。当这些应用或数据库处理数据(该数据包括涉及从客户端接收的输入的数据)时,由这些数据库和应用提供的响应对象将改变。在始发服务器中由这些应用或数据库产生的先前对象将不再“新鲜”并因此不再需要由缓存器保存。例如,给定相同的输入集合,第一实例的动态产生的对象不同于第二实例的动态产生的对象。在另一示例中,使用不同的输入集合而动态产生相同对象,以致于不同于对象的第二实例而产生对象的第一实例。

[0101] 为了获得改进的网络性能,装置 104 被设计和配置以解决在通过各种方法缓存动态产生的内容中出现的各种问题,如下文的详细描述。在这里所述的本发明的某些实施例中,装置 104 结合了一种或多种技术的集合用于更高效快捷地无效缓存器中所存储的动态产生的内容。而且,该装置结合用于执行控制并缓存闪速群的技术。缓冲存储器一般存储对对象请求的每个响应,只要这样的响应没有被标记为不可缓存。如这里所述,高效缓存动态产生的内容需要这样的技术,该技术能够及时无效在始发服务器已经历改变的缓冲存储器中的对象。及时无效允许缓存器避免提供“陈旧”内容 - 特别涉及动态产生内容的任务,特别是在内容改变无规律发生的时候。下文将阐述确保及时无效动态产生的内容的若干技术。

#### [0102] 1. 集成功能

[0103] 一方面,本发明涉及一种响应于分组处理计时器 242 使缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩逻辑 238 的功能、逻辑或操作与高速层 2-7 集成分组引擎 240 的分组处理操作相集成的技术。例如,可以在用于分组处理操作的分组处理计时器 242 的时间间隔内执行缓存管理器 232 的操作,例如在接收或传送网络分组时。在一个实施例中,通过与分组处理操作相集成和 / 或使用分组处理计时器,本发明的缓存管理器 232 可以利用低至非常小时间间隔的期限时间来缓存对象,这将在下文更详细的描述。在其他

实施例中,响应分组处理计时器 242 的缓存管理器 232 也可以接收无效命令以在缓存对象非常短的时间段内无效对象。

[0104] 图 3A 所示的方法 300 示出了用于请求缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩引擎 238 以在处理或结合时间间隔以由高速层 2-7 集成分组引擎或分组处理引擎 240 处理网络分组期间执行操作的本发明的技术的一个实施例。简单概括,在方法 300 的步骤 310,设备 104 接收网络分组或被请求传送网络分组。在步骤 315,响应于分组处理计时器 242,设备 104 请求分组处理引擎 240 来处理网络分组。作为分组处理操作的一部分或结合分组处理操作,在步骤 320,分组处理引擎 240 请求缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩引擎 238 对缓存对象执行操作。在步骤 325,缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩引擎 238 执行请求的操作,其包括这里所述发明的技术的任何一个或组合。在一个实施例中,缓存管理器 232 确定缓存对象的无效,并标记缓存对象为无效。在某些实施例中,缓存管理器 232 响应于分组处理引擎 240 的请求而刷新无效对象。当缓存管理器 232 响应于分组处理计时器 242 而正执行这些操作时,无效对象可以在毫秒级的时间段内发生以及对象具有在分组处理计时器 242 所提供的时间间隔的级别的期限时间,诸如 10ms 等。

[0105] 在本发明的方法 300 的更进一步的描述中,在步骤 310,装置 104 接收一个或多个网络分组和 / 或传送一个或多个网络分组。在某些实施例中,装置 104 请求经过网络 108 或网络 110 传送一个或多个网络分组。在另一实施例中,装置 104 在一个端口 266 上接收网络分组并在相同端口 266 或不同端口 266' 上传送网络分组。在某些实施例中,装置 104 的分组引擎 240 传送或请求传送一个或多个网络分组。在一个实施例中,装置 104 在第一网络 108 上接收或传送分组,而在另一实施例中,装置 104 在第二网络 110 上接收或传送分组。在其他实施例中,装置 104 在相同网络上接收和传送分组。在某些实施例中,装置 104 向一个或多个客户端 102a-102n 接收和 / 或传送网络分组。在其他实施例中,装置 104 向一个或多个服务器 106a-106n 接收和 / 或发送网络分组。

[0106] 在步骤 315,一旦在设备 104 的网络端口 266 上接收网络分组或一旦从设备 104 请求传送网络分组,或者一旦接收和 / 或传送一个或多个网络分组的任何组合,则设备 104 请求或触发分组处理引擎 240 的分组处理操作。在某些实施例中,通过分组处理计时器 242 提供的信号触发分组处理引擎 240 的分组处理操作。在一个实施例中,分组处理计时器 242 提供与接收和 / 或传送一个或多个网络分组有关的中断驱动或事件驱动的计时器功能。在某些实施例中,由通过设备 104 接收和 / 或发送网络分组的速率或由处理每个分组或一批分组的速率而驱动分组处理计时器 242。因此,在一个或多个分组处理操作的每个集合之后触发并复位分组处理计时器 242。在另一实施例中,分组处理计时器 242 提供时间间隔(相同或可变的时间间隔)以触发、唤醒或用信号通知分组处理引擎 240 以执行诸如处理接收的分组或传送提交的分组之类的功能或操作。如上文结合图 2 的设备 104 所讨论的,分组处理计时器 242 以毫秒级进行操作,诸如引起时间间隔或在 10ms 或更少的时间间隔内触发分组处理操作。能以各种方式提供并在分组处理引擎 240 的分组处理操作的操作中使用本发明的分组处理计时器的颗粒度计时器功能。

[0107] 在本发明的方法 300 的步骤 320,分组处理引擎 240 请求缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩引擎 238 中的一个或多个以执行操作。在一个实施

例中,分组处理引擎 240 或分组处理计时器 242 产生信号或用信号通知缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩引擎 238 中的一个或多个。在网络分组或一个或多个分组的分组处理操作之前、期间或之后的任何时刻,分组处理引擎 240 请求或用信号通知操作。在一个实施例中,一旦触发分组处理计时器 242 或分组处理计时器 242 提供的时间间隔届满,并在对网络分组执行分组处理操作之前,分组处理引擎 240 进行请求。在另一实施例中,在执行一个或多个分组处理操作期间,分组处理引擎 240 进行请求。例如,在执行操作期间,诸如在功能调用内,分组处理引擎 240 对缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩引擎 238 之一进行应用程序接口(API)调用。在其他实施例中,一旦完成网络分组处理的操作则分组处理引擎 240 进行请求。

[0108] 在步骤 325,由缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩引擎 238 中的一个或多个执行请求的操作。在某些实施例中,诸如经由内核应用程序接口(API),可以请求执行通过内核 204 所提供的任何功能或操作。因此,结合借助分组处理计时器 232 的时间或分组处理的时间间隔执行设备 104 的任何功能。在某些实施例中,同步并结合分组处理引擎 240 的分组处理操作执行请求的操作。例如,一旦完成或响应请求的操作,分组处理操作就等待并持续。在其他实施例中,与分组处理操作非同步地执行请求的操作。例如,分组处理引擎 240 发送请求以执行操作但不妨碍或等待从操作接收响应。如下文将结合图 3B 所示的本发明的方法 350 更详细讨论的,分组处理引擎 240 请求缓存管理器 232 以执行任何缓存管理功能,诸如检查对象的期限或无效、标记对象为无效或刷新无效或届满的对象。

[0109] 在某些实施例中,在步骤 320 分组处理引擎 240 发送多个请求,诸如向缓存管理器 232 发送第一请求并向加密引擎 234 发送第二请求。在其他实施例中,在步骤 320,分组处理引擎 240 诸如经由内核 230 向设备 104 的目标元件发送包括将由设备 104 分发的多个请求的单独请求。在一个实施例中,随后将请求进行彼此传送。在另一实施例中,请求取决于先前请求的状态、结果、成功或完成。例如,到策略引擎 236 的第一请求用于确定处理来自另一设备或与网络分组相关的用户的网络分组的策略。根据策略引擎 236 的策略,取决于第一请求的结果进行或不进行到缓存器的第二请求。使用设备 104 的内核空间 204 中与分组处理引擎 240 集成的缓存管理器 232、策略引擎 236、加密引擎 234 和 / 或多协议压缩引擎 238,存在由分组处理操作触发并与分组处理操作集成的这里所述的设备 104 的各种操作。

## [0110] 2. 无效颗粒度

[0111] 另一方面,本发明涉及并包含将缓存器保存的对象的期限时间配置为微小的颗粒度时间间隔的能力,诸如由分组处理计时器提供的时间间隔的颗粒度。该特性被称为“无效颗粒度”。因此,在一个实施例中,本发明可以利用低至非常小时间间隔的期限时间来缓存对象。在其他实施例中,响应分组处理计时器的缓存管理器也可以接收无效命令以在缓存对象的非常短的时间段内无效对象。通过在期限内提供该微小的颗粒度,本发明的缓存器可以缓存并提供经常变化的对象,该对象经常甚至 1 秒内数次变化。一种技术是调节本发明的设备所用的分组处理计时器,这样能够以毫秒级的时间增量操作以允许无效或期限颗粒度低至 10ms 或更少。相对比于本发明,常规的缓存器一般不能设置或具有小于一秒的期限或无效颗粒度。

[0112] 现在参考图 3B, 描述了本发明的方法 350 的实施例, 用于响应于分组处理计时器 242 和 / 或分组处理引擎 240 而无效缓存的对象或使缓存的对象届满。因此, 在本发明的某些实施例中, 可以以毫秒级 (诸如 10ms 或更少) 无效缓存的对象或使缓存的对象届满。总之, 在方法 350 的步骤 355, 缓存管理器 232 响应于分组处理计时器 242 通过分组处理引擎 240 而接收信号或请求以执行操作。在步骤 360, 缓存管理器 232 确定诸如动态产生的对象之类的缓存对象是否无效或届满。在步骤 365, 如果对象是无效的, 则缓存管理器 232 将对象标记为无效, 并在步骤 370, 从缓存管理器 232 刷新无效对象。

[0113] 在步骤 355 的进一步描述中, 在某些实施例中, 缓存管理器 232 会被信号通知或请求在网络分组处理期间的任何时间点执行涉及缓存器的操作。在一个实施例中, 在步骤 355, 在对由设备 104 接收或传送的网络分组进行处理之前缓存管理器 232 接收操作请求。在另一实施例中, 一旦完成网络分组的处理, 缓存管理器 232 就接收操作请求。例如, 分组处理引擎 240 完成网络分组的处理, 并在等待计时器 242 的下个时间间隔之前或在处理下个分组之前, 请求缓存器执行操作。在其他实施例中, 在操作分组处理期间, 分组处理引擎 240 将操作请求传送到缓存管理器 232。在另一实施例中, 缓存管理器 232 诸如从分组处理引擎 240 或分组处理计时器 242 接收信号以触发缓存管理器 232 执行操作。在某些实施例中, 信号表示无效缓存的对象或使缓存对象的期限届满。

[0114] 在某些实施例中, 缓存管理器 232 从缓存管理器 232 外部的实体接收请求以执行缓存操作, 诸如无效由服务器 106a-106n 所传送的并由分组处理引擎 240 处理的对象的请求。在一个实施例中, 缓存管理器 232 在缓存对象的 10ms 或更少时间内接收无效请求, 而在另一实施例中, 短至 5ms、2ms 或 1ms。在其他实施例中, 缓存管理器 232 响应于缓存管理器 232 的操作或功能而执行缓存操作, 诸如计时器的届满以引起无效对象或在处理任何缓存命令期间。在其他实施例中, 缓存管理器 232 使用设备 104 的分组处理计时器 242 以触发缓存操作。例如, 计时器 242 触发或以信号通知缓存器以检查在能够由计时器 242 设置的任何时间间隔上缓存对象的无效或期限。在一个实施例中, 设置计时器 242 以在所设置的 10ms 或更短时间内触发或以信号通知缓存器, 或在另一实施例中, 设置短至 5ms、2ms 或 1ms。在某些实施例中, 始发服务器 106a-106n 能设置对象的期限时间。在其他实施例中, 装置 104 或客户端 102a-102n 能设置对象的期限时间。

[0115] 在步骤 360, 缓存管理器 232 确定缓存器中所存储的对象的无效或期限。在某些实施例中, 根据计时器的期限无效缓存器中的对象。在一个实施例中, 缓存管理器 232 根据计时器的期限发布对对象的无效命令。在另一个实施例中, 响应于计时器 (诸如使用分组处理计时器 242 设置的计时器) 的届满, 由缓存管理器 232 自动无效缓存器中所存储的对象。在某些实施例中, 响应于分组处理计时器 242, 缓存管理器 232 检查对于缓存对象的任何计时器的期限。在一个实施例中, 缓存管理器 232 确定对象计时器已届满, 而在另一实施例中, 缓存管理器 232 确定对象计时器还没有届满。在另一实施例中, 响应于第二触发器或分组处理计时器 242 的第二计时器间隔的缓存管理器 232 将第二次检查之前检查的对象计时器是否已届满。

[0116] 在某些实施例中, 缓存管理器 232 解析、解释、访问、读取或另外处理无效命令或请求以识别对象用于在缓存器中无效。在一个实施例中, 缓存管理器 232 外部的实体向缓存管理器 232 发布无效命令以无效对象。在另一实施例中, 外部实体响应于分组处理计时

器 242 而发出无效命令。如果对象是有效的和 / 或还没有无效,则缓存管理器 232 响应于请求而无效对象。在某些实施例中,由缓存管理器 232 处理的无效请求响应于处理请求的分组处理引擎 240 的分组处理操作,其依次也响应于分组处理计时器 242。

[0117] 在步骤 365,缓存管理器 232 标记对象为无效。缓存管理器 232 以任何合适或所需的方式标记每个对象为无效。在一个实施例中,通过设置所存储对象的标志、特性或属性将对象标记为无效。例如,将标志设置为向缓存管理器 232 表示对象为无效的任何值。在另一实施例中,通过将对象移至用于存储无效对象的区域或缓存器的一部分而将对象标记为无效。在其他实施例中,通过数据库或链接列表或任何类型和形式的数据结构,缓存管理器 232 识别或跟踪所存储对象的无效和 / 或有效状态。在某些实施例中,缓存管理器 232 使用一个或多个对象以识别或跟踪缓存器中所存储的一个或多个对象的有效或无效。在另一实施例中,通过改变、修改或变更所存储的对象,例如删除或移除不使用的对象的一部分,或通过改变或破坏对象名称而将对象标记为无效。

[0118] 在某些实施例中,在步骤 370,缓存管理器 232 从缓存器刷新标记为无效的那些对象。在另一实施例中,一旦诸如由客户端 102a-102n 请求对象,缓存管理器 232 就从缓存器中刷新无效对象。在某些实施例中,缓存管理器 232 使用在对象无效或届满之后所接收的对象的更新副本或版本重写无效对象。在另一实施例中,缓存管理器 232 通过将另一对象存储至缓冲存储器的相同部分而再使用无效对象所占用的缓冲存储器。在另一实施例中,缓存管理器 232 并不刷新标记为无效的对象而是保持存储在存储器或缓存器的存储器中的对象。

[0119] 虽然方法 350 描述了响应于分组处理计时器和 / 或结合分组处理操作而无效并刷新缓存的对象以提供无效颗粒度,但是缓存器的任何操作和缓存器管理的任何技术以及这里所述的设备 104 的任何其他操作可以在分组处理计时器提供的微小颗粒度时间间隔上执行。在某些实施例中,缓存对象的无效或届满能以短至 100ms 时间间隔发生,而在另一实施例中,短至 50ms 时间间隔。在某些实施例中,缓存对象的无效或届满能以短至 25ms 时间间隔发生,而在其他实施例中,短至 10ms 时间间隔。而在其他实施例中,缓存对象的无效或届满能以短至 5ms 时间间隔发生,而在另一实施例中,短至 3、2 或 1ms 时间间隔。

[0120] 通过包含上面结合图 3A 和 3B 在方法 300 和 350 中所述的极小时间增量消逝之后无效对象的能力,可以实现改进动态产生对象的缓存。事实上,某些动态内容必须服从在极短的时间段内从缓存器存储和提供。然而,为了成功缓存这样的内容,根据本发明实施例的方法提供在无效对象并从缓冲存储器刷新对象之前在极短时间段缓存对象。例如,特定动态产生的对象是可缓存 1 秒之久,但较长的时间对于持续变化的内容来说经常是不被接受的。在一个实施例中,本发明的方法包括在 1 秒的微小部分之后无效缓存的内容或使缓存的内容届满。作为示例,如果装置 100 花费数毫秒产生动态响应,那么缓存器可以在小于或等于 100ms 的时间段的期间内存储和提供该响应,而不损害数据的“新鲜”。在 100ms 的周期内不会产生新的对象,因为它比产生新对象所花费的时间更短。因此,可以设置装置 104 以在持续时间中提供该先前对象。装置 104 无效低至非常小时间增量的能力经常对其中设置数据库事务隔离级别以允许重复读取或序列读取的应用环境是非常有用的。

[0121] 3. 无效命令

[0122] 常规缓存技术根据内容的预定期限时间而无效所存储的内容,所述预定期限时间

一般由管理员配置或从提供对象的服务器接收。下文描述的是用于无效内容以便更高效地缓存动态产生的内容的本发明的另一技术。本发明的技术包括在装置 104 接收无效命令的能力,该无效命令实时识别缓存器中之前存储的一个或多个对象为无效。例如,可以通过传送到客户端的网络分组或服务器向装置进行的应用程序接口(API)调用而传送该无效命令。这不同于常规方法,当提供对象时通过该常规方法服务器简单地设置在对象报头中所包括的缓存器期限时间。

[0123] 本发明的技术更具体地在图 4A 和 4B 中表示。图 4A 是示出维持诸如计算机存储器缓存器之类的缓存器的方法的流程图。简单概括并根据步骤 410,在缓存器中存储之前从始发服务器 106a-106n 提供的动态产生的对象。例如,动态产生的对象不会被识别为可缓存的或另外包括任何缓存器或缓存器控制信息。在步骤 420,在缓存器或缓存管理器 232 上接收无效命令。无效命令将之前提供的一个或多个对象识别为无效。在步骤 430,响应于无效命令,缓存器或缓存管理器 232 将该识别的对象标记为无效。

[0124] 在步骤 410 的进一步描述中,缓存管理器 232 在缓冲存储器元件中存储从任何源接收、获取或传送的动态产生的对象。在某些实施例中,从服务器 106a-106n 产生并提供动态产生的对象。在其他实施例中,由客户端 102a-102n 产生并提供动态产生的对象。在某些实施例中,装置 104 的另一部分、元件或处理产生对象并在缓存器中保存对象。在其他实施例中,由网络上另一装置 104 或另一计算设备产生动态产生的对象并且该动态产生的对象被传送或传递到装置 104。在某些实施例中,动态产生的对象没有被识别为可缓存的或识别为不可缓存的。在其他实施例中,动态产生的对象被识别为可缓存的或在缓存器控制下。

