[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)
G06F 9/445 (2006.01)

[12] 发明专利说明书

专利号 ZL 03807856.2

[45] 授权公告日 2008 年 8 月 13 日

[11] 授权公告号 CN 100410929C

[22] 申请日 2003.3.26 [21] 申请号 03807856.2

[30] 优先权

[32] 2002. 4. 8 [33] US [31] 60/370,963

[32] 2002. 4.12 [33] US [31] 60/372,186

[32] 2002. 9.23 [33] US [31] 10/253,088

[86] 国际申请 PCT/US2003/009407 2003.3.26

[87] 国际公布 WO2003/088093 英 2003.10.23

[85] 进入国家阶段日期 2004.10.8

[73] 专利权人 甲骨文国际公司

地址 美国加利福尼亚州

[72] 发明人 苏罗吉特·查特吉 拉杰·库马尔
乔纳森·克赖顿
阿洛科·斯里瓦斯塔瓦
萨默尔·乔希

[56] 参考文献

US6016499A 2000.1.18

US6345382B1 2002.2.5

US6067584A 2000.5.23

CN1302027A 2001.7.4

US6327594B1 2001.12.4

审查员 赵 颖

[74] 专利代理机构 北京康信知识产权代理有限责任公司

代理人 余 刚

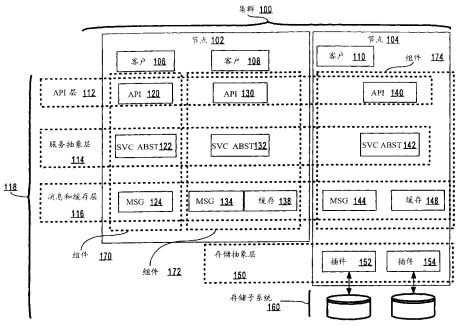
权利要求书 6 页 说明书 19 页 附图 2 页

[54] 发明名称

具有可插入体系结构的键值仓库

[57] 摘要

本发明提供了管理键值对的系统和技术，这种系统和技术使用的体系结构不把其用户限制到任何特定的平台或存储子系统。 根据本发明的一个方面，仓库是可移植的，其体系结构无须根据环境和该仓库使用的平台而做出改变。 相反的，该体系结构中与特定平台相关的部分仅限于在该仓库的存储抽象层中的插件。 这些插件提供同样的存储抽象接口给该仓库的其它层，但在实现时与不同的平台和存储子系统交互。 因此，当从一个平台迁移到另一个平台时，该仓库只简单的改变插件就可以调用持久性存储操作。



1. 一种用于储存键值对的方法，所述方法包括:

在应用编程层提供仓库接口，客户通过所述仓库接口可以进行调用，以储存并访问仓库中的键值对信息;

在第一节点上的第一组件处通过所述仓库接口接收来自所述第一节点上的客户的请求，以执行操作，其中，所述第一组件与所述第一客户关联;

响应于在所述第一组件处接收所述来自所述第一节点上的客户的请求，确定所述第一组件是否已经被指定为用于所述第一节点的缓存组件;

响应于确定所述第一组件没有被指定为用于所述第一节点的缓存组件，确定所述第一节点上的第二组件是否被已经指定为用于所述第一节点的缓存组件，其中，所述第二组件与所述第一节点上的第二客户关联;

响应于确定所述第二组件已经被指定为用于所述第一节点的缓存组件，在所述第二组件处确定所述来自所述第一节点上的客户的请求所要求的信息是否被包含在所述第一节点上的本地缓存中;

响应于确定所述来自所述第一节点上的客户的请求所要求的所述信息不包含在所述第一节点上的所述本地缓存中，确定第二节点上的第三组件是否已经被指定为主控组件，其中，所述第二节点与所述第一节点分开，其中，所述第三组件与所述第二节点上的第三客户关联;

响应于确定所述第三组件已经被指定为主控组件，在所述第三组件上执行以下步骤:

通过存储抽象层的公用接口调用多个插件中的一个，从而通过所述仓库接口进行服务调用；

确定所述多个插件中的哪个特定插件对应于执行所述操作期间要访问的存储子系统的类型；

动态装载对应于所述存储子系统的类型的特定插件；以及

通过所述公用接口进行一个或多个对所述特定插件的调用，执行所述操作；

其中，所述多个插件中的每个插件都提供所述公用接口，以访问持久储存的键值对信息；

其中，所述多个插件中的每个插件都被设计成：响应通过所述公用接口进行的调用，与特定类型的存储子系统交互；以及

其中，每个插件与之交互的存储子系统是不同于所述多个插件中的每个其它插件与之交互的存储子系统的不同类型的存储子系统。

2. 根据权利要求 1 所述的方法，进一步包括如下步骤：

在分布在集群的多个节点的组件处，通过所述仓库接口接收来自驻留在所述多个节点上的客户的调用；

将所有需要访问持久性存储器的调用引导向所述主控组件；

其中所述主控组件是所述仓库中唯一允许通过所述公用接口执行插件调用步骤的组件。

3. 根据权利要求 1 所述的方法，进一步包括如下步骤：

通过所述仓库接口接收来自客户的请求；

在通过所述公用接口进行调用获得所述请求所要求的所述信息之前，在主控缓存中查找所述信息。

4. 根据权利要求 3 所述的方法，其中：

所述第一客户驻留在所述第一节点，而所述主控缓存驻留在所述第二节点上；以及

所述方法进一步包括如下步骤：在所述主控缓存中查找所述信息之前，在所述第一节点的所述本地缓存中查找所述信息。

5. 根据权利要求 4 所述的方法，其中：

所述第一节点包括所述仓库的多个组件；

所述方法进一步包括如下步骤：仅将所述多个组件中的一个组件建立为所述缓存组件；以及

如果所述第一组件不是所述缓存组件，那么就从所述第一组件发送消息给所述缓存组件，以使所述缓存组件在所述本地缓存查找所说的信息。

6. 根据权利要求 1 所述的方法，进一步包括如下步骤：

创建代码库，所述代码库包含实现所述仓库接口的例程；

将所述代码库链接到客户代码，以允许在所述客户代码中的例程可以通过调用所述代码库中的所述例程使用所述仓库。

7. 根据权利要求 1 所述的方法，进一步包括如下步骤:

