(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105103159 A
(43)申请公布日 2015.11.25

(21)申请号 201480020417.2
(22)申请日 2014.03.11
(30)优先权数据
13/860,000 2013.04.10 US
(85)PCT国际申请进入国家阶段日
2015.10.09
(86)PCT国际申请的申请数据
PCT/IB2014/059621 2014.03.11
(87)PCT国际申请的公布数据
WO2014/167430 EN 2014.10.16
(71)申请人 国际商业机器公司
地址 美国纽约阿芒克
(72)发明人 R·O·耶雷尔 U·泽加尔
G·迪亚曼特
(74)专利代理机构 北京市金杜律师事务所 11256
代理人 王茂华 李峥宇
(51)Int.Cl.
G06F 21/60(2006.01)
G06F 17/30(2006.01)

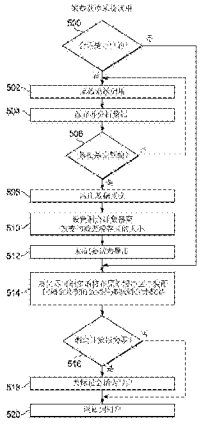
权利要求书2页 说明书11页 附图6页

(54)发明名称

为方便数据变换而导入系统调用数据

(57)摘要

本文中的技术提供被截获系统调用的“时移”,以便能够实现被截获系统调用到实际系统调用的一对多(1:n)或多对一(n:1)映射。需要在系统调用截获时出现的数据的逻辑边界(而不是物理边界)上实行的任何行动,导入(缓存)采取行动之前的数据,然后在完成时导出结果。该行动可能是相当不同的,例如检查数据、编辑数据、更改数据、限制数据、处理数据以及更新数据,等等。该技术可在数据库访问控制系统中实现。



CN 105103159 A

1. 一种用于在有存储器的计算系统中处理系统调用的方法，该系统调用具有关联的源和目的地，该方法包含：

截获该系统调用时，应用在硬件单元中执行的功能来导入与两个或更多的实际系统调用相关联的数据，直到导入了与该系统调用相关联的完整数据集合；

响应于收到对该完整数据集合实行变换的结果，通过将该结果导出至目的地而释放该系统调用。

2. 按照权利要求 1 的方法，其中，源是该系统调用，目的地是存储器的用户空间。

3. 按照权利要求 1 的方法，其中，源是存储器的用户空间，目的地是该系统调用。

4. 按照前述权利要求的任何之一的方法，进一步包含对该完整数据集合实行变换。

5. 按照权利要求 4 的方法，其中，该变换是以下之一：检查、分析、编辑、阻止、处理和更新。

6. 按照权利要求 4 的方法，其中，该两个或更多的实际系统调用导致数据被读取，且在返回到请求读取的应用之前对该完整数据集合实行该变换。

7. 按照权利要求 4 的方法，其中，该两个或更多的实际系统调用导致从应用写数据，且在发起第一系统调用写之前实行该变换。

8. 一种用于处理系统调用的装置，该系统调用具有关联的源和目的地，该装置包含：用于存储计算机程序指令的计算机存储器，当计算机程序指令被处理器执行时执行：

截获该系统调用时，应用一个在硬件单元中执行的功能，以导入与两个或更多的实际系统调用相关联的数据，直到导入了与该系统调用相关联的完整数据集合；

响应于收到对该完整数据集合实行变换的结果，通过将该结果导出至目的地而释放该系统调用。

9. 按照权利要求 8 的装置，其中，源是该系统调用，目的地是存储器的用户空间。

10. 按照权利要求 8 的装置，其中，源是存储器的用户空间，目的地是该系统调用。

11. 按照权利要求 8 至 10 的任何之一的装置，其中的指令进一步执行：对该完整数据集合实行变换。

12. 按照权利要求 11 的装置，其中，该变换是以下之一：检查、分析、编辑、阻止、处理和更新。

13. 按照权利要求 11 的装置，其中，该两个或更多的实际系统调用导致数据被读取，且在返回到请求读取的应用之前对该完整数据集合实行该变换。

14. 按照权利要求 11 的装置，其中，该两个或更多的实际系统调用导致从应用写数据，且在发起第一系统调用写之前实行该变换。

15. 一种存储在计算机存储介质上的、用于在数据处理系统中处理系统调用的计算机程序，该系统调用具有关联的源和目的地，该计算机程序包含计算机程序指令，当计算机程序指令被数据处理系统执行时执行以下步骤：