[0125] 在步骤 420,缓存管理器 232 接收将对象识别为无效的无效命令,该对象诸如是在缓存器中保存的动态产生的对象。在一个实施例中,无效命令包括向缓存器表示对象无效或可能“陈旧”的任何类型的指示或指令。在某些实施例中,无效命令识别对象并也识别对象无效的时间以及对象的哪些部分无效。在一个实施例中,缓存管理器 232 提供由始发服务器 106a-106n 远程调用的应用程序接口(API)。在某些实施例中,缓存管理器 232 提供任何类型和形式的协议用于接收命令并通过一个或多个网络分组答复命令。在一个实施例中,缓存管理器 232 或设备 104 提供可扩展标记语言(XML)API 接口用于接收并处理无效命令。例如,缓存管理器 232 提供 web 服务接口。在某些实施例中,缓存管理器 232 通过向始发服务器 106a-106n 发送确认、状态或其他响应而答复无效命令。在其他实施例中,缓存管理器 232 不答复无效命令。在一个实施例中,如果在始发服务器 106a-106n 中运行的应用执行使得所存储的对象“陈旧”的操作(诸如通过产生新的或更新的对象版本),则标记对象为无效。例如,当新闻编辑改变快速变化的新闻报道而因此想确保报道的最新版本被提供给客户端时,上述情况会发生。

[0126] 无效命令由产生对象的应用、另一服务器 106a-106n 或另一装置 104 从始发服务器发出。在一个实施例中,始发服务器 106a-106n 自动响应始发服务器 106a-106n 上动态产生的对象的改变而向缓存器 232 发出或传送该无效命令。也可以由服务器 106a-106n 和装置 104 外侧或外部的管理控制器产生无效命令。例如,管理控制器可以是在网络上运行并与装置 104 通信的任何类型和形式的程序或应用,诸如管理控制台。而且,客户端 102a-102n 能向装置 104 或缓存管理器 232 发出或传送无效命令。例如,如果客户端采取客户端 102a-102n 识别的措施能引起始发服务器上请求对象的改变,则客户端传送无效命

令。可以通过向用户的缓存器传送在缓存器上局部执行或使用 XML API 框架远程触发的命令而无效缓存器中存储的任何对象。

[0127] 根据步骤 430, 响应于无效命令, 将已识别为无效的缓存器中所存储的对象(例如, 之前提供的动态产生的对象) 标记为无效。不会从缓存器向请求客户端提供无效对象, 而是从始发服务器直接提供该无效对象。缓存管理器 232 能以任何合适或所需的方式标记每个对象为无效。在一个实施例中, 通过设置所存储对象的标志、特性或属性而标记对象为无效。例如, 将标志设置为向缓存管理器 232 表示对象为无效的任何值。在另一实施例中, 通过将对象移至用于存储无效对象的区域或缓存器的一部分而将对象标记为无效。在其他实施例中, 通过数据库或链接列表或任何类型和形式的数据结构, 缓存管理器 232 识别或跟踪所存储对象的无效和 / 或有效状态。在某些实施例中, 缓存管理器 232 使用一个或多个对象以识别或跟踪缓存器中所存储的一个或多个对象的有效或无效。在另一实施例中, 通过改变、修改或变更所存储的对象, 例如删除或移除不使用的对象的一部分, 或通过改变或破坏对象名称而将对象标记为无效。

[0128] 在某些实施例中, 装置 104 随后从缓存器刷新标记为无效的那些对象。在另一实施例中, 一旦诸如由客户端 102a-102n 请求对象, 则装置 104 从缓存器中刷新无效对象。在某些实施例中, 装置 104 使用对象的更新副本或版本重写无效对象。在另一实施例中, 装置 104 通过将另一动态产生的对象存储至缓冲存储器的相同部分而再使用无效对象所占用的存储器。

[0129] 使用本发明的缓存管理器 232 的命令无效 API, 与装置 104 通信的任何计算设备或用户请求无效缓存器中存储的对象, 诸如动态产生的对象。因此, 可以实时控制缓存器中存储的对象的无效, 而不是使用预定配置期限或无效时间段。因此, 使用这些技术, 可以从诸如数据库或始发应用服务器之类的外部应用处理节点控制缓存对象的寿命。因此, 装置 104 可以配置为与数据库一起工作使得数据库的变化自动触发从数据库(或应用)到装置 104 的无效命令用于刷新特定对象或多个对象。

#### [0130] 4. 使用无效命令来无效组

[0131] 在本发明的另一实施例中, 装置 104 同时识别并无效由缓存器所存储的对象组。在确定对象是否“陈旧”时, 在常规缓冲存储器中存储的对象中的每一个由缓存器单独地且分别地处理。当每个对象达到它的指定期限时间(一般由服务器设置并由缓存器保存在表中)时, 从缓冲存储器刷新该项。然而, 该常规方法是效率低的并最终不能成功处理在试图缓存动态产生的对象中出现的各种挑战。

[0132] 图 4B 示出了用于维持诸如计算机存储缓存器之类的缓存器的本发明的方法的另一实施例, 其中装置 104 具有产生、存储并无效之前已从始发服务器 106a-106n 提供的相关对象组的能力。简单概括, 在步骤 410, 在缓存器中存储诸如从始发服务器 106a-106n 提供的动态产生的对象之类的对象。在步骤 412, 缓存管理器 232 形成在缓存器中存储的, 之前提供的对象组。在一个实施例中, 所述组与一个或多个对象决定因素相关联或由一个或多个对象决定因素识别, 正如将在下文更详细地描述。在步骤 414, 缓存管理器 232 维持对象组的记录。在步骤 422, 缓存管理器 232 接收无效命令以无效对象组。在步骤 432, 缓存管理器 232 响应于无效命令而将对象组标记为无效。

[0133] 步骤 410 与图 4A 相同, 其中在装置 104 的缓存器中存储诸如之前已从始发服务器



106a-106n 提供的动态产生的对象之类的对象。在某些实施例中,一个或多个对象没有被识别为可缓存的,或者另外不具有任何缓存器或缓存控制器信息。例如,服务器 106a-106n 假定动态产生的对象将不能被缓存。

[0134] 根据步骤 412,装置 104 从之前从始发服务器 106a-106n 提供的并存储在缓存器中的对象集合中形成一组。任何合适或所需对象集合可以互相关联以形成一组。例如,为提供网页而产生的或与提供网页相关联的任何动态产生的对象可以形成一组。在某些实施例中,对象可以与多个组相关联。在其他实施例中,一个对象组可以形成另一个对象组的子集。在某些实施例中,所形成的对象组具有从相同服务器 106a-106n 提供的对象,而在其他实施例中,所形成的对象组具有从不同服务器 106a-106n 提供的对象。在其他实施例中,所形成的对象组可以包含来自客户端 102a-102n 的对象、来自服务器 106a-106n 的对象或由客户端 102a-102n 和服务器 106a-106n 二者产生或提供的对象。在一个实施例中,组中的一个对象是静态的而组中的另一个对象是动态产生的。在某些情况下,组中的一个对象识别为不可缓存的而组中的另一个对象被识别为可缓存的。在其他情况下,组中的对象根据服务器 106a-106n 提供的功能或应用是逻辑相关的。在另一情况下,组中的对象当与相同客户端 102a-102n 或相同用户相关联时是相关的。

[0135] 在步骤 414,保持对象组的记录。在实施这里所述的本发明操作中可以使用用于记录和保存对象组的记录或者另外关联对象的各种技术。在一个实施例中,记录可以直接保持在例如查询表中。在另一实施例中,记录可以以散列表的格式表示。在某些实施例中,缓存管理器 232 保持数据库中对象的关联或存储器中数据结构或对象。在其他实施例中,将组中的每个对象的标志、特性或属性指定给或设置为识别该组的值,诸如等于识别或引用组的名称或识别符的值,诸如下文更详细描述的对象决定因素。在某些实施例中,在识别为占有该组的缓冲存储器的一部分中配置、放置或定位对象的组。

[0136] 在步骤 422,在装置 104 或缓存管理器 232 上接收无效命令。根据图 4B 所述的实施例,无效命令识别一个或多个对象是无效的,否则是“陈旧的”。在某些实施例中,无效命令引用、识别或指定对象组的名称或识别符。在一个实施例中,无效命令包括单个无效请求以无效组中的所有对象。在另一实施例中,无效命令识别组中的一个对象以无效。在其他实施例中,无效命令包括多个无效请求以无效组中的多个对象。

[0137] 根据步骤 432,如果无效命令引用、识别或指定组的对象为无效、组中每个对象为无效或组为无效,则将之前提供的对象组标记为无效。在某些实施例中,如果无效命令将组中的对象识别为无效,则缓存管理器 232 将对象标记为无效。在其他实施例中,如果无效命令将组中的对象识别为无效,则缓存管理器 232 将组的对象标记为无效或将组中的每个对象标记为无效。在其他实施例中,当借助一个或多个无效命令而将多个对象识别为无效时,缓存管理器 232 仅仅无效对象的组。在另一实施例中,无效命令指定组的名称或识别符,以及缓存管理器 232 将组标记为无效、或将组中的每个对象标记为无效。

[0138] 在一个实施例中,装置 104 或缓存管理器 232 从缓冲存储器中刷新已被标记为无效的对象组。在某些实施例中,只有当组中的每个对象被标记为无效时才从缓存管理器刷新组中的对象。在其他实施例中,如果组的一个对象被标记为无效,则刷新整个组。在另一实施例中,一旦由客户端 102a-102n 接收对对象组或组的任何对象的请求,则刷新被标记为无效的对象组或组中的任何对象。在其他实施例中,一旦从服务器 106a-106n 接收到提

供组中一个或多个新对象的响应,则刷新被标记为无效的对象组或组中的任何对象。

[0139] 如下是上述实施例的示例。许多商业使用客户资源管理(“CRM”)应用来跟踪和评价资源管理的所有方面。经常,通过专用网络和包括因特网的公用网络实施并访问 CRM 应用。因此,这些应用(其对被频繁访问的大量数据提供访问)受益于缓存由该应用产生的数据。例如,通常产生销售报告并将之提供至远程连接的用户。由相关应用通过编辑来自销售信息的数据而产生这些销售报告,所述销售信息被提交到该应用服务器和 / 或他们潜在的数据库。当许多用户请求相同文档(即,特定销售报告)时,在没有缓存的情况下,应用服务器必须为每个请求重新产生对象,然而,如果在缓存器存储该对象,则保存应用和数据库处理,包括可能有价值的带宽,如同缓存器邻近请求的客户端。

[0140] 缓存该对象出现各种挑战,因为每次当新的销售被提交到在始发服务器(或它潜在的数据库)上运行的应用时,都需要更新销售报告中的信息。因此,必须无效在支持这些应用服务器的缓存器中存储的所有销售报告并从缓冲存储器刷新内容。然而,缓存的常规方法没有办法正确地确定潜在的数据库或应用的改变何时发生并因此不能合理评价动态内容的“新鲜性”。每当数据库或应用或始发服务器中改变发生时,缓存器必须能够识别已发生改变,以及作为改变的结果应该无效哪个对象组。如上所述,产生无效命令可以满足该需求,该无效命令包含链接到之前提供的对象组的对象确定因素。

[0141] 可以在单个体系层上形成多个相关对象组。或者,形成子对象组以产生多个体系层。在一个实施例中,由用户预指定对象组或对象子组。在另一实施例中,用户能建立规则,通过该规则装置 104 自动形成相关对象组并使之与对象决定因素相关联。

[0142] 5. 识别客户端请求或响应中的对象决定因素

[0143] 本发明实施例也能通过产生对象组并实施参数化无效而解决能够识别在始发应用服务器 106a-106n (和 / 或潜在数据库)上受状态改变影响的所有对象。在该实施例中,可以通过例如来自客户端的截取的 HTTP 请求而无效任何对象或预定对象组,以至缓存器解析以便识别对象决定因素。术语“对象决定因素”指唯一地或按其他方式引用、识别或指定一个对象或对象集合的任何信息、数据、数据结构、参数、值、数据模式、请求、答复或命令。在某些实施例中,对象决定因素是在通信中的字节或字符的模式,其与一个对象关联或用于唯一地识别该通信与该对象关联或该通信引用该对象。在一个实施例中,对象决定因素表示在始发服务器中对与对象决定因素相关联的缓存管理器 232 中所存储的之前提供的对象组是否已经或即将发生改变。在某些实施例中,对象组的对象是相关的,因为它们与至少一个对象决定因素相关联。下文将更详细地描述对象决定因素的非限制性示例和它们用途的进一步解释。

[0144] 在当前实施例的某些实施例中,对象决定因素是在客户端请求或响应中包括或嵌入的特定预定参数或数据结构。在其他实施例中,客户端 102a-102n、服务器 106a-106n 或装置 104 在通信中嵌入一个或多个对象决定因素,诸如表示对象决定因素的预定字符串或字符集。对象决定因素表示该请求是否具有使得在始发服务器 106a-106n 或与其链接的数据库中所存储的对象状态改变的效果。在一个实施例中,请求中对象决定因素的存在表示对象已发生或即将发生改变。在另一实施例中,对象决定因素的句法(syntax)、结构、参数或值表示对象已发生或即将发生改变。在一个实施例中,缓存器从客户端 102a-102n 接收对象请求。该请求包括缓存器识别将改变始发服务器或应用服务器的状态的特定参数或值

(对象决定因素),结果,其将由该始发服务器或应用服务器 106a-106n 之前产生的由缓存管理器 232 存储的特定相关对象变的“陈旧”。取决于用户设置的无效策略,参数(对象决定因素)需要无效由始发服务器之前提供并已由缓存器存储的一个或多个对象或对象组。缓存器被配置以识别受该状态改变影响的相关对象(即,链接到对象决定因素的那些对象或对象组),并借助用于将每个对象标记为无效和/或从缓冲存储器刷新该对象的方法而无效这些对象。

[0145] 上述技术在图 4C 中描述。如这里所述的其他实施例,步骤 410 包括在缓存器中存储诸如之前从始发服务器提供的动态产生的对象之类的对象。对象可以由在始发服务器 106a-106n 上运行的应用产生,或例如可以从始发服务器 106a-106n 访问的数据库提取。在某些实施例中,动态产生的对象被识别为不可缓存的或另外不被识别为可缓存的。

[0146] 根据步骤 421,缓存器截取或另外接收客户端和服务端之间的通信,诸如来自客户端的请求或来自服务器的响应等。在某些实施例中,请求针对特定对象,该对象之前已提供并存储在缓存器中。在另一实施例中,通信包括来自具有所请求对象的服务器的响应。在一个实施例中,根据建立的缓存协议和通信标准发生该接收或截取。虽然缓存管理器 232 或装置 104 一般被描述为接收请求、响应或通信,在接收该请求、响应或通信中,即使没有直接或明显地传送到缓存器,缓存器 232 或装置 104 也能由任何合适装置和/或机制截取或获取该请求、响应或通信。

[0147] 在步骤 423,在截取的通信中识别对象决定因素。缓存管理器 232 提取、解释、解析、访问、读取或另外处理截取的通信以确定或识别通信中的一个或多个对象决定因素。通信的任何参数、值、句法、数据、结构或一个或多个字符集都可以用于识别对象决定因素。在一个实施例中,缓存管理器 232 识别从客户端 102a-102n 到服务器 106a-106n 的请求中对象的名称或识别符,其中,客户端请求对象。在另一实施例中,缓存管理器 232 识别客户端 102a-102n 的请求中或来自服务器 106a-106n 的响应中第一个对象的名称或识别符,其表示对于缓存器中所存储的第二个对象已经发生或即将发生改变。在其他实施例中,缓存管理器 232 确定请求中任何字符模式是否匹配与缓存器中对象或对象组相关联的任何对象决定因素。在某些实施例中,为没有当前存储在缓存器中的对象确定对象决定因素。在其他实施例中,可以为当前标记为无效的对象确定对象决定因素。在其他实施例中,所请求对象的对象决定因素被确定与缓存对象的对象决定因素相关联。在另一实施例中,一旦第一引用、请求或响应通信中对象,缓存管理器 232 将识别的对象决定因素确认为该对象的对象决定因素。

[0148] 通过接收或解析诸如客户端请求或服务器响应之类的通信以识别对象决定因素,缓存管理器 232 或装置 104 能有效地确定是否将与所识别的对象决定因素相关的缓存对象标记为无效。因此,根据步骤 425,确定对象决定因素是否表示缓存对象发生改变。在某些实施例中,所识别的对象决定因素是没有改变、修改或产生对象的部分通信。在其他实施例中,所识别的对象决定因素是表示对于与对象决定因素相关联的对象已经或即将发生改变的部分通信。例如,通信可以是对于动态产生对象的得到请求或将改变用于一个或多个动态产生对象的数据的提交请求。在某些实施例中,通信中对象决定因素的存在表示在一个或多个对象上已经或即将发生改变。在另一实施例中,通信中命令、指示或指令的类型或名称以及对象决定因素表示在一个或多个对象上已经或即将发生改变。在另一实施例中,命

令、指示或指令的参数或变量的存在、值或设置表示在与对象决定因素相关联的一个或多个对象上已经或即将发生改变。

[0149] 在其他实施例中，缓存管理器 232 对截取的通信或对象决定因素执行散列函数、算法或操作以确定对象中是否已经发生改变。在某些实施例中，散列值与该对象的之前存储的散列值比较，并且如果不同那么缓存管理器 232 识别对象已改变。在另一实施例中，对象的散列值包括在通信或对象决定因素中。在一个实施例中，通信表示对象已通过参数的值或设置而改变，诸如使用布尔标记。在其他实施例中，下文更详细描述实体标签控制和有效机制能用于识别对象并确定对象是否改变。

[0150] 如果表示已改变，那么在步骤 431 与对象决定因素相关联或由对象决定因素识别的对象被标记为无效。在某些实施例中，由截取的通信所请求的对象根据步骤 431 被标记为无效，并根据步骤 440 从始发服务器 106a-106n 中被检索。另外，在其他实施例中，根据步骤 450 从缓存器中检索请求的对象。在一个实施例中，从缓存器中刷新被标记为无效的任何对象。

[0151] 6. 根据对象决定因素无效对象组

[0152] 本发明的上述实施例描述了根据客户端请求中对象决定因素的识别而无效缓存管理器 232 中之前提供的对象的情况。在另一实施例中，该一般概念也用于识别并无效与一个或多个对象决定因素相关联的对象组。该实施例在图 4D 中示出。

[0153] 图 4D 中所述方法以与图 4C 的方法相同的形式开始。步骤 410 包括在缓存器中存储诸如之前从始发服务器提供的动态产生的对象之类的对象。在某些实施例中，一个或多个对象没有被识别为可缓存的。根据步骤 412 并类似于图 4B，之前提供的对象形成组。在一个实施例中并根据本发明的对象决定因素技术，对象组与至少一个对象决定因素相关联并由至少一个对象决定因素识别。如下文更详细地描述，在某些实施例中，组与对象决定因素的关联性取决于用户缓存策略的特性和细节，诸如由策略引擎 236 定义、控制或使用的策略。在其他实施例中，组的一个或多个对象决定因素包括组中对象的一个或多个对象决定因素。在另一实施例中，组的对象决定因素包括组中对象的对象决定因素的组。

[0154] 根据步骤 414，如果合适时维持组的记录，以及其相关的对象决定因素。该步骤类似于图 4B 所示的步骤 414。在一个实施例中，在查询表中维持组的记录和 / 或任何对象决定因素。在其他实施例中，以散列表的格式维持组的记录和 / 或任何对象决定因素。散列表被设计以有效地存储在它们的字母和数字序列中有宽阔间隙的非连续键。在另一实施例中，在散列表的顶端建立索引系统。在某些实施例中，缓存管理器 232 维持作为一组的对象与数据库中一个或多个对象决定因素的关联性、或存储器中的数据结构或对象。在其他实施例中，将组中的每个对象的标志、特性或属性指定给或设置为识别该组的值，诸如等于识别或引用组的名称或识别符，或组的对象决定因素的值。在某些实施例中，在识别为占有该组的缓冲存储器的一部分中配置、放置或定位对象的组。在另一实施例中，一个或多个对象决定因素被存储以与对象组相关联。