通过例程接收来自客户的调用，所述例程用于实现所述仓库接口的一部分；以及

根据在所述调用中的参数的值，确定是否把与所述客户相关联的所述仓库的组件建立成缓存组件，所述缓存组件负责管理所述仓库的缓存。

8. 一种用于储存键值对的方法，所述方法包括:

提供仓库接口，客户通过所述仓库接口进行调用，以储存和访问仓库中的键值对信息;

通过例程接收来自客户的调用，所述例程用于实现所述仓库接口的一部分；以及

根据所述调用中的参数值，确定是否把与所述客户相关联的所述仓库的组件建立成用于管理所述仓库的缓存的缓存组件。

9. 根据权利要求 8 所述的方法，其中:

所述仓库支持多个访问模式，所述访问模式包括缓存访问模式和缺省访问模式;

所述接收调用的步骤包括接收指定了缓存访问模式和缺省访问模式中之一的调用；以及

所述方法包括以下步骤：如果所述调用指定了所述缓存访问模式，就把与所述客户相关联的所述仓库的组件建立成缓存组件。

10. 根据权利要求 9 所述的方法，进一步包括如下步骤：

作为对确定所述调用指定所述缺省访问模式的响应，传递请求以从所述仓库读取信息，所述信息被所述组件从所述客户接收到与第二客户相关联的第二缓存组件；以及

使所述第二缓存组件响应所述请求，检查由所述第二组件所管理的缓存中是否有来自所述仓库的信息。

11. 根据权利要求 10 所述的方法，其中，所述组件和所述第二缓存组件驻留在单一节点上。

12. 根据权利要求 11 所述的方法，进一步包括如下步骤：

如果第二缓存组件管理的缓存不包含来自所述客户的特定请求所需要的信息，那么所述第二缓存组件发送所述请求到驻留在第二节点的第三组件，所述第二节点相对所述节点是远程节点。

13. 根据权利要求 9 所述的方法，其中：

所述多个访问模式进一步包括安装访问模式和只读访问模式；

当最初创建所述仓库时，所述客户使用所述安装访问模式；以及

当从所述仓库启动时，所述客户使用所述只读访问模式。

14. 一种用于储存键值对的方法，所述方法包括：

提供仓库接口，客户通过所述仓库接口进行调用，以储存和访问专门为键值对信息设计的仓库中的键值对信息；

通过所述仓库接口接收来自客户的一个或多个调用；其中，在持久存储器上分配用于所述仓库的任何结构以持久地储

存所述键值对信息之前，在所述仓库接口处接收所述一个或多个调用；以及

根据所述一个或多个调用，在所述持久存储器上创建用于持久地储存所述键值对信息的结构。

15. 根据权利要求 14 所述的方法，其中：

所述仓库支持多个访问模式，包括安装访问模式和缺省访问模式；

接收一个或多个调用的步骤包括接收至少一个指定安装访问模式的调用；以及

所述方法包括以下步骤：将所述安装访问模式授权给所述客户，当所述客户处于所述安装访问模式时，能够排它地使用所述仓库。

16. 根据权利要求 15 所述的方法，其中：

所述多个访问模式包括只读访问模式；且所述方法包括如下步骤：

接收来自第二客户的请求，从而使用只读访问模式执行操作；

在执行所述操作过程中，回避所述仓库的并发控制机制。

具有可插入体系结构的键值仓库

相关申请

本申请主张以下美国临时专利申请的优先权，其全部内容结合于此作为参考：

美国临时专利申请号 60/370,963，标题为“在全局命名空间中的个性化内容（Personalized Content Within a Global Namespace）”，2002 年 4 月 8 日提交；以及

美国临时专利申请号 60/372,186，标题为“在全局命名空间中实现个性化内容的方法（Approach for Personalized Content Within a Global Namespace）”，2002 年 4 月 12 日提交。

技术领域

本发明涉及数据仓库，更具体而言，涉及存储键值对的数据仓库。

背景技术

许多类型的信息通常以键值对的形式存储，其中该对中的“键”部分是一个标签，而该对中的“值”部分提供与该标签相关的值。比如，有关计算机系统的配置信息包括下面的键值对：（“内存”，512M）表明该计算机系统的动态内存的容量是 512 兆字节。

典型的，需要储存大量的键值对的软件程序或系统会包括一个仓库来储存该信息，并包括管理该仓库的逻辑。当这些仓库被用于存储配置数据时，这些仓库通常被称作注册表。

当每个需要键值对仓库的程序或系统实现并管理其自身的仓库时，其结果是专用仓库的不断增长以及大量重复性的工作。为了解决该问题，键值对仓库设计者为其仓库提供了应用编程接口（API）以允许特定的第三方应用使用他们的仓库。比如，操作系统可能允许为该操作系统设计的第三方应用来储存键值对到由该操作系统管理的仓库。

不幸的是，这种“开放”的仓库不提供通用的解决方案，原因在于它们通常与特定的平台或存储子系统相关，因此它们不能或不便于用作通用目标的键值对仓库。由于它们被设计成使用特定的有关环境和运行平台的假设条件，他们通常不能被不符合这些假设条件的应用或系统所使用。

附图说明

本发明参考附图以举例方式进行描述而不是以限制性方式加以描述，其中同样的参考数字指的是同样的部件，其中：

图 1 的方框图是根据本发明的实施例的键值对仓库；以及

图 2 的方框图是描述本发明实例所实现的计算机系统。

具体实施方式

本文描述了提供集群级访问的共享的，键值对仓库的方法和系统。在下面的描述中，出于解释的目的，大量的具体细节被提出以便提供对本发明透彻的理解。但是，很显然，本发明可以在没有这

些具体细节的情况下实现。在其它情形下，已知的结构和设备被显示在方框图中以避免对本发明不必要的阻碍。

功能概述

提供了使用不限制其用户到任何特定平台或存储子系统的体系结构管理键值对的技术。根据其一个方面，仓库是可移植的，其体系结构无须根据环境和该仓库使用的平台而做出改变。相反的，该体系结构中与特定平台相关的部分仅限于在该仓库的存储抽象层（storage abstraction layer）中的插件。每个插件提供同样的存储抽象接口给该仓库的其它层，但在实现时与其它插件不同的平台和存储子系统交互。