[0155] 步骤 421 和 423 类似于图 4C 中所示的步骤 421 和 423。根据步骤 421，缓存管理器 232 或装置 104 截取或另外接收客户端 102a-102n 和服务器 106a-106n 之间的通信，诸如来自客户端的对于之前提供的并在缓存器中存储的对象的请求。在一个实施例中，缓存管理器 232 截取从客户端 102a-102n 到服务器 106a-106n 的请求。在某些实施例中，请求

针对缓存器中存储的对象。在其他实施例中,该请求是到服务器 106a-106n 的指令、命令或指示,其使得缓存器中存储的对象发生改变,诸如使得对象被动态产生。在另一实施例中,缓存管理器 232 截取从服务器 106a-106n 到客户端 102a-102n 的响应,该响应包括或识别缓存器中所存储的对象。

[0156] 在步骤 423,在截取的通信中识别对象决定因素。如上所述,对象决定因素表示在始发服务器 106a-106n 上请求对象中是否已经或即将发生改变。然而,在图 4D 的实施例中,对象决定因素与对象组相关联。这样能高效地无效缓存器中存储的受特定对象决定因素影响的所有对象。在某些实施例中,识别组中的对象的对象决定因素。在其他实施例中,识别对象决定因素,例如对于对象组的组对象决定因素。在另一实施例中,识别组中一个或多个对象的对象决定因素的组合。

[0157] 因此,根据步骤 427,确定对象决定因素是否表示之前提供的对象组中的改变。在某些实施例中,截取的通信中组的对象决定因素的存在表示对于组中一个或多个对象或所有对象已经或即将发生改变。在其他实施例中,截取的通信中的命令、指示或指令的名称和类型表示该改变。在另一实施例中,通信中任何参数或变量的存在、值或设置也表示该改变。

[0158] 如果在步骤 427,对象决定因素表示组中的改变,那么根据步骤 435 将缓存器中之前提提供的对象组标记为无效。在某些实施例中,根据步骤 440 从始发服务器 106a-106n 请求或检索组的一个或多个对象或所有对象。如果在步骤 427,对象决定因素并不表示组中的改变,则在某些实施例中,根据步骤 450 从缓存管理器 232 检索请求作为部分的截取通信并之前提供以及存储在缓存器中的任何对象。在一个实施例中,由缓存管理器 232 从缓存器中刷新被标记为无效的任何对象或对象组。

#### [0159] 7. 组的指定

[0160] 缓存器管理员会具体地指定将哪些对象包括在特定组中。无论何时在缓存器中存储对象,管理员可以根据配置使得对象成为一个所配置的或固有的组中的成员。所配置的组可以根据管理员之前确立的配置或可替换地根据应用行为和涉及对象无效的其他数据。如果对象的配置组是动态的,则对象也可以是固有组的一部分。由重要的无效参数的值分组固有组中的对象。

[0161] 通过允许非常灵活的对象分组,缓存器可以使得无效具有灵活性和协调性,这是高效地缓存动态产生的内容所必需的。缓存器可以同时无效非常具体的对象组,因此使得缓存器更加响应无效动态产生的内容的频繁需要。当缓存器将对象指定给一个组时,该组确定与该对象相关的若干事件,包括无效参数和命中决定因素,以便使其与一个或多个对象决定因素相关联。

[0162] 在客户资源管理(“CRM”)示例中,缓存器管理员预定每个分组。例如,管理员配置缓存器以通过名称分组每个销售部门。因此,管理员可以指定汽车部、机车部、等,并且每次在进入缓存器的请求中识别对象决定因素时,缓存器可以借助对象决定因素无效链接到合适部门的指定组中所存储的所有对象。

#### [0163] 8. 基于规则的分组

[0164] 可替换地,缓存器管理员能建立一些规则,这些规则允许缓存器装置确定在运行中哪些对象包括在特定一个组或多个组中。该基于规则的分组根据所建立的规则依赖于组

的指定,所述建立的规则将对象链接到重要对象决定因素,缓存器利用该重要对象决定因素产生相关组。该方法的示例涉及利用缓存器识别将哪些对象放入每个组所使用的规则来配置缓存器。

[0165] 再次参见 CRM 示例,规则表明缓存器应该将应用上设置的销售部门的每个子部门识别为它本身的分组。这样,可以产生分组,而不需要缓存器管理员必须具体地识别每个分组而是允许缓存器根据相关规则确定。该技术产生了一种更灵活并通常更少工作量的加强方式以指定分组。缓存器管理员可以配置一个规则,该规则表明销售的每个子部门(即,销售\汽车、销售\摩托车等)应该由缓存器产生新的分组。当由应用借助缓存器处理并返回来自汽车销售部的请求时,缓存器可以识别销售的每个子分组并根据预先配置的规则自动地为其产生分组。

[0166] 每当缓存器发现对于类型报告/销售/汽车或报告/销售/摩托车的对象的新请求时,由该缓存器实施规则。当摩托车销售部请求表示它是销售部的子分组,然后是自行车销售部等时重复该处理,因为缓存器识别这些子分组并为它们中的每个确立对象分组。当已知无效请求到达链接到这些分组之一的缓存器时,或如果在客户端请求中识别相关对象决定因素(例如,在解析请求中发现销售报告提交到摩托车销售部销售/摩托车),则缓存器知道无效摩托车销售部分组中的所有缓存对象。

[0167] 这样,当缓存器识别到应用提供的数据已经或即将发生改变(或者因为缓存器识别到缓存器所接收的请求内容将触发应用的改变,或者因为发生某些外部改变)时,上述技术能通过分组处理使得缓存器快速简单地识别哪些对象需要无效。这样,缓存器能够无效因为应用或数据库状态的改变而不再“新鲜”的大量动态产生的对象。

[0168] 也可以使用智能统计引擎增强缓存器成功保存并从其缓冲存储器提供动态产生的内容的能力,所述智能统计引擎检查请求和响应业务的模式以便在一个时间段上确定能提供最大缓存利益的对象集。该引擎可以或者集成在缓存装置本身中,或者作为试探法在单独计算机中运行以选择某些对象子集用于进一步调查以确定动态缓存的合适性。

[0169] 9. 进一步使用对象决定因素

[0170] 如上所述,对象决定因素可以是任何数据结构,其表示在始发服务器中对与对象决定因素相关联的缓存器中所存储的之前提供的对象组是否已经或即将发生改变。可以根据请求中嵌入的预定字符串值设置对象决定因素。例如当进来的请求具有特定 USERID 时,USERID 可以链接到缓冲存储器中的对象组,每次提交或其他请求来自该特定 USERID 时,所述对象组应该被无效。对象决定因素的可能候选也可以包括使用初始提供对象的服务器的服务识别符。服务识别符包含在 HTTP 请求中存在的服务 IP 地址、TCP 端口和服务识别符。

[0171] 存在在请求中的另一可能对象决定因素是请求统一资源定位器(“URL”)。对于缓存静态对象的情形,请求 URL 一般足以唯一地识别对象。然而,对于请求动态产生的内容,URL 中存在的信息不足以识别缓存的对象。因此,缓存器必须检查请求中的其他信息以寻找包括在 HTTP 报头、cookie 报头或其他自定义 HTTP 报头中的对象决定因素。缓存器可以另外在客户端请求中的多个其他位置中寻找相关参数信息的子集,包括但不限于:在 URL 查询字符串中、在 POST 主体中、在 cookie 报头中或在任何其他请求或响应报头中。

[0172] 在解析用于对象决定因素的 URL 中存在的问题是 URL 和其他报头除了包含与缓存器决定相关的信息之外还包含许多信息。因此,缓存器必须能够通过许多信息进行解析以

便能够识别合适的对象决定因素。此外,报头中的数据经常是任意排序的,这意味着在HTTP报头中放置该数据没有标准方式并因此简单的比较经常不足以在该字符串中定位相关对象决定因素。

[0173] 如果不存在预配置策略以将特定对象决定因素与缓冲存储器中存储的相关对象或对象组相匹配,则在另一实施例中缓存器仍能进行该确定。例如,缓存器能检查并解析请求的各种方面以寻找在该请求中是否发现任何其他对象决定因素和其是否用于将该请求链接到应该被无效的缓冲存储器中所保存的特定对象。可替换地,根据特定预定试探法,也可以使得缓存器检查对缓存器所确定的特定对象决定因素的请求有目的地链接到特定对象或对象组。例如,当对于更新与特定 USERID 相关的日历的请求进入缓存器时,本发明的实施例可以被设置以识别需要无效所有缓存的对象,该所有缓存的对象的 USERID 等于更新日历的请求的 USERID 并且该所有缓存的对象包含任何一个特定日期的用户日历。

[0174] 缓存器也假设对象决定因素以非指定的顺序存在于 URL 主干(URL stem)、URL 中存在的查询、POST 主体中或 cookie 报头中作为名称 = 值或类似对的组。在一个实施例中,假设查询被格式化为名称 = 值对的列表。因此,用户可以配置哪些参数名是重要的。使用首次它的访问 URL 锁住每个缓存的对象。URL 看上去象 /site/application/special/file.ext?p1=v1&p2=v2&p3=v3。/site/application/special/file.ext 是 URL 主干。p1=v1&p2=v2&p3=v3 是 URL 查询并包含参数值对。这些参数值对也存在于 POST 主体或 Cookie 报头中。

[0175] 在一个实施例中,用户或管理员确立 p1 和 p2 将是无效参数或对象决定因素。此后缓存器将自动地分组具有匹配 p1 和 p2 值的对象。实施该分组的一种方式是将 p1 和 p2 映射到数据库表中的主关键字,即映射到表中的单独可识别对象,缓存器知道如何引用这些对象以便确定无效状态。为了更新那些数据库表中的某些内容,为了反映缓存器中保存的数据不再有效的事实,缓存器将指定 p1 和 p2 的新值并当下次缓存器提供该内容时识别该新值时,它将知道无效其存储器中保存的链接对象。当缓存器遇到该请求时,一旦看见更新请求它就知道必须无效具有匹配 p1 和 p2 值的组 - 因为缓存器知道起源中的数据将改变,因此影响与那些 p1 和 p2 对象决定因素相关的所有对象。

[0176] 为了解决其中管理员没有将请求中嵌入的特定参数预配置为对象决定因素的更复杂情况,缓存器可以利用用户配置策略以从请求中提取相关对象决定因素用于帮助识别何时无效对象的分组。然后决定因素字符串用于定位缓存器中保存的对象组并无效该对象。这些对象决定因素可以用于配置缓存器以产生重要参数值的列表。如果引入的写请求具有对于重要参数的匹配值,那么应该无效约束这些参数名的对象。可替换地,用户可以指定策略框架操作,该策略框架操作从请求中提取对象决定因素字符串。从写请求中提取对象决定因素字符串并无效具有匹配决定因素字符串的所有对象。在该可选方案中,请求到达缓存器,缓存器确定请求字符串是否匹配无效策略。无效策略指定其中应该无效内容组的对象。

[0177] 可替换地,缓存器可以使用在客户请求中存在的任何其他用户信息。如上所述,认证和授权集成允许缓存器访问用户信息。如果缓存对象的相关分组被链接到用户或用户组,则 USERID 或 GROUPID 可以是决定因素之一。虽然用户信息经常是重要的对象决定因素,但是用户信息经常不出现在 HTTP 请求中。在本发明的另一实施例中,本发明的动态缓存方

面可以与本申请人的另一个专利申请相结合。为了实现该目的,参考申请号为 11/169,002 的本申请人的上述待决专利申请(“the Integrated Caching patent (集成缓存专利)”)。该申请描述了一种将缓存器和多种其他网络组件相集成的系统和方法,该多种其他网络组件包括执行特定种类的验证、访问控制和查帐(A A A)基础结构的能力。根据由应用产生的数据的安全级别被应用到而是从缓存器提供的数据。该技术允许应用能缓存以别的方式不能被缓存的敏感、访问控制的信息。

[0178] 本方法允许缓存器识别这样的用户,该用户在 HTTP 请求中不包括可识别的用户信息但能借助在集成缓存专利中所述的 AAA 方法可识别。该方法能使缓存器通过检查从 AAA 处理共享的授权状态信息而识别对特定请求的相关用户。在另一实施例中,集成能使安全策略应用到缓存器中保存的信息中以避免未授权用户访问缓存器中保存的信息。

[0179] 该方法还克服了以下事实提出的挑战:在缓存器能响应来自客户端的相关请求之前动态产生的数据的重要部分需要请求该数据的客户端被授权以及认证。缓存器必须具有授权由认证用户进行的请求的能力以便应用可以缓存访问控制的对象并通过将该动态缓存的技术以及认证和授权信息相集成,可以实现该安全性。如果对于用户或用户组个性化对象,则 USERID 或 GROUPID 将是对象决定因素之一。因此,根据由应用产生的数据的安全级别也应用到缓存信息。该技术允许应用能缓存以别的方式不能被缓存的敏感、访问控制的信息。

[0180] 最后,可以从请求中解析像日期时间、开始时数据库状态等的其他信息并将该信息用作对象决定因素以确定缓存器中存储的对象是否仍然有效。缓存器可以通过在对象组中配置合适的期限行为而考虑该情况,所述对象组被配置对这种外部变量是灵敏的。

[0181] 为了进一步克服由以下事实提出的挑战:必须由缓存器解析和截取对动态内容的请求,根据本发明实施例的缓存器可以限制认为哪些参数是对缓存器的相关对象决定因素。这样,可以提高从缓存器提供对象而不是向可应用的应用服务器转送该请求的成功率。通过示例,来自客户端的请求查询包含城市和国家参数。然而,缓存器可以被配置以符合缓存器保存内容以识别可以对来自客户端的请求提供响应的应用的需求,查询表示来自给定国家的所有客户而不考虑城市值。为此,城市参数是不相关的,并且缓存器能识别该事实。如果只有城市参数匹配而不管对国家参数指定什么,则可替换实施例涉及配置缓存器,以便可以从缓存器提供响应。

[0182] 总之,缓存器实施一般的参数化对象匹配。在该方法中,缓存器被配置以识别请求中信息的子集,该信息子集将用作对象决定因素并链接到特定对象,因此当识别该对象决定因素时,缓存器可以在评价对象或对象组是否仍“新鲜”并是否能够从缓存器提供时使用该出现(或反之没出现该决定因素)。缓存器维持每次请求进入时它所查询的表以检查配置参数,用于确定请求的数据是否仍然“新鲜”,并其也允许缓存器将相关数据与缓冲存储器中保存的合适对象匹配。

[0183] 10. 具体数

[0184] 在另一实施例中,缓存器可以使用具体数以无效对象组。在由于起始状态改变而缓存器需要同时改变每个对象组的状态的地方,具体数提供了一种用于实现该无效的简单技术。然而,识别每个对象并单独改变状态是保证缓存器中保存的数据的“新鲜性”的低效率方法,而使用具体数提供一种更简单高效的方法来无效对象组。当前实施例描述每个对



象如何指向表示组的数据结构并因此服务器只需要发送一个命令,该命令改变该组的数据结构中的状态。当来自客户端的对缓存对象的随后请求到达时,缓存器必须首先判定状态是否改变。为此,缓存器查询数据结构以参考组的状态是否改变。

[0185] 为了有效地实施数据结构,缓存器必须能够确定是否查询状态改变。因此,缓存器必须能够确定它是否已查询组中的状态改变。这正是具体数有用的地方。缓存器将动态产生的对象关联到内容组中。可以通过具有特定索引值或数据结构中包含的“具体数”的散列表查询处理来表示这些内容组中的每一个。此后,无论何时缓存器接收缓存器识别为引起状态改变的客户请求时,客户端解析对相关参数的客户请求,根据识别的对象决定因素而执行散列查询,并增加数据结构中的索引或具体数。每当客户端请求保存在指定分组中的对象时,缓存器对对象执行散列算法,并比较它与该内容组在数据结构中的原始保存值。如果存储的值与由缓存器对该对象计算的数值相同,那么缓存器知道该内容保持“新鲜”并可以提供给请求者。如果缓存器检测到在为该对象计算的当前具体数和数据结构中为该内容组所保存的数之间的差异,那么缓存器知道保存的对象不再“新鲜”。然后,缓存器无效保存的对象并将请求发送至应用服务器。当响应返回时,缓存装置将在缓冲存储器中保存新的响应并将该响应再次链接到新的数据结构。此后,每当缓存器接收到对该分组中的对象的请求时,缓存器可以进行比较并假设数据结构不再进一步改变,缓存器可以提供新保存的对象。

[0186] 通过以该方式利用对象组的无效,缓存器可以非常快速地无效 - 并且花费的时间是恒定的而不论无效的对象数量。通过该更快速更有效的无效处理,本发明的技术能使缓存器更高效地处理动态产生的对象。该方法允许位于应用前方的缓存装置更积极地保存和提供动态产生的对象而不因为数据的快速改变而提供无效或“陈旧”的内容。本实施例能使缓存器提供频繁或不可预知地改变的数据,因此改进缓存器的性能。缓存器也能通过使用用户命令并同时检查和分组各种网络业务而无效缓存存储器中保存的对象和对象组。

#### [0187] 11. 闪速缓存器和闪速群

[0188] 本发明的另一实施例还包括一种能够增加缓存器命中率(hit rate)以极快改变动态产生的对象的技术,所述动态产生的对象以别的方式是不能被缓存的。当缓存管理器 232 从客户端接收到对特定对象的第一请求时,缓存管理器 232 向始发服务器 106a-106n 转送该请求用于处理,因为作为首次请求,缓存管理器 232 还不具有该保存的对象。当产生响应对象并然后通过缓存管理器 232 将该响应对象返回到请求的客户时,缓存管理器 232 自动保存对象的副本。随着对象极快速地改变,与本发明相反的常规缓存器只是将所有响应传递到始发服务器 106a-106n 用于处理,因为该内容总是被假设不再“新鲜”,并因此对缓冲存储器不再有效。

[0189] 一方面,本发明针对一种“闪速缓存器”技术,用于在缓存管理器 232 或装置 104 在传送或等待传送对于对象的第一请求者的响应的处理期间处理对于由缓存管理器 232 接收的对象的另外请求。现在参考图 5,本发明的方法 500 描述了本发明的闪速缓存器技术。简单概括并根据图 1 和 2,在步骤 510,本发明的缓存管理器 232 为由第一客户端(例如图 1 所示的客户端 102a)所请求的对象接收来自始发服务器 106a-106n 的响应。该对象包含由始发服务器 106a-106n 之一产生的动态产生的对象,以及来自始发服务器 106a-106n 的响应包括该动态产生的对象。在步骤 515,缓存管理器 232 诸如通过网络堆栈、例如设备 104

的 TCP/IP 堆栈请求将响应传送到第一客户端。在步骤 520, 该响应可以被保存或保持在缓冲器中同时等待传送。例如, 设备 104 调节到慢速连接或低带宽客户端 102a 的通信, 并因此设备 104 排队表示响应的网络分组。

[0190] 在步骤 525, 当第一客户端 102a 的响应在缓冲器中等待传送或另外在正传送的处理中时和 / 或在完成将响应传送到第一客户端 102a 之前, 缓存管理器 232 从第二客户端 102B 接收对于对象的第二请求。在步骤 530, 缓存管理器 232 确定对象当前正在缓冲器中, 并从缓冲器中向第二客户端 102B 提供对第二请求响应的对象。在一个实施例中, 当设备使用单个 TCP/IP 堆栈时, 可以提供相同的对象用于传送到第一客户端 102a 和第二客户端 102B。在步骤 535, 第一响应被传送到第一客户端 102a, 以及响应被传送到第二客户端 102B。在步骤 540, 从缓冲器中移除第一客户端 102B 的响应。

[0191] 在进一步的描述中, 在步骤 510, 设备 104, 诸如由缓存管理器 232, 截取或另外接收来自始发服务器 106a-106n 的响应, 该响应是针对第一客户端 102a 对一个对象 (诸如动态产生的对象) 的请求。在一个实施例中, 动态产生的对象诸如由始发服务器 106a-106n 识别为不可缓存的, 或另外不被识别为可缓存的。在某些实施例中, 缓存管理器 232 从第一客户端 102a 接收请求并将该请求转送到始发服务器 106a-106n。在一个实施例中, 在将请求转送到始发服务器 106a-106n 之前缓存管理器 232 为所请求的对象检查缓存管理器 232。在某些情况下, 缓存管理器 232 确定对象是无效的或已届满的。在其他情况下, 缓存管理器 232 确定对象没有存储在缓存管理器 232 中或另外在缓存管理器 232 中不可得。例如, 已经刷新该对象或这是第一次从始发服务器 106a-106n 请求对象。在某些实施例中, 响应包括该对象, 并在其他实施例中, 响应指示状态, 诸如关于对象的失败或错误消息。

[0192] 在步骤 515, 提供对于由第一客户端 102a 请求的响应用于传送到第一客户端 102a。在某些实施例中, 设备 104 接收响应, 并请求分组处理引擎 240 以将响应传送到客户端 102a。在其他实施例中, 缓存管理器 232 接收响应并请求分组处理引擎 240 以将响应传送到客户端 102a。在一个实施例中, 设备 104 包括一个网络堆栈用于向客户端 102a-102n 接收和传送网络分组。在某些实施例中, 网络堆栈包括 TCP/IP 堆栈。在其他实施例中, 设备 104 包括多个网络堆栈, 每个堆栈都与一个或多个客户端 102a-102n 相关联或由一个或多个客户端 102a-102n 使用。如本领域技术人员能识别并理解的, 一个或多个网络堆栈与内核 204、网络端口 226 和 / 或分组处理引擎 240 相关联。

[0193] 在步骤 520, 在向第一客户端 102a 传送在步骤 515 所请求的响应期间, 设备 104 在缓冲器中排队、保持或另外保存响应。在某些实施例中, 缓冲器是网络堆栈, 诸如 TCP/IP 堆栈的一部分。例如, 缓冲器包括在 TCP/IP 堆栈或网络驱动器、滤波器或其他网络相关软件中使用或由它们使用的数据结构, 所述网络相关软件处理或操纵网络堆栈的任何部分。在其他实施例中, 缓冲器是网络端口 226 的一部分。在其他实施例中, 缓冲器是分组处理引擎 240 的一部分, 或在其他实施例中, 内核包括缓冲器。缓冲器可以是位于设备 105 的任何部分中的任何存储器或储存器元件。在许多实施例中, 在缓冲器中排队、保持或保存响应为一个或多个网络分组, 诸如基于 TCP/IP 协议的分组。在一个实施例中, 在一个或多个数据结构中保持响应, 所述数据结构提供本发明使用的缓冲器。如本领域技术人员能认识并理解的, 可以以任何合适的形式在缓冲器中排队、保持或保存响应。

[0194] 而且, 可以任意的、预定的、隐含的、明确的或以其他方式并使用任何合适的装置

和 / 或机制在任何时间段期间,在缓存器中保存或排队响应。在一个实施例中,当网络分组根据传送率、分组排队率、网络通信量和拥塞或任何其他网络特性并如由设备 104 所确定的内容诸如通过分组处理引擎 240 和 / 或网络端口 226 等待由设备 104 传送时,在缓冲器中保存响应。在另一实施例中,对于预定时间段,响应被保持在缓冲器中。在某些实施例中,设备 104 根据客户端网络连接的诸如带宽、连接类型等的特性或客户端 102a 的特性排队、管理网络分组并向客户端 102a 传送网络分组。例如,对于客户端 102a 来说,该客户端 102a 到设备 104 的连接比设备 104 到始发服务器 106a-106n 的连接慢,设备 104 能调节或另外管理网络分组到客户端 102a 的传送以便为客户端 102a 提供所需性能或行为或者客户端 102a 的网络连接。因此,设备 104 根据任何排队或缓冲器管理逻辑、功能、规则或操作在缓冲器中排队或保持响应达所需时间段。

[0195] 在步骤 525,当第一客户端 102a 的响应保持在缓冲器、诸如 TCP/IP 堆栈的缓冲器中、或正在传送时或另外在完成将响应传送到第一客户端 102a 之前,设备 104 诸如通过缓存管理器 232 截取或另外接收来自第二客户端 102B 对于对象的第二请求,所述对象是针对第一次请求从缓存管理器 232 获取的。即,在一个实施例中,由第二客户端 102B 请求的对象保存在缓冲器中等待传送。在某些实施例中,设备 104 对于第一客户端 102a 所请求的相同对象从多个客户端 102b-102n 接收多个请求,所述相同对象目前正保持在缓冲器中以用于传送到第一客户端 102a 或另外目前正传送到第一客户端 102a。在某些实施例中,缓存管理器 232 确定第二请求使用任何合适的装置和 / 或机制、诸如这里所述的任何技术,例如使用对象决定因素正在请求与第一请求相同的对象。在某些实施例中,装置 104 或缓存管理器 232 使用任何类型和形式的排队机制而排队第二请求和队列中的任何其他请求。因此,在一个实施例中,当装置 104 或缓存管理器 232 代表服务器 106a-106n 排队并响应请求时,客户端 102a-102n 不需要重新提交请求。在另一个实施例中,装置 104 或缓存管理器 232 透明地向客户端 102a-102n 排队和响应请求而不将请求提交到服务器 106a-106n。装置 104 或缓存管理器 232 能在存储器中使用诸如排队、对象或数据结构的任何机制以排队使用本发明技术响应的来自客户端 102a-102n 的请求。

[0196] 在步骤 530,设备 104 确定由第二客户端 102b 请求的对象在缓冲器保存的响应中或在当前正传送到第一客户端 102a 的响应中。在某些实施例中,缓存管理器 232 确定对象存储在缓冲器中。在某些情况下,缓存管理器 232 首先为该对象检查缓冲存储器或存储器,并然后为该对象检查缓冲器。在其他情况下,缓冲器被认为是用于缓存的对象的存储位置,该缓存的对象将由缓存管理器 232 搜索或管理。在其他实施例中,缓存管理器 232 请求分组处理引擎 242 检查对象是否在缓冲器中。在其他实施例中,缓存管理器 232 通过内核 204 的任何应用编程接口 (API) 检查对象是否在缓冲器中。在某些情况下,缓存管理器 232 对接到任何驱动器、网络处理器、滤波器或操纵或处理网络堆栈的任何部分的其他软件以确定对象是否在缓冲器中。

[0197] 继续在步骤 530,本发明提供缓冲器中存储的或作为对第一客户端 102a 的响应而被传输的对象用于响应在步骤 525 来自第二客户端 102B 的第二请求。在某些实施例中,设备 104 诸如经由分组处理引擎 104 通过使用缓冲器的任何部分和网络堆栈的相关数据结构而形成对第二请求的响应。在一个实施例中,对象在缓冲器中只保存一次,例如在单个网络堆栈配置中并用于形成网络分组以分别传送到第一客户端 102a 和第二客户端 102B。在其

他实施例中,来自缓冲器的对象的副本用于形成对第二请求的响应。在其他实施例中,从第一网络堆栈向第二网络堆栈提供对象以形成第二客户端 102B 的第二请求的响应。可以在缓冲器中保持第一客户端 102a 的响应,以及缓冲器中响应的任何部分或所有响应都用于提供对第二客户端 102B 的响应。在一个实施例中,在不修改的情况下可以使用在缓冲器中保存的第一客户端的响应以响应第二客户端 102B。在另一实施例中,缓冲器中保存的表示第一客户端 102a 的响应中的对象的数据被修改以提供对第二客户端 102B 的响应。

[0198] 在本发明的方法 500 的步骤 535,从设备 104 向第一客户端 102a 传送第一响应,并从设备 104 传送响应于第二客户端 102B 的第二请求的第二响应。到第一客户端 102a 和第二客户端 102B 的响应可以以任何顺序、利用互相之间的任何时间间隔、使用相同网络堆栈 267 或不同网络堆栈从设备 104 传送。在某些实施例中,也在缓冲器中排队、保持或保存等待传送的对于第二客户端 102B 的响应。因此,可以通过在缓冲器中保存的并且还没有传送、刷新、移除或另外不可使用的第一响应或第二响应提供对于相同对象的任何另外请求。

[0199] 在步骤 540,从缓冲器刷新或另外移除第一客户端 102a 的响应。在某些实施例中,如果第二客户端 102B 的响应也被保存到缓冲器,则也移除第二客户端的响应。在其他实施例中,第二客户端的响应仍然保持在缓冲器中并当从第三客户端接收对于对象的第三请求时使用第二客户端响应再次实施方法 500 的步骤 525 至 535。

[0200] 另一方面,本发明针对一种用于处理如下情况的“闪速群”技术,其中在该情况中在服务器正处理并返回对第一请求者的响应对象期间装置 104 或缓存管理器 232 接收对于相同对象的另外请求,例如几乎同时的请求。一旦由缓冲器响应所有该几乎同时的请求,则从缓冲存储器立即刷新对象,而不需要另外期限时间或由应用或管理员进行的无效操作。本发明的该技术能在极短的时间量内缓存和提供对于对象的数据,所述对象另外被认为是不可缓存的。该方法在向大量并发用户提供快速变化的数据的应用中产生巨大的改进,所述应用诸如实时股票报价或快速变化的新闻报道。

[0201] 图 6 描述了当本发明的缓存器响应第一请求而接收这样内容时向每个客户端请求者提供相同内容所采取的步骤的方法 600。简单概述方法 600,在步骤 610,缓存管理器 232 或装置 104 从第一客户端 102a 接收对于来自始发服务器 106a-106n 的对象的第一请求,所述对象例如是动态产生的对象。在步骤 615,缓存器将第一请求转送给始发服务器 106a-106n。当等待对第一请求的响应时或在响应第一客户端 102a 的第一请求之前,在步骤 620,缓存器从第二客户端 102b 接收对于该对象的第二请求。在某些实施例中,缓存器并不将第二请求转送给始发服务器而是排队第二请求。在步骤 625,缓存器从始发服务器接收对于第一请求的响应,并在步骤 630,响应第一请求向第一客户端 102a 以及响应第二请求向第二客户端 102b 传送所接收的响应和 / 或响应中的对象。在步骤 635,在完成将所接收的响应和 / 或响应中的对象传送给第一客户端 102a 或第二客户端 102b 之前缓存器从第三客户端 102c 接收对于该对象的另外第三请求。在步骤 640,缓存器也向第三客户端 102c 传送所接收的响应和 / 或响应中的对象。一旦响应请求,则在步骤 645 缓存器从缓冲存储器中刷新对象,例如而不需要任何期限时间或由应用、始发服务器或管理员进行的无效操作。

[0202] 在进一步的描述中并根据图 1 和 2、本发明的方法 600 的步骤 610,在一个实施例中,装置 104 的缓存管理器 232 在客户端 102a 和始发服务器例如 106a 之间通信期间的任何时刻从客户端 102a-102n (例如图 1 中的客户端 102a)接收对于一个对象(诸如动态产生

的对象)的请求。在一个实施例中,来自第一客户端 102a 的请求可以是第一客户端 102a 第一次请求对象。在一个示例中,客户端 102a 一旦连接到始发服务器 106a 则首次请求对象。在另一示例中,缓存管理器 232 代表客户端 102a 或任何其他客户端 102a-102n 之前还没有缓存或请求来自始发服务器 106a 的对象。在其他实施例中,来自客户端 102a 的请求是对于之前请求对象的随后请求。在其他实施例中,缓存管理器 232 之前已缓存在步骤 610 由客户端 102a 所请求的对象。因此,可以在客户端 102a-102n 和始发服务器 106a-106B 之间通信期间的任何时刻实施图 6 所示的本发明的技术,而不论对象是否目前被缓存、之前已经被缓存或从未被缓存。

[0203] 在步骤 615,缓存管理器 232 将请求从客户端 102a 转送到始发服务器 106a。在一个实施例中,缓存管理器 232 向始发服务器转送该请求而不检查对象是否保存在缓存管理器 232 中。在某些实施例中,缓存管理器 232 首次检查客户端 102a 所请求的对象是否从缓存管理器 232 可得,并如果不可得,那么向始发服务器 106a 转送请求。在其他实施例中,缓存管理器 232 检查对象是否在缓存器中保存然而无效或另外需要更新,或另外将被无效或即将需要更新。在另一实施例中,缓存管理器 232 确定应该动态重新产生并从始发服务器 106a-160N 提供缓存器中保存的动态产生的对象。因此,如果缓存管理器 232 确定应该从始发服务器 106a-160n 请求对象,则缓存管理器 232 将请求从客户端 102a 转送到始发服务器 106a-160n。在一个实施例中,缓存管理器 232 向始发服务器 106a 转送从第一客户端 102a 接收的相同或原始请求。在另一实施例中,缓存管理器 232 发送对于对象由缓存管理器 232 形成的请求,在由第一客户端 102a 的请求中请求或识别该对象。

[0204] 当等待来自始发服务器 106a-160N 的对于第一客户端 102a 的请求的响应时和/或在响应第一客户端 102a 的请求之前,在步骤 620,缓存管理器 232 截取或另外接收对于对象的一个或多个另外请求,该对象在步骤 610 由第一客户端 102a 的首次请求被识别、与该首次请求相关联或被请求。例如,在一个实施例中,缓存管理器 232 从第二客户端 102B 接收对于对象的第二请求,所述对象是由始发服务器 106a 响应首次请求而动态产生的。在一个实施例中,装置 104 或缓存管理器 232 使用任何类型和形式的排队机制而排队第二请求和队列中的任何其他请求。在某些实施例中,对于对象的第二和/或另外请求能近似同时的发生或由缓存管理器 232 近似同时接收。在其他实施例中,可以在接收第一请求和从始发服务器 106a 接收对于第一请求的响应之间的任何时刻由缓存管理器 232 截取或接收第二和/或另外请求。在另一实施例中,在接收第一请求和向客户端 102a 提供对于第一请求的响应之间的任何时刻由缓存管理器 232 接收第二和/或另外请求。

[0205] 在一个实施例中,在响应第一请求之前,缓存管理器 232 并不转送来自第二客户端 102b 对于对象的第二请求和/或来自任何其他客户端 102c-102n 对于对象的任何另外请求。在一个实施例中,在步骤 625,缓存管理器 232 例如在装置 104 中排队或另外持有第二和另外请求,直到接收到对于第一请求的响应。因此,对于第二客户端 102B 透明的缓存管理器 232 并不转送第二客户端的请求。然而,在某些实施例中,对于第二客户端 102B 来说看来好像例如正在由始发服务器 106a-106n 处理。

[0206] 在某些实施例中,来自第二客户端 102a 的第二请求会请求与在步骤 610 的首次请求相同的对象,但还包括对于静态的、动态的或另外的另一对象或其他信息的另外请求。例如,第二客户端 102B 的请求包括多个请求。在这些实施例中,缓存管理器 232 识别与结合第

一客户端 102a 的首次请求已请求的对象相关的部分请求,并不处理所识别的部分直到接收到对于第一请求的响应。对于来自第二客户端 102B 的请求的其他部分,缓存管理器 232 以任何合适的形式(例如通过这里所述的本发明的任何技术)处理这些其他请求。例如,第二请求包括对于第二对象的请求,缓存管理器 232 为其请求来自始发服务器 102a-102B 的第二对象或获得来自缓存管理器 232 的第二对象。

[0207] 在步骤 625,本发明的方法 600 接收对于第一请求的响应,所述第一请求在步骤 615 转送到始发服务器 106a。在某些实施例中,响应包括对象,诸如由任何服务器 106a-106n 动态产生的对象。在其他实施例中,响应包括到由缓存管理器 232 提取或另外获取的对象位置的指针。在其他实施例中,响应表示某些错误消息或另外表示不能提供所请求的对象。在一个实施例中,缓存管理器 232 比较所接收的响应和对应的请求以确定响应对于请求是否合适或恰当。在其他实施例中,缓存管理器 232 获得响应(诸如动态产生的对象)的内容,并产生、修改或另外提供传送给第一客户端 102a 所需的响应。

[0208] 在步骤 630,本发明的缓存管理器 232 使用对于第一请求所接收的响应和/或响应中的对象以响应对于相同对象的第一请求和第二或另外请求。在一个实施例中,缓存管理器 232 响应第一请求向第一客户端 102a 传送所接收的响应,并响应第二请求向第二客户端 102B 传送所接收的响应。在另一实施例中,缓存管理器 232 向第一客户端 102a 传送所接收的响应,但修改所接收的响应并响应第二请求向第二客户端 102B 传送所修改的响应。例如,缓存管理器 232 修改对于第二客户端 102B 的响应以包括会话信息或对应于第二请求的其他信息。在其他实施例中,缓存管理器 232 获得从接收的响应所请求的对象,并产生、修改或另外提供对应于第一请求和第二请求并包括对象的响应,诸如第一响应和第二响应,并向第一客户端 102a 和第二客户端 102B 传送对应响应。因此,缓存管理器 232 在响应来自客户端 102a-102b 对于对象的请求中使用从始发服务器 106a-106n 接收的对于第一请求的响应的所有或任何部分。

[0209] 在某些实施例中,在从服务器 106a 接收对于第一请求的响应之后并在步骤 625 完成将响应传送给请求对象的第一客户端 102a、第二客户端 102b 或任何其他客户端 102a-102c 之前,在步骤 635 缓存管理器 232 从其他客户端 102c-102n 截取或另外接收对于对象的另外请求。例如,缓存管理器 232 从第三客户端 102C 接收对于对象的第三请求。在这些实施例中,缓存管理器 232 使用来自始发服务器 106a-106n 的对于第一请求所接收的并在步骤 630 传送到客户端 102a 和/或 102b 的所有或部分响应以响应这些另外请求。在另一实施例中,缓存管理器 232 已经完成将响应传送到第一客户端 102a 和第二客户端 102b,并仍有任何响应在存储器中或可另外获得以使用借助于第一请求提供的对象而响应第三客户端 102c 的第三请求。因此,在步骤 640,本发明的缓存管理器 232 可以响应第三请求或进一步的另外请求,而不需要再次请求来自始发服务器的对象和/或不需要在缓存器中存储所请求的对象。

[0210] 在某些实施例中,装置 104 或缓存管理器 232 排队第二请求、第三请求或队列中的任何其他请求。因此,在一个实施例中,当装置 104 或缓存管理器 232 代表服务器 106a-106n 而排队并响应请求时,客户端 102a-102n 不需要再次提交请求。在另一实施例中,装置 104 或缓存管理器 232 排队并响应对客户端 102a-102n 透明的请求而不向服务器 106a-106n 提交请求。装置 104 或缓存管理器 232 使用诸如排队、对象、或存储器中的数据结构中的任何机

制以排队来自客户端 102a-102n 的请求用于使用本发明的技术来响应。另外,装置 102 或缓存管理器 232 以任何顺序,而不需要以接收的顺序或来自队列的先入先出形式响应客户端请求。