另外，该体系结构的可插入特性导致交叉平台的可移植性。比如，该仓库可以用于任何数量的通用操作系统/硬件组合，比如Solaris/SUN, AIX/IBM, HPUX/HP，WINDOWS/DELL，等。因此，当从一个平台迁移到另一个平台时，该仓库只简单的改变插件就可以调用持久性存储操作。

根据一个方面，该仓库的性能通过维护有关键值对信息的主缓冲区以及有关键值对信息的本地缓冲区而得到增强。根据一个实施例，在每个节点管理单一本地缓冲区，而与任何给定节点的缓冲区相关的客户端不专属于该仓库。比如，某节点的缓冲区与在大量使用该仓库的节点的客户端相关，通过让该客户端发出适当的对仓库的调用（call）。

系统概述

参照图 1，该方框图描述了根据本发明的实例，提供集群范围的访问共享的键值对仓库 **118** 的系统。具体而言，图 1 描述了包括两个节点 **102** 和 **104** 的集群 **100**。所显示的两节点集群实例简化了

对系统的描述。但是，此处描述的本发明和技术不局限于任何特定数量的节点的集群。

此处使用的术语“集群”指任何能够彼此通信的一组节点。集群可以包括，比如说，一组联网计算机。在某些情形下，集群由集群管理软件作为一个单元进行管理。集群管理软件是一个能够利用键值对仓库 **118** 的软件系统的例子。具体而言，集群管理软件可以使用键值对仓库 **118** 作为集群注册表，来储存由集群管理软件管理的集群的配置信息。但是，应该注意集群管理软件只是能够使用仓库 **118** 的客户（client）的一个例子。本发明不局限于任何特定类型的客户。相反，在许多情形下，各种分离类型的客户能使用仓库 **118**。

再次参照图 1，节点 **102** 包括仓库 **118** 的两个客户 **106**、**108**，而节点 **104** 包括仓库 **118** 的一个客户 **110**。客户 **106**、**108** 和 **110** 被称为“客户”是因为它们请求仓库 **118** 的服务以便管理它们感兴趣的键值对。客户 **106**、**108** 和 **110** 一般代表任何形式的需要存储键值对的软件程序，而并不局限于任何特定类型的软件程序。

在图 1 描述的实施例中，仓库 **118** 包括几个层次的功能。具体的，仓库 **118** 包括 API 层 **112**、服务抽象层（service abstraction layer）**114**、消息和缓存层 **116**、存储抽象层 **150**、以及存储子系统 160。一般的，API 层 **112** 代表面向所有客户 **106**、**108** 和 **110** 的通用接口，通过该接口客户可以进行调用，以储存、访问和管理在仓库 118 中的键值对。正如后面将详细描述的，在 API 层 **112** 中的例程为客户 **106**、**108** 和 **110** 提供了独立于实际存储子系统 **160** 的接口，该实际存储子系统用来持久地储存键值对。

服务抽象层 **114** 包括的例程用来确定如何处理 **API** 层 **112** 的例程所接收到的调用。消息和缓存层 **116** 包括的例程可以被仓库 **118** 的组件调用，以便与仓库 **118** 的其它组件通信。另外，消息和缓存

层 116 包含的例程可以管理键值对的缓存，从而使得并非所有来自客户的请求都需要访问存储子系统 160。

存储抽象层 150 包括插件 152 和 154。每个插件提供同样的存储访问 API 给层 114 和 116 中的例程。但是，实现通用存储 API 的例程根据与插件相关的存储子系统的类型的不同而导致其插件也不同。存储子系统 160 代表任何形式的能够存储键值对的可靠的存储系统。后面将详细的描述每个层。

主控和缓存组件

根据一个实施例，仓库 118 的一个组件被指派为主控组件。主控组件的消息和缓存层 116 的例程能独占（排它）访问存储抽象层 150。主控组件还管理用来存放来自仓库 118 的信息的主控缓存。在图 1 描述的实施例中，组件 174 是主控组件。因此，组件 174 中的消息和缓存层 116 的例程管理主控缓存 148，并能独占访问驻留在存储抽象层 150 中的插件 152 和 154。

集群中的每个节点还包括单一“缓存组件”。节点的缓存组件维护来自仓库 118 的信息。与缓存组件相关联的客户此处被称为“缓存层客户”。根据一个实施例，主控组件是其驻留的节点的缓存组件。

在所描述的实施例中，组件 172 是节点 102 的缓存组件，因此它管理着缓存 138。客户 108 与组件 172 相关联，是节点 102 的缓存层客户。组件 170 同样驻留在节点 102，不是缓存组件，因此没有维护其自己的缓存来存放来自仓库 118 的信息。

根据一个实施例，客户建立其所关联的组件为缓存组件的方式是基于通过该客户调用 API 层 112 的例程传递信息给仓库 118。比如，客户 108 使其自己成为节点 102 的缓存层客户的方式是通过适

当的调用 API **130**。该调用传递给 API **130** 一个值来指示组件 **172** 将成为节点 **102** 的缓存组件。

因为缓存层客户通常能更快的访问由缓存组件管理的缓存，所以在特定的节点的缓存层客户最好是使用仓库 **118** 最频繁的客户。比如，客户 **106** 和客户 **108** 分别代表两种不同类型的客户，其中客户 **108** 更频繁的使用仓库 **118** 而客户 **106** 相对少的使用仓库 **118**。在这些情况下，客户 **108** 被选择为缓存层客户。那么客户 **108** 就可以被指派对组件 **172** 进行适当的调用以便导致组件 **172** 成为节点 **102** 的缓存组件。

根据另一个实施例，节点 **102** 中客户使用仓库的实际情况被监控，使用仓库 **118** 最频繁的组件被动态的选择为缓存组件。在该实施例中，甚至提供例程根据与该组件关联的客户的变化的访问模式来动态的把缓存责任从一个组件传递给另一个组件。