[0211] 在步骤 645,缓存管理器 232 刷新来自始发服务器 106a 的对于第一请求的响应、对于客户端请求的任何响应和 / 或由来自缓存管理器 232 的请求所请求的对象。在一个实施例中,一旦完成传送最后响应,诸如在一个实施例中的步骤 630 或在另一实施例中的步骤 640,缓存管理器 232 立即从缓存器刷新响应和 / 或对象。在某些实施例中,缓存管理器 232 在预定时间段(诸如根据期限时间)维持对象。在其他实施例中,缓存管理器 232 在缓存器中维持对象直到始发服务器 106a-160N 无效对象。在另一实施例中,对象在存储器或储存器中维持直到为了其他目的重写或另外由缓存管理器 232 使用存储器或储存器。

[0212] 虽然上面关于从多个客户端对于单个对象的请求而一般性地描述了本发明的方法 500 和 600 的技术,但本领域的技术人员理解并认识到,本发明的闪速缓存器和闪速群处理技术可以用于处理来自多个客户端的单个请求或请求序列中的多个对象请求,其可以顺序、并发或近似并发或同时发生。因此,可以在网络环境 200 中缓存管理器 232 或装置 104 的操作期间以任何频率和以任何次数执行实施方法 500 和 600 的多个示例。而且,来自每个客户端的每个请求或请求序列会请求不同的对象集,一个或多个客户端请求的一个或多个对象具有公共集合。

[0213] 例如根据本发明的闪速群技术,第一客户端请求第一对象和第二对象,而第二客户端请求第二对象和第三对象,以及第三客户端请求第一对象和第三对象。在某些实施例中,本发明的缓存器使用多路复用和 / 或多路分解技术以对多个客户端之间突出的多个请求的每个对象请求一次对象,并相应地向每个客户端提供相应响应。另外对于上面示例,缓存器可以从始发服务器为第一客户端和第三客户端获取一次第一对象、为第一客户端和第二客户端获取一次第二对象以及为第二客户端和第三客户端获取第三对象。在某些实施例中,当缓存器接收对于客户端的一个或多个请求的所有对象时,缓存器向客户端传送相应响应或多个响应。在其他实施例中,一旦由缓存器接收对象,则缓存器使用每个对象响应客户端。

[0214] 类似地,根据本发明的方法 500 的闪速缓存器技术,装置的缓冲器包括等待传送的多个对象并由装置接收对于那些对象的另外请求。在一个实施例中,本发明的装置使用单个 TCP/IP 堆栈,从所述堆栈中多个缓冲器与多个对象相关联并在传送期间临时存储多个对象。因此,缓冲器中的第一对象用于响应来自第一客户端和第二客户端对于第一对象的请求,以及缓冲器中的第二对象用于响应来自第三客户端和第四客户端对于第二对象的请求,或响应来自第一客户端或第二客户端中任一个或两者对于第二对象的请求。

[0215] 而且,如本领域技术人员将明白和理解的,可以互相结合或组合而实施本发明的闪速缓存器和闪速群技术。例如,本发明的装置能通过闪速缓存器技术而提供对来自缓冲器对象的其他请求的响应,并确定缓冲器中的对象是无效的或另外需要从始发服务器请求。在这,装置会通过请求来自始发服务器的对象而转换到闪速群技术。然后,装置使用从始发服务器请求的对象以响应在装置中接收并排队的对象同时等待始发服务器响应的其他请求。

[0216] 12. Etag 插入

[0217] 另一方面,本发明针对使用技术以通过在对客户端的响应中插入诸如实体标签和缓存控制信息之类的信息而增加缓存器命中率以使缓存器能够检查在随后请求中的命中。本发明提供一种用于当始发服务器没有将对象识别为可缓存时将动态产生的对象识别为可缓存的技术。在某些实施例中,诸如处理对对象的 HTTP 请求和响应的实施例,本发明的技术在响应中插入实体标签或“etag”以对对象提供缓存控制,假定没有来自始发服务器的实体标签和 / 或缓存控制信息。简单参考本领域技术人员所知道的 HTTP 协议,“etag”是基于 HTTP 消息的 Etag 报头字段,其为诸如对象的请求变量提供实体标签的当前值,并用于与来自相同源的其他实体比较。Etag 提供在 HTTP 协议中有效的机制。同样,基于 HTTP 的消息包括缓存 - 控制报头字段以指定指示用于缓存机制以及请求 / 响应消息或通信链。缓存 - 控制指示指定所需的缓存行为。

[0218] 现在参考图 7A 和 7B 并根据图 1 和 2,步骤用于表示实施本发明的“Etag”技术的实施例。简单概括图 7A、本发明的方法 700,步骤 710 接收具有由客户端 102a-102n 所请求的对象的响应,该对象是诸如从始发服务器 106a-106n 提供的并由装置 104 的缓存管理器 232 接收的对象。在步骤 715,缓存管理器 232 确定来自响应的对象是否在标签实体控制下或另外具有实体标签。如果对象不具有实体标签,那么在步骤 720,缓存管理器 232 为对象产生实体标签。在步骤 725,缓存管理器 232 修改响应以提供包括缓存器所产生的实体标签的实体控制信息。在步骤 730,缓存管理器 232 向缓存管理器 232 存储具有实体标签信息的对象。在步骤 735,缓存管理器 232 使用修改的响应而响应请求对象的客户端 102a-102n。因此,客户端 102a-102n 以对客户端 102a-102n 透明的形式从缓存管理器 232 接收实体标签控制对象,以至客户端 102a-102n 没有意识到始发服务器 106a-106n 没有提供具有实体标签信息的对象。

[0219] 在进一步的描述中,在本发明的步骤 710,缓存管理器 232 针对客户端 102a-102n 对对象的请求接收来自始发服务器 106a-106n 的响应。在某些实施例中,从客户端 102a-102n 发送到服务器 106a-106n、或从代表客户端 102a-102n 的缓存管理器 232 发送到服务器 106a-106n 的请求中接收响应。在一个实施例中,在为了匹配和 / 或有效对象检查缓存管理器 232 之后缓存管理器 232 从客户端 102a-102n 到服务器 106a-106n 转送请求。在另一实施例中,缓存管理器 232 在实施结合图 6 所述的本发明的闪速群技术期间请求来自服务器 106a-106n 的对象。

[0220] 在方法 700 的步骤 715,缓存管理器 232 确定在步骤 710 响应中接收的对象是否在实体标签控制下或具有实体标签。在某些实施例中,缓存管理器 232 检查、检验、验证或另外读取提供响应的一个或多个分组的任何字段、报头、数据或其他信息。如本领域技术人员应该认识到并理解的,要检查或读取的一个或多个分组的一部分取决于用于响应的一个或多个协议的类型。在一个实施例中,缓存管理器 232 确定响应分组的实体标签部分正丢失、为空或另外没有提供。在另一实施例中,缓存管理器 232 确定响应分组的实体标签部分被破坏、无效或另外不能由缓存管理器 232 和 / 或客户端 102a-102B 使用。在某些实施例中,缓存管理器 232 确定响应具有实体标签或在实体控制下但不是以缓存管理器 232 所需的类型、形式或方式。除了实体标签或替代实体标签,缓存管理器 232 以类似于实体标签的形式确定是否在响应中存在所需的缓存 - 控制信息。在其他实施例中,缓存管理器 232 通过策略引擎 236 确定对象是否应该在实体标签和 / 或缓存控制下。因此,在某些实施例中,



如果装置 104 通过策略引擎 236 确定对象不应该在实体标签或缓存控制下,则缓存管理器 232 不继续执行本发明的 etag 技术,诸如步骤 720 至 725。

[0221] 在步骤 720,缓存管理器 232 为所接收响应的对象产生所需的实体标签和缓存控制信息。Etag 包含任何类型和 / 或形式的实体标签表示并在一个或多个字符中包含任何数字、文字数字、字母。在 HTTP 的实施例,etag 包含字符串或引用的字符串,并进一步包含前缀用于根据 HTTP 协议表示 etag 是弱的或强的。在某些实施例中,缓存管理器 232 使用实体报头、或 etag 计数器,其顺序地或以任何所需增量增加 etag 计数器。在某些实施例中,etag 计数器对于所有对象是全局的,缓存管理器 232 为该所有对象产生 etag。在其他实施例中,缓存管理器 232 具有多个 etag 计数器,每个 etag 计数器与对象或对象组相关联或特定于对象或对象组。

[0222] 在一个实施例中,etag 包含与对象相关联的通用标识符 (UID) 或全局通用标识符 (GUID)。在另一实施例中,etag 包含之前上文所述的具体数。在某些实施例中,缓存管理器 232 从在装置 104 上运行的应用、程序或其他形式可执行指令获取 etag,或在其他实施例中,从通过网络由装置 104 可访问的另一系统或设备中获得 etag。在其他实施例中,缓存管理器 232 使用任何合适的算法、商业规则或逻辑以便为对象产生所需的唯一 etag。在一个实施例中,装置 104 包含数据库、文件、或其他有组织的存储元件,用于为缓存管理器 232 产生并维持对象的 etag。

[0223] 在方法 700 的步骤 725,缓存管理器 232 修改来自始发服务器 106a-106n 的响应以包括所需的实体标签和 / 或缓存控制信息,诸如在步骤 720 由缓存器产生的实体标签。在一个实施例中,缓存管理器 232 经由协议 (例如,HTTP 协议) 的实体标签报头插入实体标签信息。在另一实施例中,缓存管理器 232 修改响应中的任何实体标签报头字段以包括所需的实体标签。在又另一实施例中,缓存管理器 232 用缓存管理器 232 所需使用的实体标签字段代替响应中的实体标签字段。在其他实施例中,将实体标签信息插入、修改或放入一个网络分组或多个网络分组的任何部分中用于将响应传送到客户端 102a-102n。

[0224] 另外,步骤 725,缓存管理器 232 插入、修改或另外放入任何所需的缓存控制信息在协议的任何报头、字段或网络分组的任何其他部分中用于传送响应。由缓存控制器 232 通过任何合适的装置和 / 或机制产生该缓存 - 控制信息。在一个实施例中,缓存 - 控制信息被配置并通过设备 104 的策略引擎 236 获取。在另一实施例中,缓存管理器 232 根据缓存管理器 232 的任何算法、商业规则或逻辑确定所需的缓存 - 控制信息。在某些实施例中,缓存 - 控制信息可以基于客户端 102a-102n 和装置 104 或缓存管理器 232 之间的网络和缓存性能的任何历史。

[0225] 在本发明的方法 700 的步骤 730,缓存管理器 232 以任何合适或所需的方式在缓存管理器 232 中存储产生的实体标签和对象。在某些实施例中,实体标签包括在存储对象的一部分中或成为存储对象的一部分。在其他实施例中,单独地但与对象相关联保存实体标签。在另一实施例中,如本领域技术人员所知道的,通过数据库、链接列表、查询表或用于将一个实体和另一个实体相关联的任何其他机制,使实体标签与对象相关联。另外,在某些实施例中,使用对象和 / 或实体标签所提供的缓存控制信息还被存储以与缓存管理器 232 中的对象相关联。

[0226] 在步骤 735,缓存管理器 232 使得装置 104 向客户端 106a-106n 传送修改的响

应。例如,缓存管理器 232 请求分组处理引擎 240 以向客户端 102a-102n 传送修改的响应。因此,客户端 102a-102n 接收具有实体标签和缓存器控制信息的对象,即使对于客户端 102a-102n 未知,但在一个实施例中,始发服务器 106a-106n 并不产生该信息,或者缓存管理器 232 改变由服务器 106a-106n 产生的该信息。本发明的 etag 技术可以确保每个对象具有确认符、即 etag 或具有由缓存管理器 232 控制或另外需要的 etag。如果对象不具有确认符或所需的确认符,那么缓存管理器 232 就插入它自己的 etag。这些 etag 确认符可以避免对重复请求对象提供完整的响应,正如结合图 7B 所讨论的。

[0227] 图 7B 描述了用于使用实体标签以由客户端 102a-102n 检查对象是否被修改的本发明的方法 750。因此,客户端 102a-102n 可以缓存客户端 102a-102n 上的对象并利用缓存管理器 232 或始发服务器 106a-106n 检查对象是否被改变。简单概括方法 750,在步骤 775,缓存管理器 232 接收来自客户端 102a-102n 的请求以检查对象的实体标签。在步骤 760,缓存器比较从客户端 102a-102n 的请求所接收的实体标签和缓存器中存储的对象的实体标签。在步骤 765,缓存管理器 232 确定来自客户端的实体标签是否匹配缓存器中的对象的实体。如果实体标签不匹配,那么在步骤 770,缓存管理器 232 使用为对象提供当前实体标签的响应来响应客户端 102a-102n。如果实体标签匹配,那么在步骤 775,缓存管理器 232 使用表示对象没有被修改的响应来响应客户端 102a-102n。

[0228] 在进一步的描述中,在步骤 755,缓存管理器 232 通过任何类型和 / 或形式的通信接收来自客户端 102a-102n 的请求以检查或确认对象的实体标签。在一个实施例中,请求包含具有 If-Match 或 If-None-Match 请求报头字段的 HTTP 消息,所述报头字段包括由客户端 102a-102n 提供的实体标签字符串值。例如,客户端 102a-102n 在客户端缓存器中具有对象的副本或另外在客户端 102a-102n 上可得到。客户端 102a-102n 使用请求中对象的客户端副本的实体标签以确定或确认对象是否是当前的。

[0229] 在步骤 760,缓存管理器 232 比较从客户端 102a-102B 接收的实体标签值和对象的实体标签以确定缓存管理器 232 是否具有与客户端 102a-102n 不同的对象版本。缓存管理器 232 使用任何合适的装置和 / 或机制以比较来自客户端 102a-102n 的实体标签值和缓存对象的实体标签值。在一个实施例中,缓存管理器 232 对于来自请求的 etag 值和与对象一起保存在缓存器中的 etag 执行位比较。在另一实施例中,缓存管理器 232 对实体标签和 / 或缓存对象执行并使用散列算法以确定对象是否被修改。在其他实施例中,缓存管理器 232 请求来自服务器 106a-106n 的对象,并比较所接收的对象和缓存的对象以确定对象是否被修改。

[0230] 如果在步骤 765,来自客户端 102a-102n 的实体标签不匹配与对象一起保存在缓存管理器 232 中的实体标签,则在步骤 770,缓存管理器 232 向客户端发送一个响应,该响应表示实体标签不匹配或另外对象已经被修改。在一个实施例中,缓存管理器 232 向客户端 102a-102n 发送一个响应,该响应为对象提供新实体标签值。在另一实施例中,缓存管理器 232 向客户端 102a-102n 发送一个响应,该响应提供新版本的对象、或修改的对象,诸如具有在缓存管理器 232 中保存的新实体标签的修改对象。如果在步骤 765,来自客户端 102a-102n 的实体标签匹配与对象一起保存在缓存管理器 232 中的实体标签,那么在步骤 775,缓存管理器 232 向客户端发送一个响应,该响应表示实体标签的确匹配或另外对象没有被修改。

[0231] 虽然参考 HTTP 协议一般性地描述了本发明的 etag 技术,但是本领域技术人员应该认识到并理解可以使用任何类型和 / 或形式的协议实施本发明以提供实体标签和缓存信息。在某些实施例中,协议不支持实体标签和缓存信息字段,而在其他实施例中,协议支持实体标签和缓存信息字段。在任何这些实施例中,当服务器不在响应中提供该信息时或当缓存器决定以装置 104 控制或所需的方式改变或控制服务器提供的实体和缓存控制信息时,本发明的技术提供实体标签和缓存控制信息。

[0232] 为了解决动态缓存由原始服务器和应用动态产生的内容时产生的挑战,本发明的缓存装置可以对诸如 Outlook® Web Access、Oracle® Configurator 和 Serena® Team Track® 之类的特定常用的应用包含预封装的策略。使用对于这些应用处理数据的方式的理解设计这些封装,因此能使缓存器关于由在缓冲存储器中保存的该应用产生哪个对象并且保持该对象多久作出更明智的决定。

[0233] D. 示例性的基于计算机系统的实施例

[0234] 本发明的功能可以使用硬件、软件或其组合来实施或在一个或多个计算设备或其他处理系统中实施。例如,图 8A 和 8B 描述了用于实施本发明的任何技术、方法和功能的示例计算设备 800。因此,本发明的装置 104 可以是计算设备 800 或经设计和构造的特定目的的设备以提供这里所述本发明的任何技术、功能和操作。

[0235] 图 8A 和 8B 描述了计算设备 800 的框图,并在某些实施例中,也被称为网络设备 800,用于实施本发明的实施例。如图 8A 和 8B 所示,每个计算设备 800 包括中央处理单元 802 和主存储器单元 822。如图 8A 所示,常规计算设备 800 包括可视显示设备 824、键盘 826 和 / 或诸如鼠标的指示设备 827。每个计算设备 800 也包括诸如一个或多个输入 / 输出设备 830a-830b (一般使用附图标记 830) 之类的另外可选组件,以及与中央处理单元 802 通信的缓冲存储器 840。

[0236] 中央处理单元 802 是响应并处理从主存储器单元 822 提取的指令的任何逻辑电路。在许多实施例中,中央处理单元由微处理器单元提供,诸如:由加利福尼亚的芒廷维尤的 Intel 公司制造的产品;由伊利诺斯州的绍姆堡的 Motorola 公司制造的产品;由加利福尼亚州的圣克拉拉的 Transmeta 公司制造的产品;由纽约的怀特普莱恩斯的国际商业机器公司制造的产品;或那些由加利福尼亚州的桑尼维尔的 Advanced Micro Devices 制造的产品。计算设备 800 可以是基于任何这些处理器,或能够如上述操作的任何其他处理器。

[0237] 主存储器单元 822 可以是能够存储数据并允许直接由微处理器 802 直接访问任何存储位置的一个或多个存储器芯片,诸如静态随机存取存储器 (SRAM)、突发 SRAM 或同步突发 SRAM (BSRAM)、动态随机存取存储器 (DRAM)、快速页面模式 DRAM (FPM DRAM)、增强的 DRAM (EDRAM)、扩展数据输出 RAM (EDO RAM)、扩展数据输出 DRAM (EDO DRAM)、突发扩展数据输出 DRAM (BEDO DRAM)、增强的 DRAM (EDRAM)、同步 DRAM (SDRAM)、JEDEC SRAM、PC100 SDRAM、双数据率 SDRAM (DDR SDRAM)、增强的 SDRAM (ESDRAM)、同步链接 DRAM (SLDRAM)、直接 Rambus DRAM (DRDRAM) 或铁电随机存取存储器 (FRAM)。主存储器 822 可以是基于任何上述的存储器芯片、或能够如这里所述操作的任何其他可获得的存储器芯片。在图 8A 所示的实施例中,处理器 802 通过系统总线 850 与主存储器 822 通信(下文将详细描述)。图 8A 描述了其中处理器通过存储器端口 803 直接与主存储器 822 通信的计算设备 800 的实施例。例如,在图 8B 中,主存储器 822 可以是 DRDRAM。

[0238] 图 8B 描述了其中主处理器 802 通过第二总线直接与缓冲存储器 840 通信的实施例,第二总线被称为后端总线。在其他实施例中,主处理器 802 使用系统总线 850 与缓冲存储器 840 通信。缓冲存储器 840 一般比主存储器 822 具有更快的响应时间并一般由 SRAM、BSRAM 或 EDRAM 提供。