当对仓库执行写操作时，各种技术可以被使用来管理缓存。比如，一个实施例使用“写通”方法在写操作期间维护缓存。本发明不局限于任何特定的方法在写操作期间管理缓存。

在图 1 所描述的实施例中，集群 **100** 有单一主控组件 **174** 有权限与存储抽象层 **150** 进行交互。因为抽象层 **150** 仅能从单一主控组件 **174** 中访问，与资源共享和并发控制相关的各种问题都可以避免。但是，另一个实施例可以包括一组主控组件有权限直接与存储抽象层 **150** 交互。各种并发控制技术可以被使用来避免该系统中的潜在的有害的交互。比如，键的命名空间可以被分区，每个主控组件被指派一个命名空间分区。在该实施例中，每个主控组件仅被允许访问存储抽象层 **150** 来操作分配给主控组件的命名空间分区的键。

API 层

希望使用仓库 **118** 来管理键值对的客户通过调用在 API 层 **112** 的例程实现这一目的。根据一个实施例，在 API 层 **112** 中的例程提供仓库 **118** 支持的所有的操作的接口。在一个实施例中，API 层 **112** 的例程执行参数验证和错误检测。如果对 API 层 **112** 的例程的调用传递测试在 API 层 **112** 被执行，接着调用被向下传递给在服务抽象层 **114** 的适当例程。

API 层 **112** 的例程可能以代码库的形式提供给客户的开发者。开发者，在其客户端上，包括代码来调用在代码库中的例程。接着，在客户代码被编译时代码库被静态链接到客户代码，或者在运行时被动态的链接到客户代码。

根据一个实施例，API 层 **112** 提供的接口包括为多种编程语言的接口。比如，API 层 **112** 可以提供 JAVA API 供客户用户使用或用 JAVA 编程语言编译，以及提供 C 语言 API 供客户用户使用或用 C 编程语言编译。

根据另一个实施例，API 层 **112** 只提供 C 语言接口，而仓库 **118** 根据所接收的用其它语言编写的客户调用，包括一个或多个附加的模块来调用 API 层 **112**。这些模块有效地把来自客户的某种语言的调用转换成 API 层 **112** 所提供的接口所支持的语言的调用。比如，该模块可以提供基于 JAVA 的仓库 API 到基于 JAVA 的客户，并接着使用 JNI 包装器把调用转换成基于 JAVA 的仓库 API 调用通过该模块转换成由 API 层 **112** 提供的基于 C 的仓库接口。

服务抽象层

服务抽象层 **114** 确定如何处理来自客户的调用。有多种因素可以确定如何处理任何给定的调用。这些因素包括，举例而言，发起调用（making call，进行调用）的客户的标志符，施加到该调用上的访问模式，以及由该调用所请求的操作的类型。访问模式将在下面被进一步详细描述。

根据这些因素，在服务抽象层 **114** 中的例程可以访问本地缓存，访问主控缓存，或调用在消息和缓存层 **116** 中的消息例程来发送请求到仓库 **118** 的不同的组件。如果在服务抽象层 **114** 中所调用的例程不是在缓存组件或主控组件中，那么在服务抽象层 **114** 中所调用的例程就调用在消息和缓存层 **116** 中的例程来传递请求到本地缓存组件。

如果服务抽象层 **114** 中所调用的例程是在缓存组件中，那么服务抽象层 **114** 中所调用的例程就检查本地缓存看是否该缓存包含了回答该请求的信息。如果本地缓存没有包含回答该请求的信息，那么服务抽象层 **114** 中所调用的例程就调用在消息和缓存层 **116** 中的例程来传递请求到主控组件。

如果服务抽象层 **114** 中所调用的例程是在主控组件中，那么服务抽象层 **114** 中所调用的例程就检查主控缓存看是否该缓存包含了回答该请求的信息。如果主控缓存没有包含回答该请求的信息，那么服务抽象层 **114** 中所调用的例程就调用在消息和缓存层 **116** 中的例程来激活在存储抽象层 **150** 中的适当的插件以便从某个存储子系统 **160** 中的持久存储器中提取信息。

根据一个实施例，服务抽象层 **114** 中的例程所调用的位于消息和缓存层 **116** 的消息例程的所有调用都是无状态的。在该实施例中，

服务抽象层 **114** 通过在消息和缓存层 **116** 的消息例程发送的每个消息都包含执行预期操作所必须的所有信息。

根据一个实施例，缓存组件和主控组件的服务抽象层 **114** 的例程具有验证客户的责任。在验证操作期间，该例程确定该客户是否是值得“信任”的。没有被信任的客户不会被允许执行特定类型的操作。

消息和缓存层

消息和缓存层 **116** 包括的例程用来访问和管理缓存，以及用来与仓库 **118** 中的其它组件通信。根据一个实施例，在一个节点中只能有一个组件使用缓存例程。节点上的其它组件间接的使用缓存，其方式是转发请求给管理缓存的组件。

根据一个实施例，缓存例程如何满足请求对调用缓存例程的例程而言是完全透明的。具体而言，一旦接收到请求，缓存例程就检查它管理的缓存。如果缓存中包含满足该请求所需的信息，那么缓存例程就从缓存中提取信息并提供该信息给调用例程。但是，如果缓存中不包含满足该请求所需的信息，那么缓存例程就通过其它方式获得所需的信息。比如，如果缓存例程属于缓存组件，那么缓存例程就调用消息例程从主控组件请求所需的信息。如果缓存例程属于主控组件，那么缓存例程就调用适当的存储抽象层插件从存储子系统提取所需的信息。该缓存例程除了提供所请求的信息返回调用例程之外，还可能导致该信息被存储在它管理的缓存中。

根据一个实施例，在消息和缓存层 **116** 中的消息例程被配置成按照网络字节顺序发送，至少当该消息必须穿越在集群中的平台边界时。比如，假设节点 **102** 运行在第一个平台，而节点 **104** 运行在第二个平台。在该情形下，在节点 **102** 中的消息例程以网络字节顺

序发送消息给节点 **104** 中的消息例程。类似的，在节点 **104** 中的消息例程以网络字节顺序发送消息给节点 **102** 中的消息例程。通过用网络字节顺序发送消息来穿越平台边界，发送者发送的消息可以被驻留在不同于发送者的平台的接收者精确的重新组建。