[0239] 在图 8A 所示的实施例中,处理器 802 通过局部系统总线 850 与各种 I/O 设备 830 通信。各种总线用于将中央处理单元 802 连接到任何 I/O 设备 830,包括 VESA VL 总线、ISA 总线、EISA 总线、微通道结构(MCA)总线、PCI 总线、PCI-X 总线、PCI 快速总线或 NuBus。对于其中 I/O 设备是视频显示器 824 的实施例,处理器 802 使用改进图形端口(AGP)与显示器 824 通信。图 8B 描述了其中主处理器 802 通过 HyperTransport、快速 I/O 或 InfiniBand 直接与 I/O 设备 830b 通信的计算机 800 的实施例。图 8B 还描述了其中混合局部总线和直接通信的实施例:处理器 802 使用局部互连总线与 I/O 设备 830a 通信同时直接与 I/O 设备 830b 通信。

[0240] 计算设备 800 可以支持任何合适的安装设备 816,例如用于接收诸如 3.5 英寸盘、5.25 英寸盘或 ZIP 盘之类的软盘的软盘驱动器、CD-ROM 驱动器、CD-R/RW 驱动器、DVD-ROM 驱动器、各种格式的磁带驱动器、USB 设备、硬盘驱动器或适于安装诸如软件 820 或其部分之类的软件和程序的任何其他设备,所述软件或其部分涉及本发明或另外提供本发明的任何技术。计算设备 800 进一步包括存储设备 828,诸如一个或多个硬盘驱动器或独立冗余磁盘阵列,用于存储操作系统和其他相关软件、并用于存储诸如涉及本发明的软件 820 的任何程序之类的应用软件程序。可选的,任何安装设备 816 可以用作存储设备 828。

[0241] 而且,计算设备 800 包括网络接口 818,以通过各种连接而对接到局域网(LAN)、广域网(WAN)或因特网,所述连接包括但不限于标准电话线、LAN 或 WAN 链接(例如,802.11、T1、T3、56kb、X.25)、宽带连接(例如,ISDN、帧延迟、ATM)、无线连接、或上述任何或所有连接的某些组合。网络接口 818 包括内置网络适配器、网络接口卡、PCMCIA 网络卡、卡总线网络适配器、无线网络适配器、USB 网络适配器、调制解调器或任何其他设备,该任何其他设备适于将计算设备 800 对接到能够通信的任何类型网络,并且执行上述操作。

[0242] 大量 I/O 设备 830a-830n 存在于计算设备 800 中。输入设备包括键盘、鼠标、跟踪区、滚动球、麦克风和绘画写字板。输出设备包括视频显示器、扬声器、喷墨打印机、激光打印机和染料升华打印机。如图 8A 所示,I/O 设备由 I/O 控制器 823 控制。I/O 控制器控制诸如键盘 826 和指示设备 827 之类的一个或多个 I/O 设备,例如鼠标或光笔。而且,I/O 设备也为计算设备 800 提供存储设备 828 和 / 或安装介质 816。在其他实施例中,计算设备 800 提供 USB 连接以容纳手持式 USB 存储设备,诸如由加利福尼亚洲的 los Alamitos 的 Twintech 工业公司制造的设备的 USB 闪速驱动线。

[0243] 在其他实施例中,I/O 设备 830 可以是系统总线 850 和外部通信总线之间的电桥 870,所述外部通信总线诸如 USB 总线、Apple Desktop 总线、RS-232 串行连接、SCSI 总线、FireWire 总线、FireWire 800 总线、以太网总线、AppleTalk 总线、千兆位以太网总线、异步传输模式总线、HIPPI 总线、超 HIPPI 总线、serialPlus 总线、SCI/LAMP 总线、FibreChannel 总线或串行附加小型计算机系统接口总线。

[0244] 图 8A 和 8B 中所述的种类的计算设备 800 一般在操作系统的控制下操作,其控制任务的调度并访问系统资源。计算设备 800 可以运行任何操作系统,诸如任何版本的

Microsoft® Windows 操作系统、不同版本的 Unix 和 Linux 操作系统、用于 Macintosh 计算机的任何版本的 Mac OS®、任何嵌入式操作系统、任何网络操作系统、任何实时操作系统、任何开源操作系统、任何私有操作系统、用于移动计算设备或网络设备的任何操作系统、或能够在计算设备上运行并执行这里所述操作的任何其他操作系统。一般的操作系统包括:WINDOWS 3.x、WINDOWS 95、WINDOWS 98、WINDOWS 2000、WINDOWS NT 3.51、WINDOWS NT 4.0、WINDOWS CE 以及 WINDOWS XP,所有这些由华盛顿州雷蒙德的微软公司制造;由加利福尼亚州库珀蒂诺的苹果公司制造的 MacOS;由纽约的 Armonk 的国际商用机器公司制造的 OS/2;以及由犹他州的盐湖城的 Caldera 公司发布的免费可得的操作系统, Linux;或它们中的任何类型和 / 或形式的 Unix 操作系统。

[0245] 在其他实施例中,计算设备 800 具有与设备一致的不同处理器、操作系统和输入设备。计算设备 800 可以是任何工作站、桌上型电脑、膝上型电脑或笔记本计算机、服务器、手持计算机、移动电话、任何其他计算机、或能够通信并具有足够处理器能力和存储器能力以执行这里所述本发明操作的其他形式的计算或电信设备。而且,计算设备 800 可以是任何类型和 / 或形式的网络设备、诸如远程访问设备、虚拟专用网络(VPN)设备、安全套接层(SSL)VPN 设备、路由器、交换机、电桥、或能够执行这里所述本发明的操作的任何形式的其他网络设备。

[0246] 虽然一般性地讨论了本发明的装置使用 TCP/IP 或借助 TCP/IP 通信,但是该装置可以使用任何修改的传输控制协议或与任何修改的传输控制协议通信,所述协议诸如事务 TCP (T/TCP)、具有选择确认的 TCP (TCP-SACK)、具有大窗口的 TCP (TCP-LW)、诸如在 TCP-Vegas 协议中的拥塞预测和 TCP 欺骗。此外,本发明的装置可以使用任何节点间或高性能协议。而且,本发明的装置可以使用任何其他传输和网络协议或利用任何其他传输和网络协议操作,所述协议诸如在因特网分组交换(IPX)协议上的序列分组交换(SPX)协议。

[0247] 结论

[0248] 虽然上面已描述了本发明的各种实施例,但应该理解它们仅仅是示例性的而不是限制性。因此,本领域技术人员应该理解在不背离附加权利要求限定的本发明的精神和范围的情况下可以在形式和细节上进行各种变化。因此,本发明的宽度和范围不应该由上述示例性实施例限制,而只应当根据下面的权利要求书和其对等物来限定。

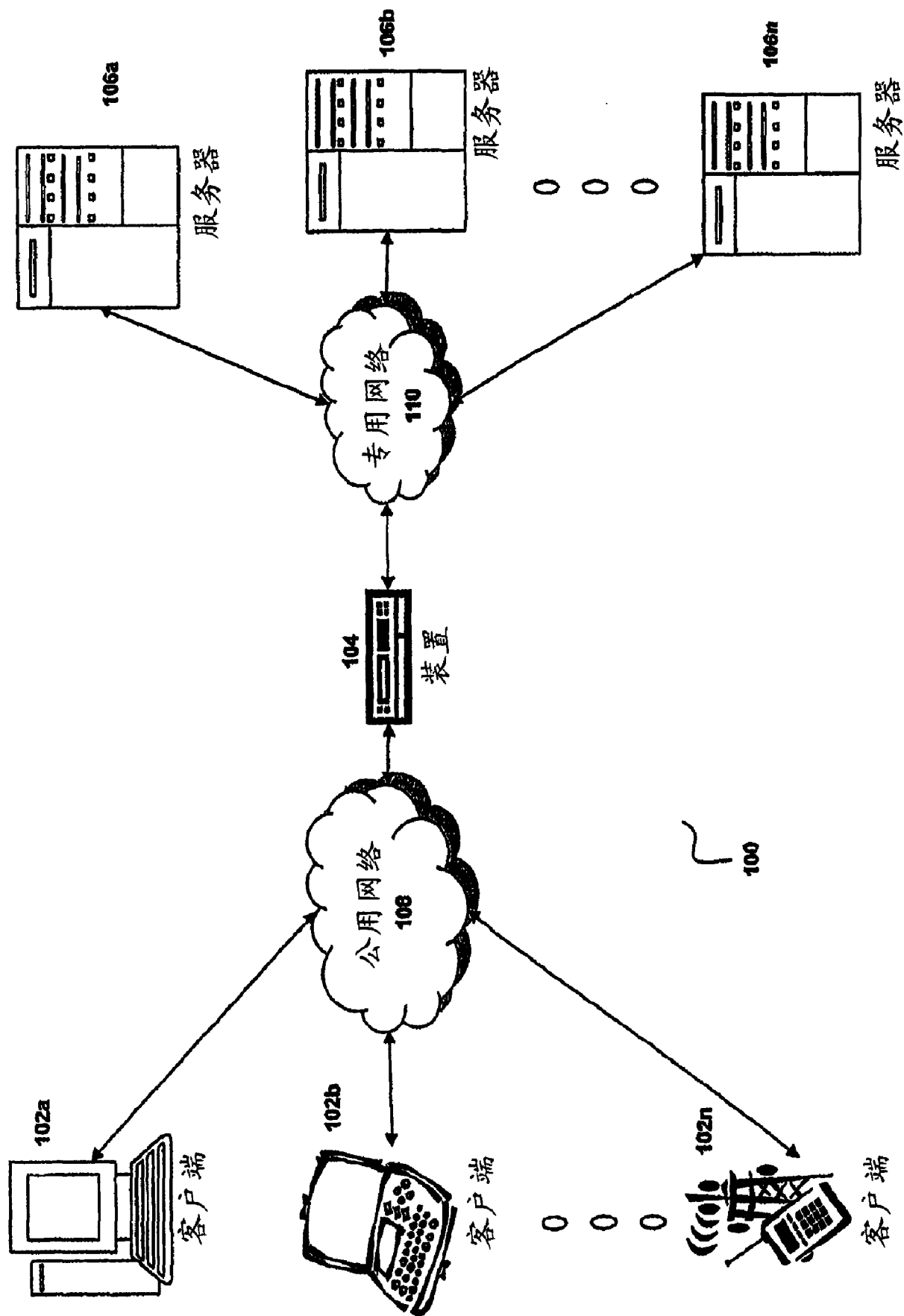


图 1

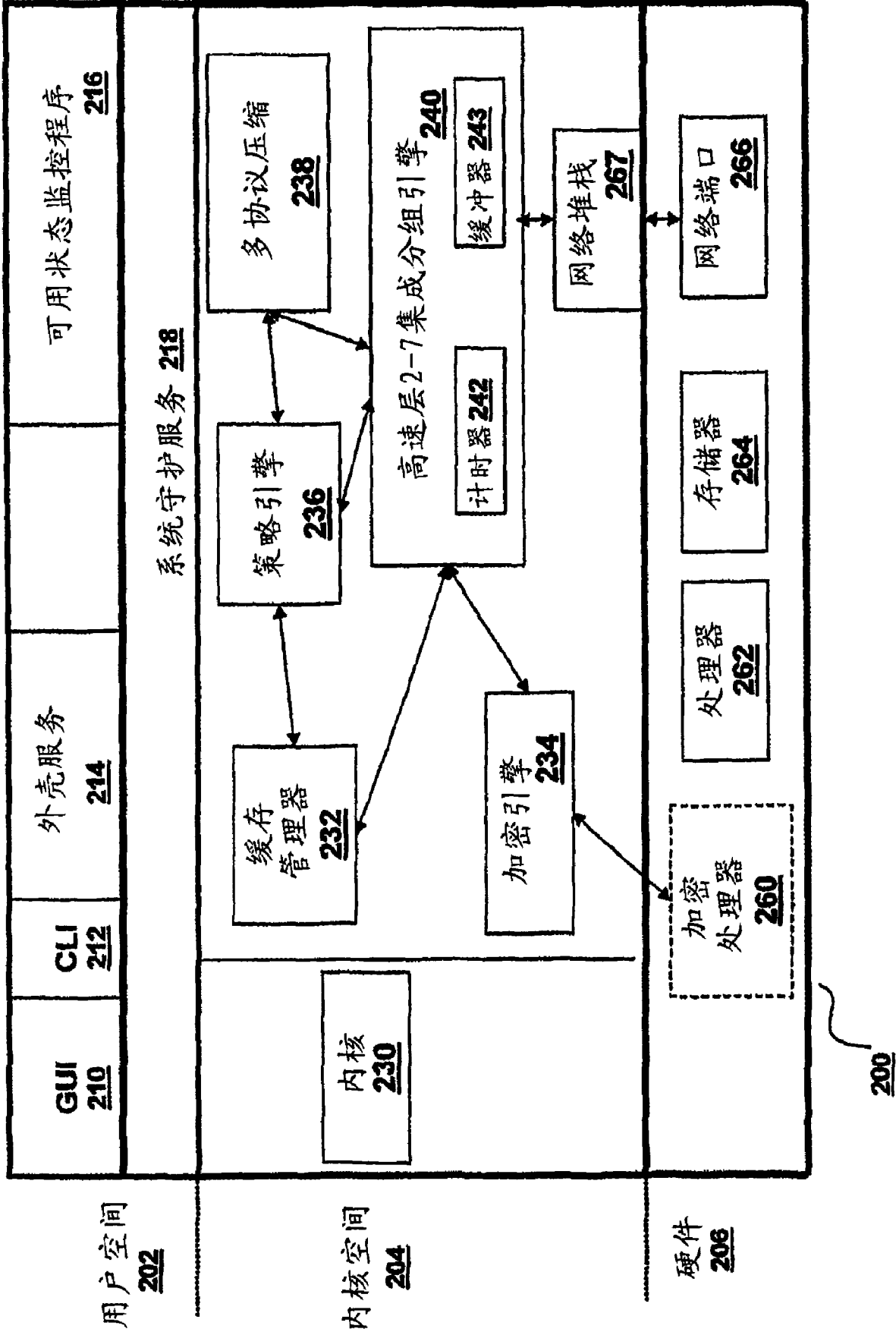
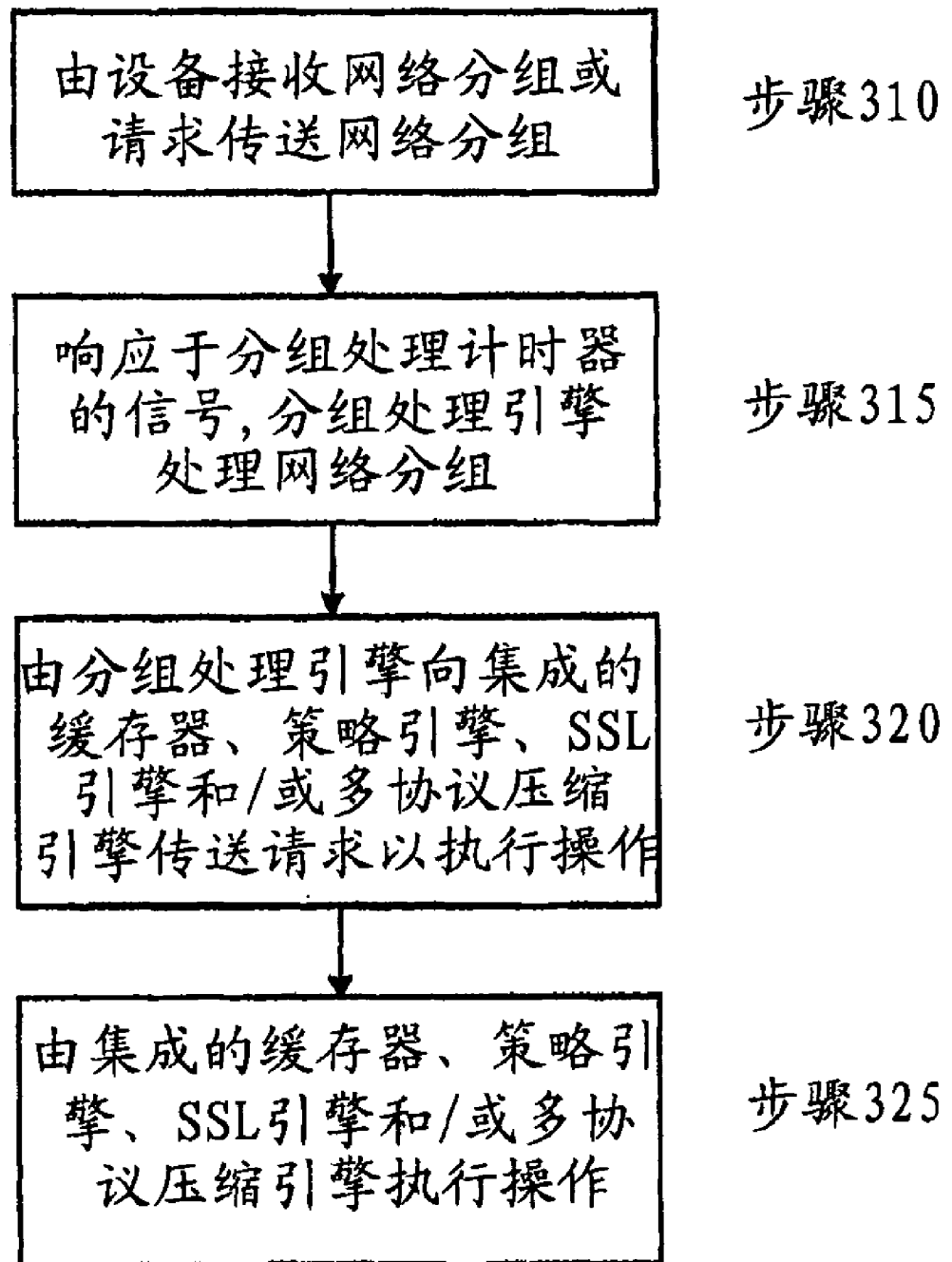