存储抽象层

根据一个实施例，存储抽象层 **150** 提供的例程完全抽象于后端存储器的类型，该后端存储器被用于持久地储存由仓库 **118** 管理的键值对信息。比如，插件 **152** 和插件 **154** 两者都提供相同的接口给主控组件 **174** 的消息和缓存层 **116** 中的缓存例程，即使插件 **152** 被设计成操作完全与插件 **154** 的存储子系统不同的存储子系统。尽管插件 **152** 和插件 **154** 提供了相同的接口给缓存例程，但是实现该接口的例程的逻辑根据该插件所设计的与之交互的后端存储器的类型的不同而可能完全不同。

重要的是，由于所有平台相关的逻辑都包含在存储抽象层 **150** 的插件中，所以仓库 **118** 的所有其它层的例程都不是平台相关或存储子系统相关的。这样，使用仓库 **118** 的客户可以有效的与平台相关的设计细节所隔离，从而使包括客户和仓库 **118** 的高层组件在内方便的跨越平台。

由于存储抽象层 **150** 使用一个或多个提供相同接口的插件实现，仓库 **118** 不局限于任何特定的后端子系统或平台。任何子系统或平台都能够提供与所使用的公用接口相关的功能。该后端平台包括，但不限于，LDAP，MSCS，共享原设备，在无共享集群中的原设备和/或私有文件系统文件，集群文件系统（Cluster File Syetem，CFS）以及分布式配置库。

当仓库 118 被请求执行操作来请求访问在持久性存储中的键值对时，就会发起一个调用（通常来自主控组件的消息和缓存层的缓存例程）给存储抽象层 150。根据一个实施例，仓库 118 根据一个或多个不同的可能因素选择特定的插件来调用。比如，仓库 118 可以包含其值由管理员设置的环境变量，该环境变量的值指示哪种类型的存储子系统被用于持久存储仓库 118 所管理的数据。另外，仓库 118 可以包括发现逻辑来检测哪个存储子系统对仓库 118 的特定安装而言是可用的。如果仅有一种类型的子系统可用，那么就选择与该类型的子系统相关的插件。如果有几种类型都可用，那么仓库 118 就根据其它的各种原因来选择仓库 118。这些原因包括但不限于存储容量或存储子系统 160 的可用剩余空间。一旦做出了选择，适当的插件就被动态的装载到易失性内存中，并调用在该插件中的例程。

由于每个插件被设计成与不同类型的后端存储系统交互，插件的选择就确定了后端平台的特性，在该后端平台上持久地储存由仓库 118 管理的键值对信息。比如，如果目标键值对位于共享的存储设备，在插件 152 中的例程被调用。如果目标键值对被由 LDAP 服务器管理的存储子系统所管理，那么在插件 154 中的相应的例程就被调用。根据一个实施例，被调用的例程的名字，以及调用的参数，在两种情况下是相同的，因为插件 152 和 154 提供的接口是相同的，从而使得存储子系统 160 的特性对客户 106，108 和 110 以及仓库 118 的高层例程而言是透明的。

存储抽象层 150 提供的接口允许在存储抽象层 150 之外的例程通过存储抽象层 150 的例程调用来执行仓库 118 所需的各种操作。在一个实施例中，仓库 118 支持节点相关键。具体而言，仓库 118 的多个客户（节点 102 的客户 106 和 108，节点 104 的客户 110）每个都希望储存键值对，而每个客户所使用的键名都是相同的，但是在不同节点上的每个客户针对该键名所使用的值可能是不同的。

比如，每个客户都希望储存键值对“背景色”=X，而X可能是蓝，绿或紫等一种颜色。使用节点相关键，仓库**118**将为每个客户分别存储“背景色”的值，根据该客户碰巧运行的节点。对目前这个例子而言，在节点**102**上的客户**106**和**108**将看到有关背景色键的同样的值，而在节点**104**上的客户**110**则将看到对同一个背景色键的不同的值。

例样请求序列

根据一个实施例，请求仓库**118**所管理的信息的过程首先检查与请求者驻留在同一个节点的缓存。如果该信息没有在本地节点缓存，那么就检查主控缓存。如果该信息没有在主控缓存，那么就调用存储抽象层**150**从可靠的存储中提取信息。

比如，假设客户**106**请求仓库**118**管理的特定的键值对，该信息目前没有驻留在任何缓存中。根据一个实施例，为客户**106**提取键值对的操作如下处理:

最初，客户**106**调用在API层**112**中的例程（API **120**）请求读取期望的键值对。API **120**发送请求到服务抽象层**114**的例程（SVC ABST **122**）。SVC ABST **122**确定如何处理请求。在该例子中，SVC ABST **122**确定请求应该转发到节点**102**本身的缓存。由于组件**170**不是节点**102**的缓存组件，转发请求给缓存的操作导致组件**170**与另一个组件**172**通信。为了完成该通信，该请求被传递到组件**170**的消息和缓存层**116**的例程（MSG **124**）。

MSG **124**与缓存组件**172**的消息和缓存层**116**的例程MSG **134**通信。MSG **134**向上传递请求给缓存组件**172**的服务抽象层**114**的例程（SVC ABST **132**）。SVC ABST **132**确定应该检查缓存**138**中

是否包含所请求的信息。SVC ABST **132** 调用在消息和缓存层 **116** 的适当的例程。

当在本地缓存 **138** 中没有发现信息时，就从 MSG **134** 发送消息给主控组件 **174** 的消息和缓存层 **116** 的例程(MSG **144**)。MSG **144** 向上传递请求给主控组件 **174** 的服务抽象层 **114** 的例程(SVC ABST **142**)。SVC ABST **142** 确定应该检查缓存 **148** 中是否包含所请求的信息。SVC ABST **142** 调用在消息和缓存层 **116** 的适当的例程。

当在主控缓存 **148** 中没有发现信息时，就发送消息给存储抽象层 **150** 的插件（比如插件 **152**)。该插件 **152** 导致所请求的信息从一个存储子系统 **160** 的持久存储被提取到缓存 **148** 中。MSG **144** 接着就通过传递回所请求的信息返回来自 MSG **134** 的调用。MSG **134** 导致该信息被存储在缓存 **138**，并通过传递回所请求的信息返回来自 MSG **124** 的调用。

MSG **124** 传递该消息给 SVC ABST 122，并接着返回该消息给 API **120**。最后，API 120 发送请求的信息给客户 **106**。