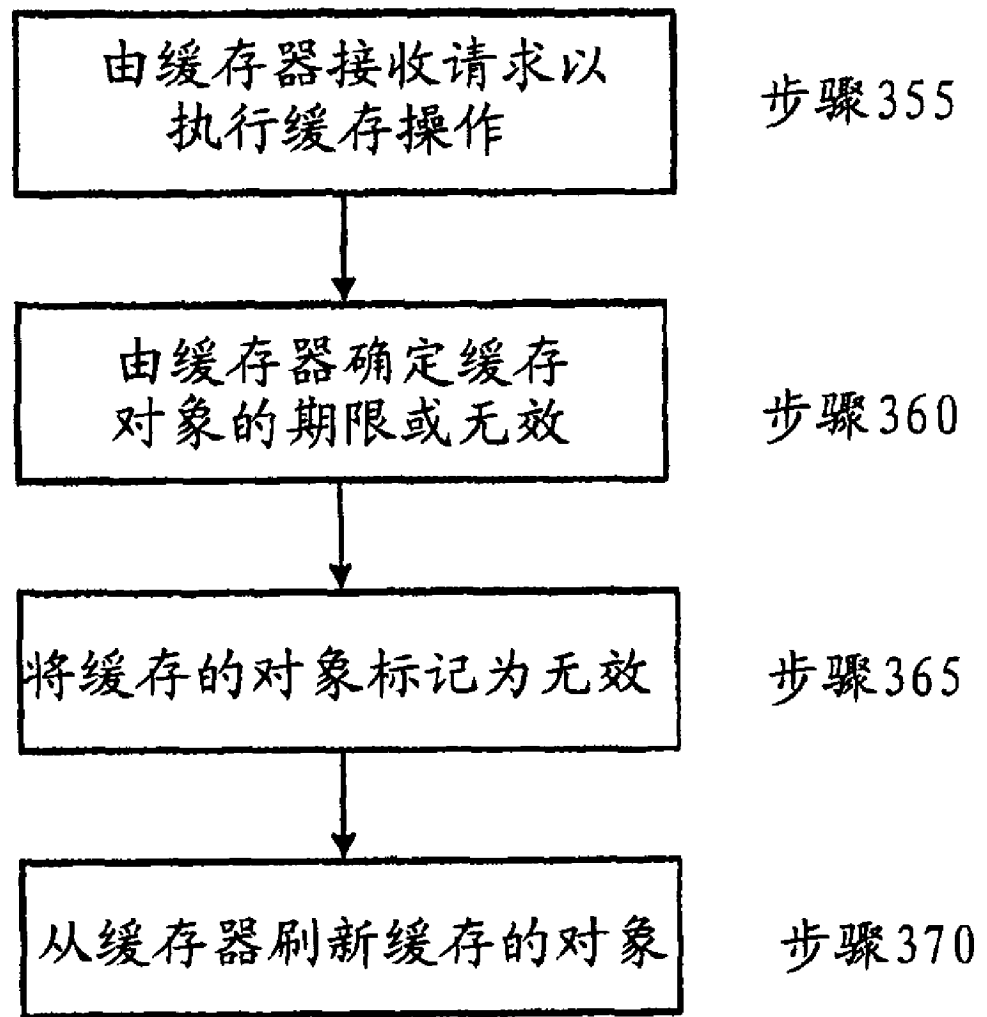
图 2



300

图 3A





350

图 3B

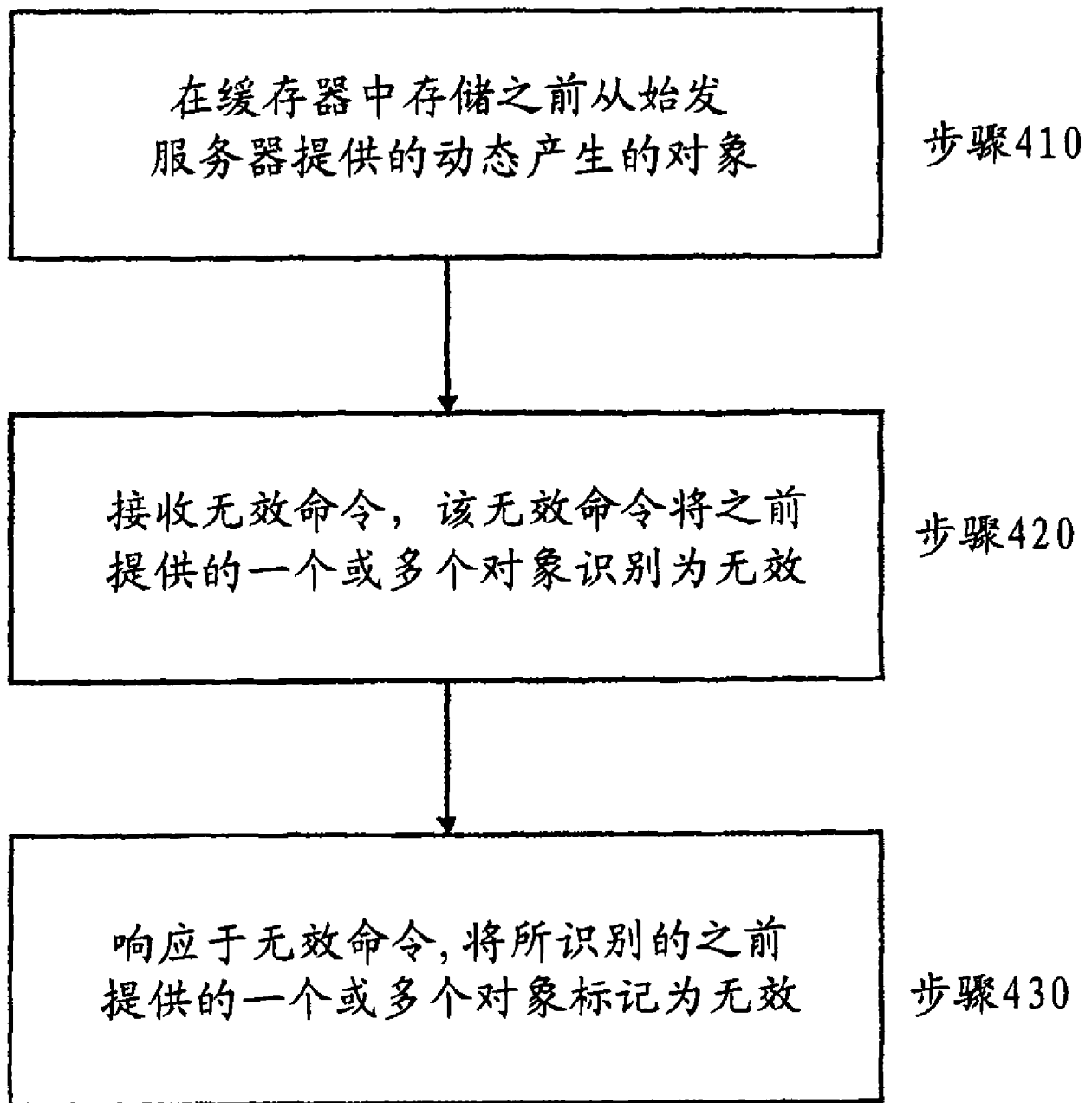


图 4A

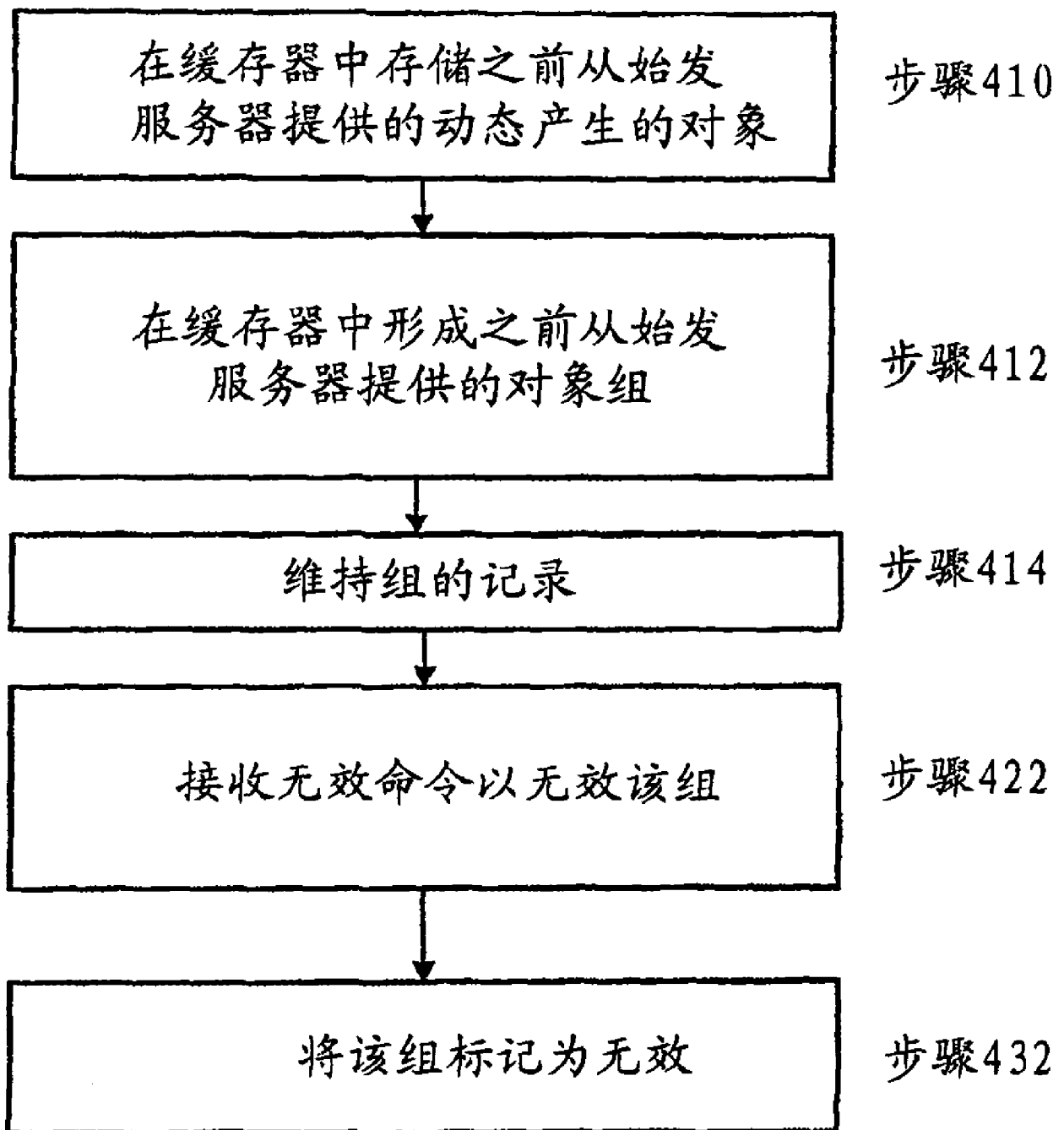


图 4B

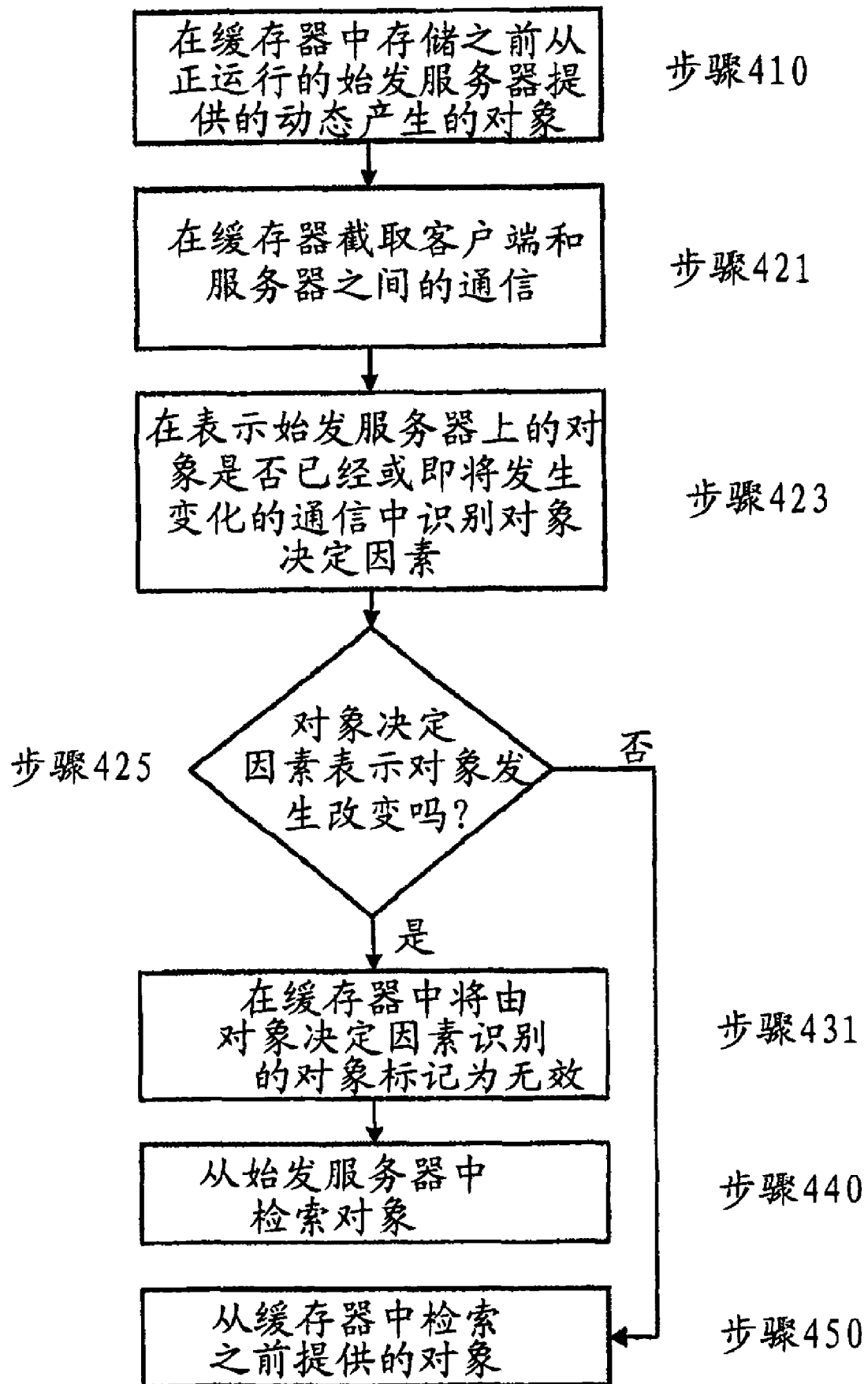


图 4C

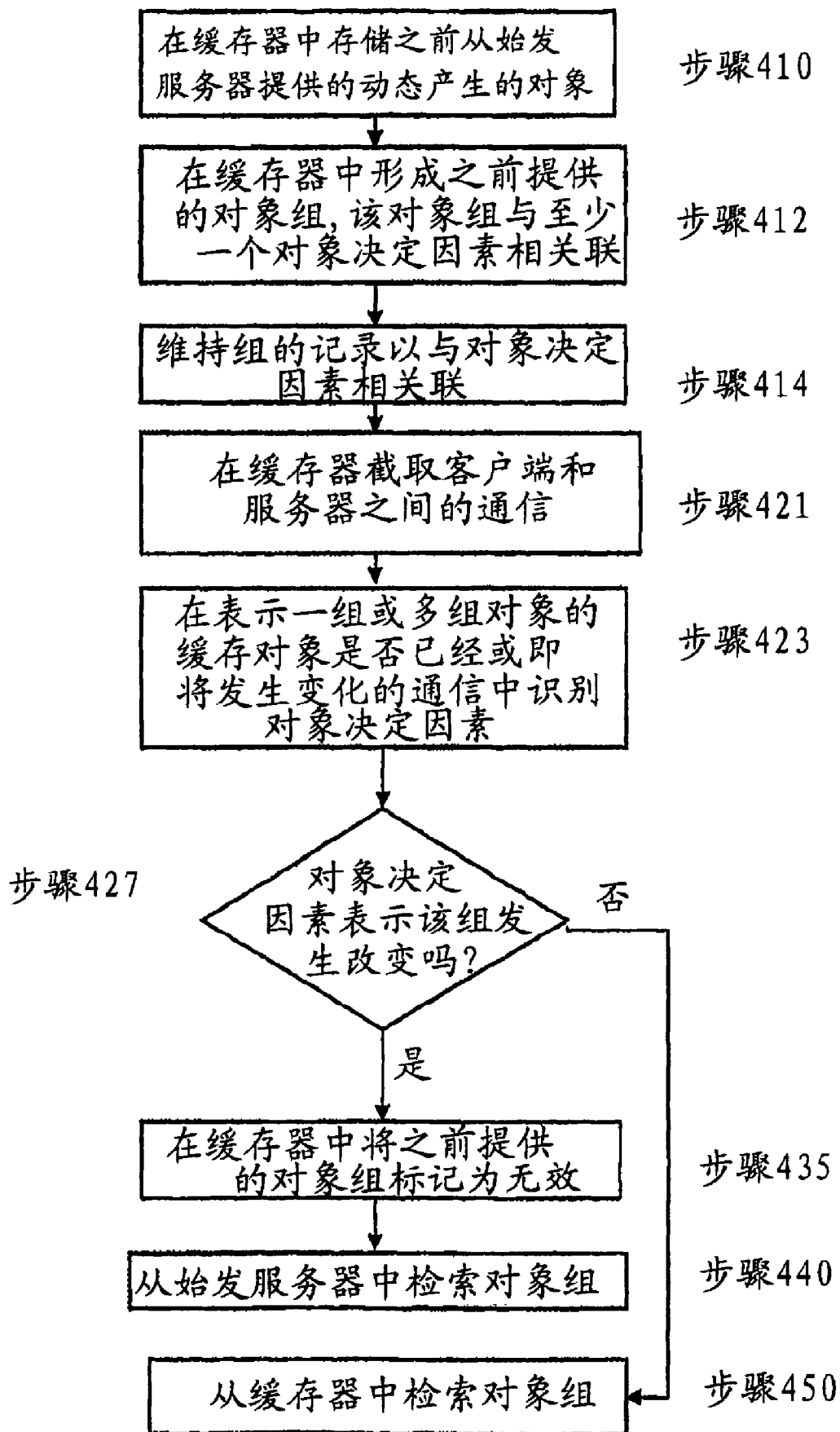
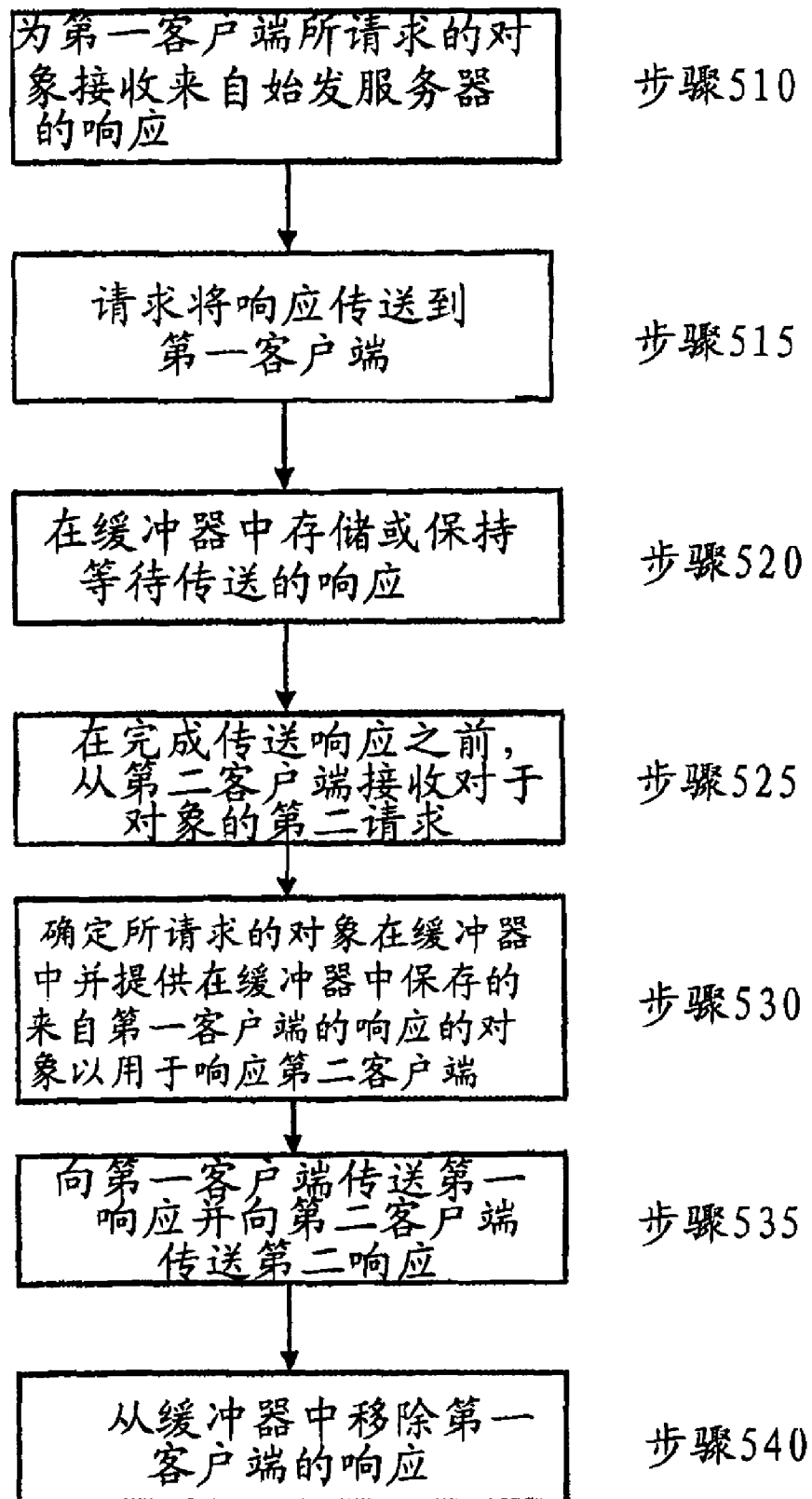
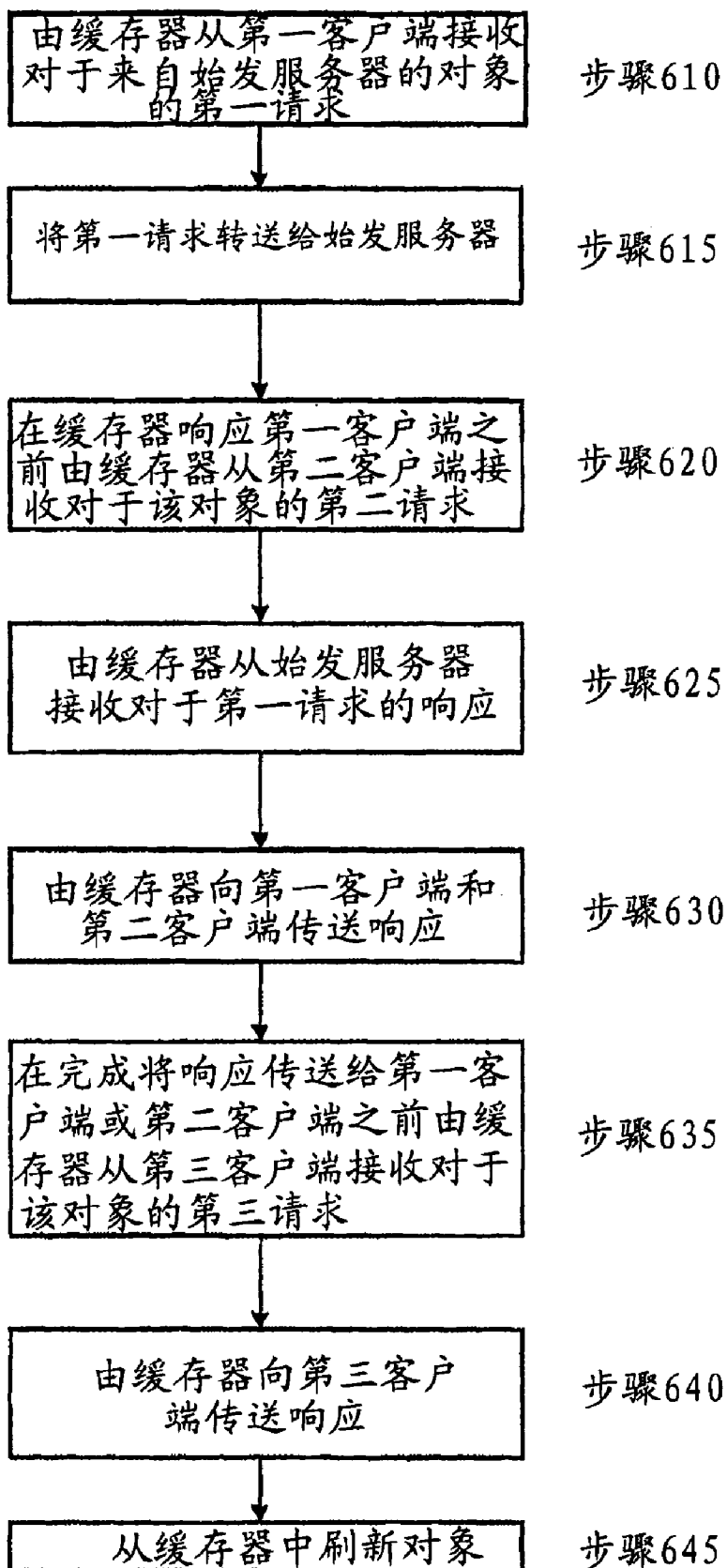