由于所请求的信息被存储在缓存 138，节点 **102** 的任何客户针对该同一信息的后续请求都被处理成从缓存 **138** 提取数据。由于所请求的信息被存储在主控缓存 **148**，在本地缓存中没有该信息的节点的客户针对该同一信息的后续请求都可能被处理成从主控缓存 **148** 提取数据。

访问模式

根据一个实施例，在 API 层 **112** 中提供的例程可以被四种访问模式中的任何一种所访问。客户可以指定特定的访问模式到 API 层 **112** 的例程，比如，通过传递参数值来指示访问模式。不是在每个调用中都传递这样的参数给 API 层 **112**，而是在客户发起的对 API

层 **112** 的第一个调用中指定期望的模式，这称为“初始化例程”。该初始化例程可以传回一个句柄给客户，接着客户就使用该句柄实现对 API 层 **112** 的所有未来的调用。该句柄可以与指示客户访问模式的数据相关联。因此，在后续所有调用中传递句柄将有效的通告处理这些后续调用的例程有关用来服务这些调用的访问模式的信息。

根据一个实施例，同一客户的不同模块可能以不同的访问模式访问仓库 **118**。比如，在客户 **108** 中的第一个模块可能以只读模式访问仓库 **118**，而在客户 **108** 中的另一个模块可能以缓存访问模式访问仓库 **118**。类似的，客户可能通过适当的对 API 层 **112** 的调用从一种访问模式转化到另一种访问模式。

特定客户请求的访问模式取决于该客户的角色和/或该客户想要执行的操作类型。根据一个实施例，仓库 **118** 所支持的模式包括安装模式，只读访问模式，缓存访问模式，以及缺省访问模式。

安装模式：安装模式是客户（通常是仓库安装程序）用来执行创建或重建仓库工作的模式。根据一个实施例，一次只能有一个客户使用安装模式，并禁止所有其它客户执行任何仓库操作。

只读访问模式：通过请求只读访问模式，客户表明它将只进行只读操作。当客户处于只读访问模式执行操作时，在仓库 **118** 中的所有并发控制逻辑被禁用/回避。因此，相比允许写操作的模式而言，只读访问模式下的操作通常被更快的以更少的消耗执行。只读访问模式可以并发的被多个客户共享，由于读操作通常不会与其它读操作发生访问冲突。

根据一个实施例，仓库 **118** 使用集群配置软件来在仓库 **118** 的操作期间执行某些并发控制。该集群配置软件还使用仓库 **118** 来存

储集群 **100** 的启动信息。在该实施例中，当集群 **100** 被启动时，该集群配置软件使用只读访问模式访问集群配置信息，以确保集群配置软件的并发控制例程直到集群配置软件自身完全初始化之后才被执行。

缓存访问模式：缓存访问模式是客户用来建立组件作为客户的节点的缓存组件的模式。比如，组件 **172** 被建立为节点 **102** 的缓存组件以响应客户 **108** 以缓存访问模式初始化组件 **172**。当客户 **108** 发起调用来初始化组件 **172** 时，就为缓存 **138** 分配资源。当在缓存访问模式中执行读操作时，在服务抽象层 **114** 中的例程就激活消息和缓存层 **116** 中的缓存例程以便在缓存 **138** 中搜索需要的信息。

缺省访问模式：缺省访问模式是如下客户所使用的模式，（1）没有与缓存组件相关联的客户，（2）没有执行需要安装或只读访问模式的操作的客户。当按照缺省访问模式执行操作时，在服务抽象层 **114** 中的例程使用消息和缓存层 **116** 的消息例程转发读取请求给本地缓存组件。

根据一个实施例，客户能够从一种模式转化到另一种模式。该转化可以通过一定方式而发起，举例而言，在对 API 层例程的后续调用中传递不同于对 API 层例程的前期调用的访问模式参数值。根据另一个实施例，单一客户进程的不同线程或模块使用不同的访问模式。比如，客户的第一个模块的所有调用都传递一个访问模式值给 API 层例程来获得一个访问模式，而同一客户的第二个模块的所有调用都传递一个不同的访问模式值给 API 层例程来获得不同的访问模式。

硬件综述

图 2 是描述可以执行本发明一个实施例的计算机系统 **200** 的框图。计算机系统 **200** 包括总线 **202** 或用于传递信息的其他通信结构，并且包括与总线 **202** 耦合并用于处理信息的处理器 **204**。计算机系统 **200** 还包含主存储器 **206**，例如随机访问存储器（RAM）或是其它动态存储器，这个存储器与总线 **202** 耦合，用于保存信息以及处理器 **204** 所要执行的指令。在运行处理器 **204** 所执行指令的过程中，主存储器 **206** 还可用于保存临时变量或是其它中间信息。计算机系统 **200** 还包括一个只读存储器（ROM）**208** 或其它静态存储设备，它与总线 **202** 耦合，用于保存静态信息和涉及处理器 **204** 的指令。此外还提供了诸如磁盘或光盘的存储设备 **210**，所述设备与总线 **202** 耦合，以便保存信息和指令。

计算机系统 **200** 可以经由总线 **202** 而与阴极射线管（CRT）之类的显示器 **212** 相连，以便将信息显示给计算机用户。包含字母数字及其它键的输入设备 **214** 与总线 **202** 相连，以便将信息和命令选择传递到处理器 **204**。另一种用户输入设备是光标控制 **216**，例如鼠标、轨迹球或光标方向键，用于将方向信息和命令选择传递给处理器 **204** 以控制显示器 **212** 上的游标移动。这种输入设备通常在第一轴（例如 x）和第二轴（例如 y）这两个轴上具有两个自由度，由此设备能够确定一个平面上的位置。

本发明涉及使用计算机系统 **200** 来执行这里所描述的技术方法。根据本发明的一个实施例，处理器 **204** 执行主存储器 **206** 中包含的一个或多个指令的一个或多个序列，计算机系统 **200** 对此做出响应，由此执行这些技术方案。这些指令可以从诸如存储设备 **210** 等等的另一种计算机可读介质读入主存储器 **206**。通过执行主存储器中包含的指令序列，处理器 **204** 执行这里描述的处理步骤。在替