图 4D



500

图 5



600

图 6

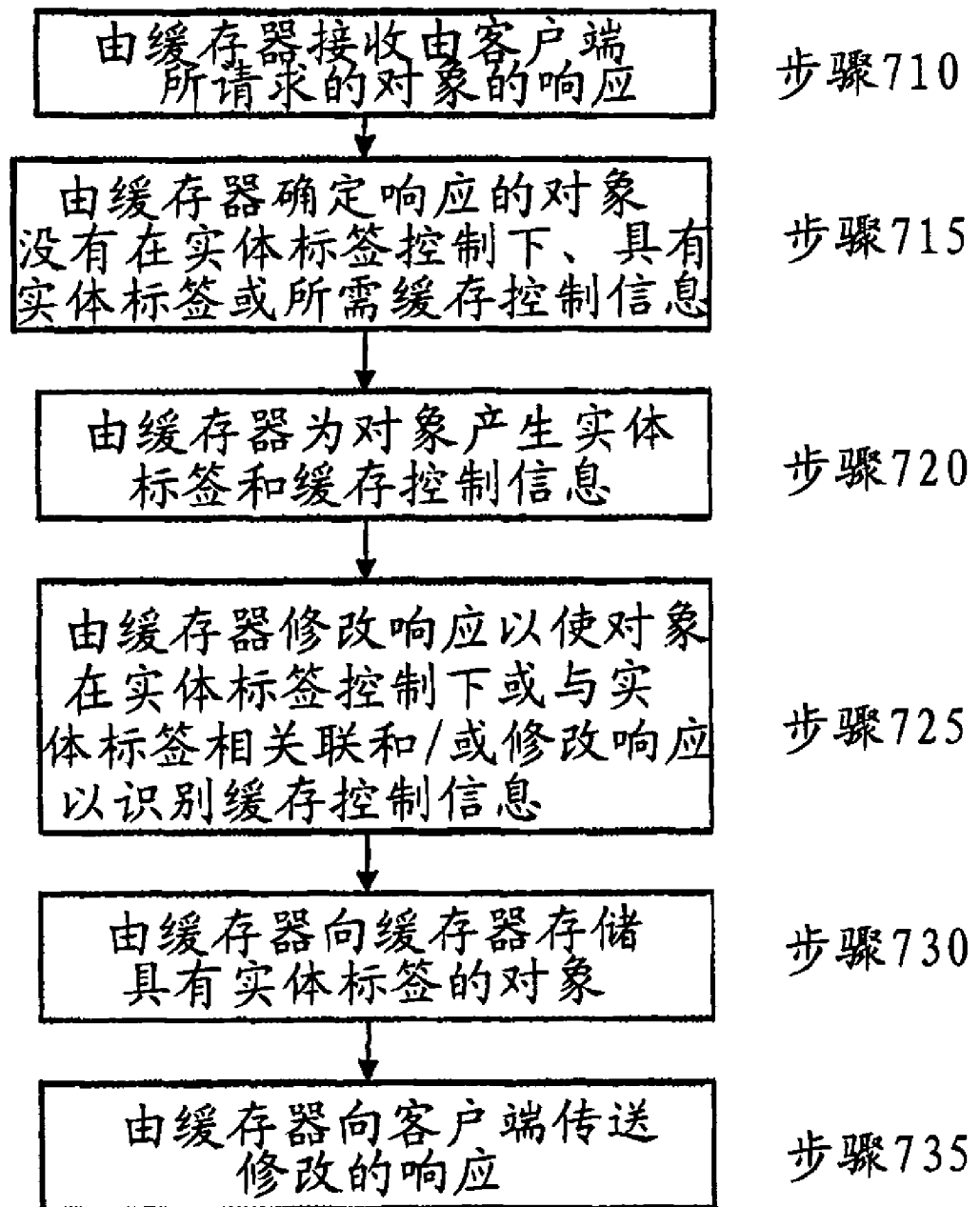


图 7A



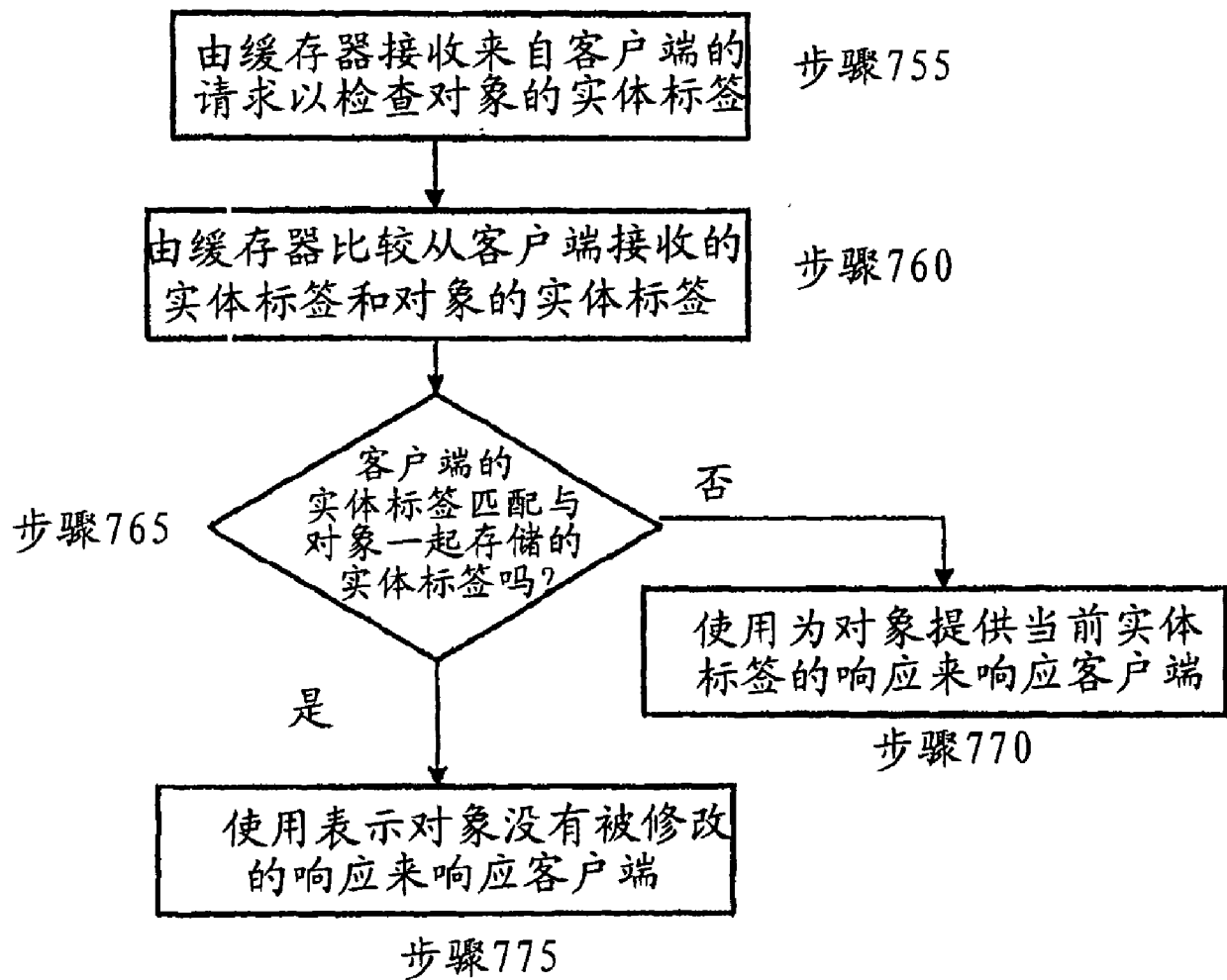
750

图 7B

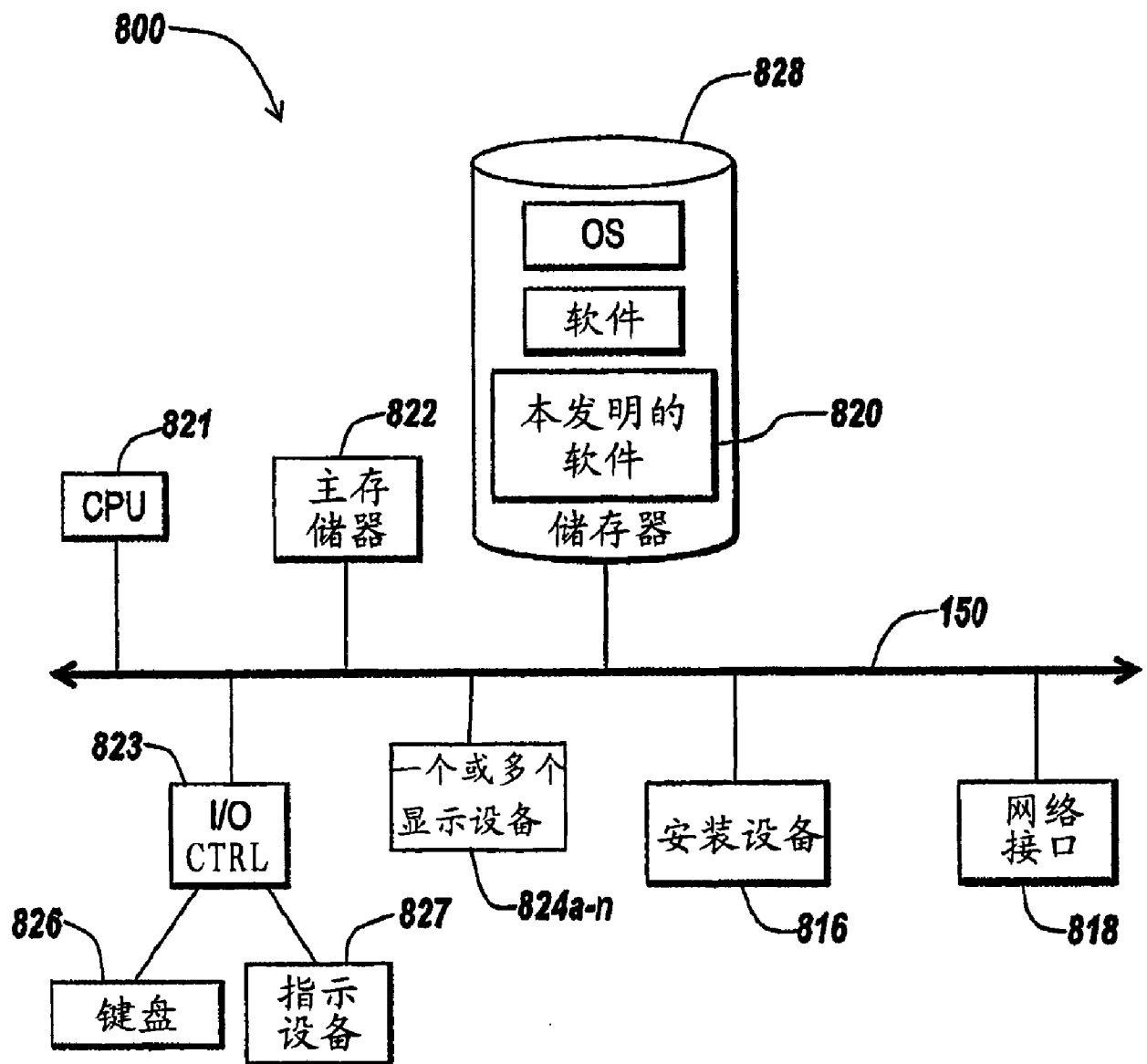


图 8A

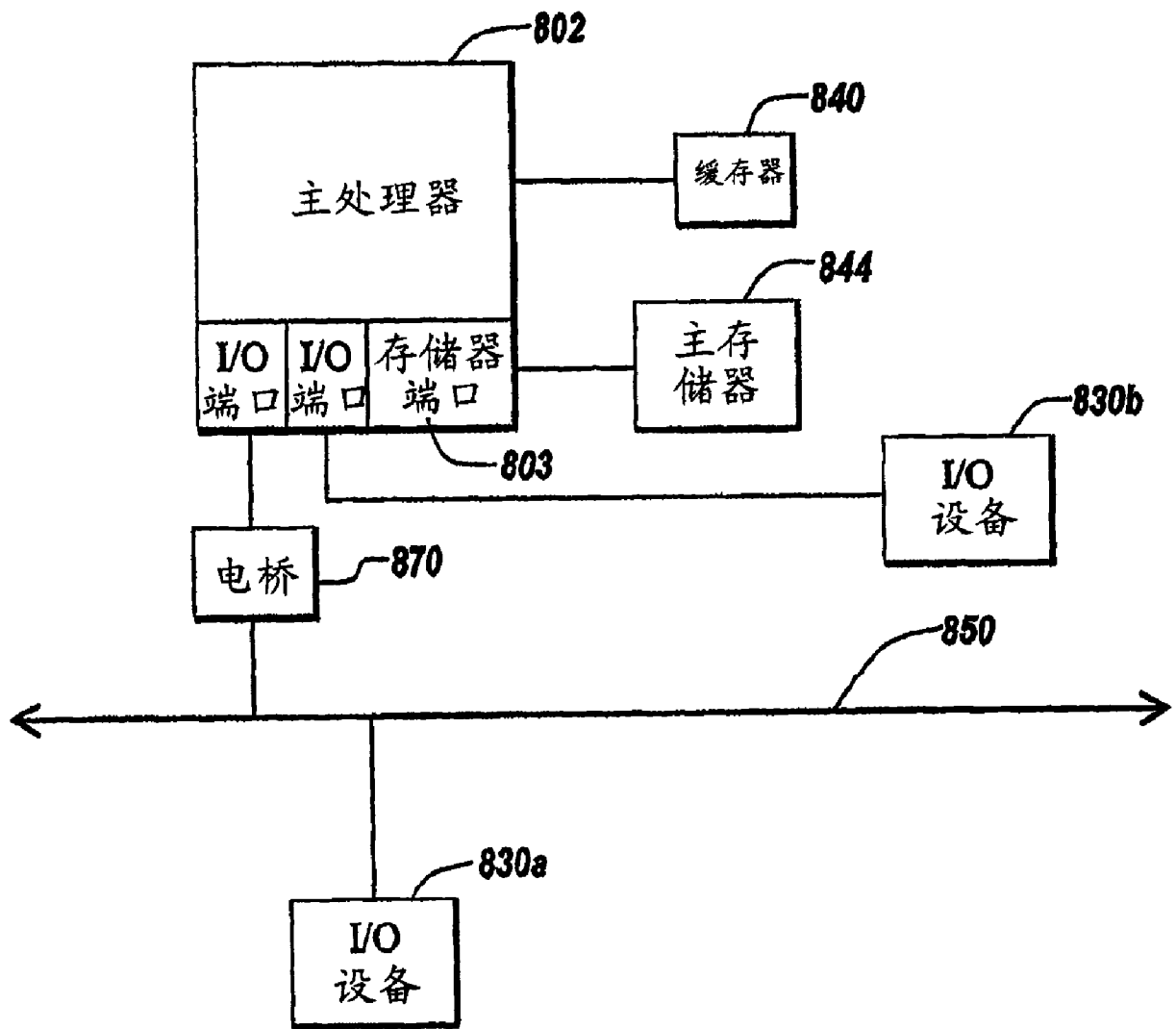


图 8B