换实施例中，硬布线电路可用于取代软件指令或是与之组合，由此实现本发明。因此，本发明的实施例不限于硬件电路和软件的任何一种特定组合。

这里使用的术语“计算机可读介质”是指任何一种参与向处理器 **204** 提供指令以供执行的介质。这种介质可以采取很多形式，其中包括但不局限于：非易失介质、易失介质和传输介质。举例来说，非易失介质包括光盘或磁盘，例如存储设备 **210**。易失介质包括动态存储器，例如主存储器 **206**。传输介质包括同轴电缆、铜线和光纤，其中包括了构成总线 **202** 的线路。传输介质还可以采取声波或光波的形式，例如无线电波和红外数据通信中产生的信号。

举例来说，计算机可读介质的通用形式包括：软盘、软磁盘、硬盘、磁带或任何其它磁介质、光盘或任何其它光学介质、穿孔卡、纸带纸条或具有孔洞图案的任何其它物理介质、RAM、PROM 和 EPROM、FLASH-EPROM、其它任何存储芯片或盒式磁盘机、如下所述的载波或是计算机可以读取的其它任何介质。

各种形式的计算机可读介质可以承载一个或多个序列的一个或多个指令以便处理器 **204** 执行。举例来说，最初将指令传送到远程计算机磁盘上。远程计算机可以将指令加载到它的动态存储器中，并且使用调制解调器而经由电话线来发送指令。计算机系统 **200** 的本地调制解调器可以在电话线上接收数据并且使用红外发射机来将数据转换成红外信号。红外检测器可以接收红外信号中传送的数据，而适当的电路则可将数据放置到总线 **202** 上。总线 **202** 将数据传送到主存储器 **206**，处理器 **204** 从主存储器 **206** 中提取并执行指令。主存储器 **206** 接收的指令还可选地在处理器 **204** 执行之前或之后被储存在存储设备 **210** 中。

计算机系统 **200** 还包括与总线 **202** 相连的通信接口 **218**。通信接口 **218** 提供了与网络链路 **220** 耦合的双向数据通信，其中网络链路 **220** 与本地网络 **222** 相连。举例来说，通信接口 **218** 可以是一个向相应类型的电话线路提供数据通信连接的综合业务数字网（ISDN）的网卡或是调制解调器。作为另一个实例，通信接口 **218** 可以是一个 LAN 网卡，它向兼容的 LAN 提供数据通信连接。此外还可以实施无线链路。在任何一种这类实施中，通信接口 **218** 都会收发电、电磁或光信号，这些信号传送的是那些代表不同类型信息的数字数据流。

网络链路 **220** 通常经由一个或多个网络来向其它数据设备提供数据通信。举例来说，网络链路 **220** 可以经由本地网络 **222** 而将一个连接提供给主机 **224** 或是互联网服务供应商（ISP）**226** 运作的数据设备。ISP **226** 进而又通过现在通常称为“互联网”的全球分组数据通信网络 **228** 来提供数据通信业务。本地网络 **222** 和互联网 **228** 都使用了传送数字数据流的电、电磁或光信号。经由不同网络的信号以及网络链路 **220** 上经由通信接口 **218** 的信号传送的是那些往返于计算机系统 **200** 的数字数据，而这些信号即为传送信息的载波的示范性形式。

计算机系统 **200** 可以经由一个或多个网络、网络链路 **220** 以及通信接口 **218** 来发送消息和接收数据，其中包括了程序代码。在互联网实例中，服务器 **230** 可以经由互联网 **228**、ISP **226**、本地网络 **222** 以及通信接口 **218** 来发送一个用于应用程序的被请求码。

接收到的代码可以在接收时由处理器 **204** 执行和/或存入存储设备 **210** 或其它非易失存储器中，以供稍后执行。这样，计算机系统 **200** 可以通过载波形式来获取应用码。

在前面的说明中，本发明的描述是参照特定的实施例的。但是，显而易见地是，可以对本发明进行各种修正和改变而不偏离本发明的广泛的精神和范围。因此，说明书和附图应该视为是示范性而不是限制意义的。

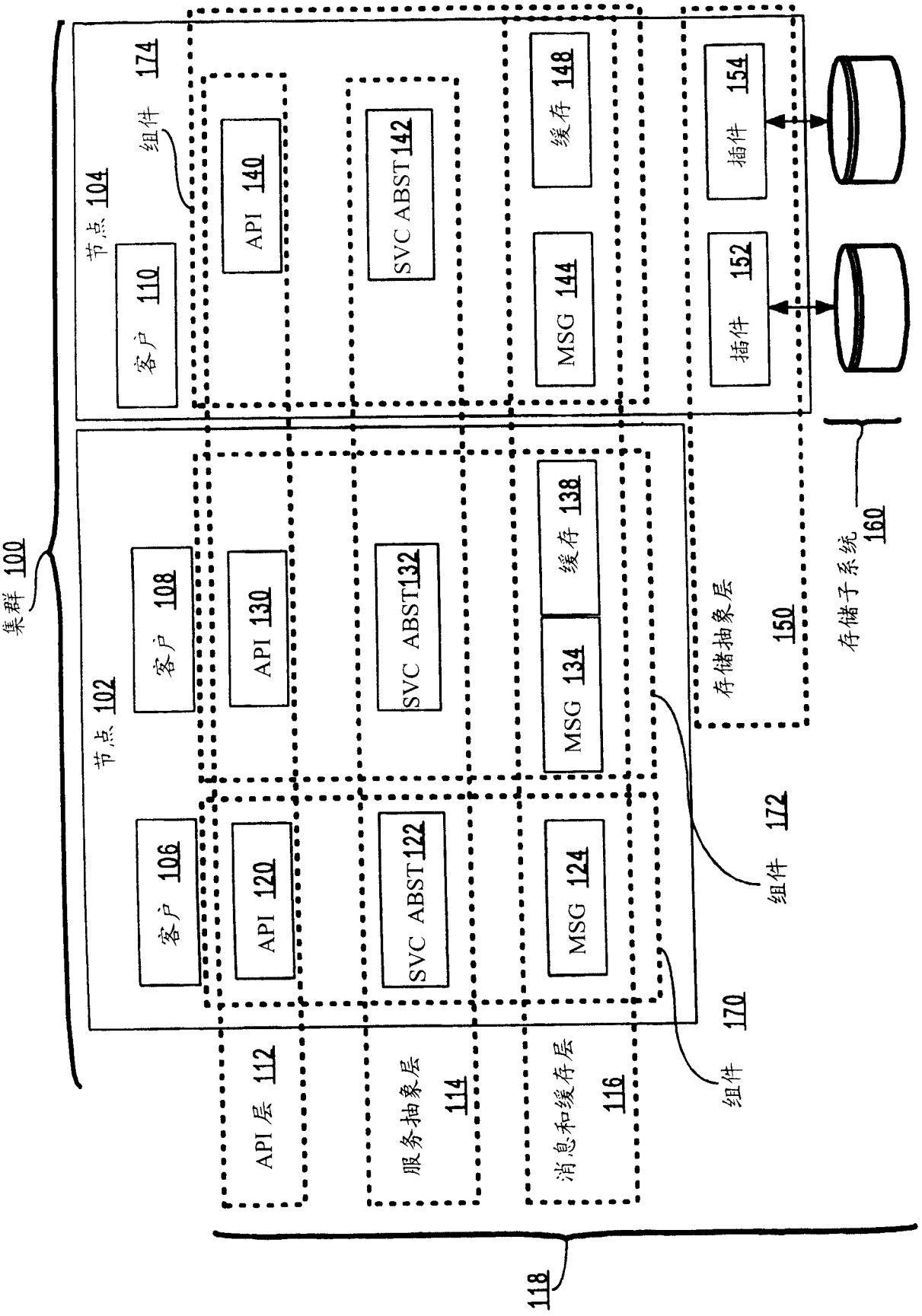

集群 100
节点 102
客户 106
API 120
SVC ABST122
MSG 124
客户 108
API 130
SVC ABST132
MSG 134
缓存 138
节点 104
客户 110
API 140
SVC ABST142
MSG 144
缓存 148
插件 152
插件 154
存储抽象层 150
组件 172
组件 174
组件 170
API 层 112
服务抽象层 114
消息和缓存层 116
118
存储子系统 160

图 1

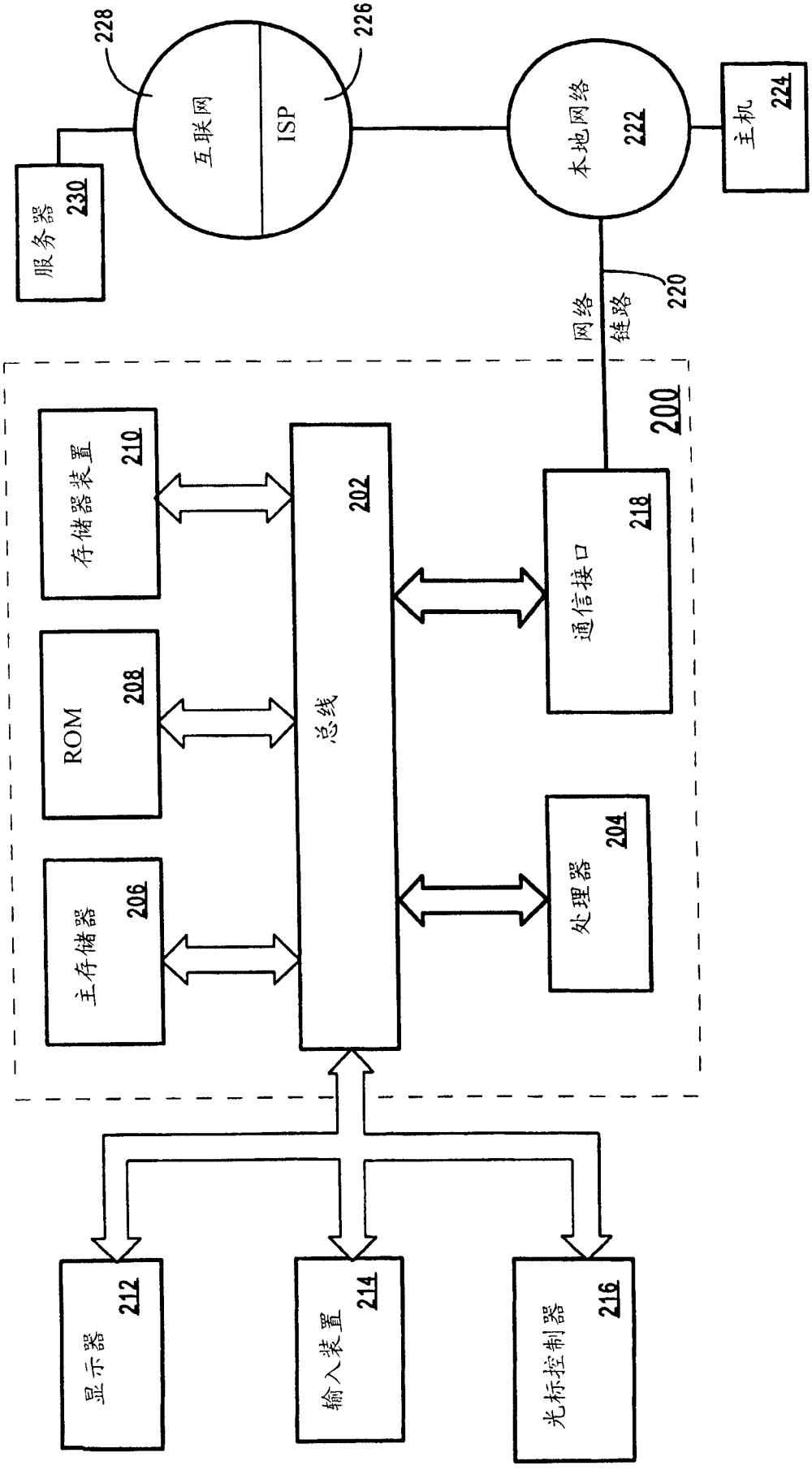

228
互联网
ISP
226
服务器
230
本地网络
222
主机
224
网络链路
220
200
存储器装置
210
202
通信接口
218
ROM
208
总线
主存储器
206
处理器
204
显示器
212
输入装置
214
光标控制器
216

图 2