截获该系统调用时，应用一个在硬件单元中执行的功能，以导入与两个或更多的实际系统调用相关联的数据，直到导入了与该系统调用相关联的完整数据集合；

响应于收到对该完整数据集合实行变换的结果，通过将该结果导出至目的地而释放该系统调用。

16. 按照权利要求 15 的计算机程序，其中，源是该系统调用，目的地是存储器的用户空

间。

17. 按照权利要求 15 的计算机程序，其中，源是存储器的用户空间，目的地是该系统调用。

18. 按照权利要求 15 至 17 的任何之一的计算机程序，其中的指令进一步执行 ：对该完整数据集合实行变换。

19. 按照权利要求 18 的方法，其中，该变换是以下之一 ：检查、分析、编辑、阻止、处理和更新。

20. 按照权利要求 18 的计算机程序，其中，该两个或更多的实际系统调用导致数据被读取，且在返回到请求读取的应用之前对该完整数据集合实行该变换。

21. 按照权利要求 18 的计算机程序，其中，该两个或更多的实际系统调用导致从应用写数据，且在发起第一系统调用写之前实行该变换。

为方便数据变换而导入系统调用数据

技术领域

[0001] 优选实施例总体涉及保护诸如事务处理环境的分布式计算环境中的资源。

背景技术

[0002] 在计算环境中，“系统调用”是程序或进程通过其请求来自操作系统内核的服务的机制。在典型的操作系统 (OS) 中，如 Linux(Linux 是林纳斯·托瓦兹 (Linus Torvalds) 在美国和 / 或其他国家的注册商标）中，操作系统将虚拟存储器分隔为内核空间和用户空间，前者被保留用于运行操作系统内核、内核扩展、和设备驱动程序，后者是在所有用户模式的应用程序工作的地方。系统调用提供在用户空间中执行的程序或进程与在内核空间中执行的操作系统内核之间的接口。

[0003] 出于安全的原因，分析和修改被程序读或写的数据可能是有益的。如果需要进行过滤或编辑（或者更一般来说，某种其它变换），则需要截获流经系统的数据并对其采取行动。当截获系统调用时，可以将这个数据发送出去用于分析，甚至可以根据预先设定的规则被修改。当需要编辑时，最简单的方法是在被截获系统调用到实际系统调用的一对一 (1:1) 映射内实行编辑。有几款现有的商业产品执行此功能。

[0004] 然而，常常可能跨多个系统调用拆分单一事务的数据。例如，当为了对通过操作系统发送和接收的数据实行变换而截获系统调用时，不能保证被截获的数据的量包含整个逻辑包 (packet)。例如，考察一个正在接收 TCP 包的 MySQL 数据库。该包分两部分被接收，首先被接收的是包头 (header)，然后是包体 (body)。通常，包头首先被接收，并指示后续包中的字节数。现在，假设要编辑对一个数据库的查询，以将结果限制到特定的列（例如，诸如‘where EMPLOYEE = 20’的 SQL 语句）。要做到这一点，就要将该数据库查询修改得适合附加子句 (clause)。然而，在被截获系统调用到实际系统调用的 1:1 的映射中，已经将包头传递到数据库，因此要改变包的大小已为时太晚。更一般来说，可能跨多个系统调用拆分单一的（例如，从数据库读的）查询。无论所需的查询或功能的性质如何，一旦将系统调用中被读或写的数据返回到用户进程 (user process)，要修改数据就为时太晚了。这种限制使现有的解决方案不能改写或编辑跨越多个系统调用的被截获数据。

发明内容

[0005] 优选实施例提供一种在具有存储器的计算系统中的处理系统调用的方法，该系统调用具有关联的源和目的地，该方法包含：截获该系统调用时，应用在硬件单元中执行的功能来导入与两个或更多的实际系统调用相关联的数据，直到导入了与该系统调用相关联的完整数据集合；响应于收到对该完整数据集合实行变换的结果，通过将该结果导出至目的地而释放该系统调用。

[0006] 优选地，源是该系统调用，目的地是存储器的用户空间。更优选地，源是存储器的用户空间，目的地是该系统调用。更优选地，该方法进一步包含对该完整数据集合实行变换的步骤。更优选地，该变换是以下之一：检查、分析、编辑、阻止、处理和更新。

[0007] 优选地，该两个或更多的实际系统调用导致数据被读取，且在返回到请求读取的应用之前对该完整数据集合实行该变换。优选地，该两个或更多的实际系统调用导致从应用写数据，且在发起第一系统调用写之前实行该变换。

[0008] 本文中的技术提供被截获系统调用的“时移”(time-shifting)，以便能够实现被截获系统调用到实际系统调用的一对多 (1:n) 或多对一 (n:1) 映射。需要在系统调用截获时出现的数据的逻辑边界（而不是物理边界）上实行的任何行动，导入 (spool)（缓存）采取行动之前的数据，然后在完成时导出 (unspool) 结果。该行动可能是相当不同的，例如检查数据、编辑数据、更改数据、限制数据、处理数据以及更新数据，等等。

[0009] 本文所描述的方法假定系统调用具有相关的源和目的地。在“读”的情况下，源是原始系统调用本身，而目的地是用户空间。在“写”的情况下，源是用户空间，而目的地是原始系统调用。根据本公开，实际上对与系统调用关联的数据实施“保持”(hold)。特别地，在读的情况下，该保持导入数据，在所有数据都被读取之前不返回到用户空间。在写的情况下，该保持导入数据，直到所有将要被写入的数据都被从用户空间读出。以这样的方式保持数据，使得能够对调用被截获系统调用的用户进程透明地实行变换。此外，使用这种方法使得能够远程地进行数据的处理和实行变换，以例如避免存储器限制，从而能将不同的数据集合互相关联。

[0010] 以上概述了所公开主题的一些更相关的特点。这些特点应该被解释为仅仅是说明性的。正如下面所述的那样，可以通过以不同方式应用所公开的主题或者通过修改该主题而得到许多其它有益的结果。

附图说明

[0011] 为了更完整地理解本发明及其优点，现在结合附图作出以下描述，附图中：

[0012] 图 1 示出可以在其中可实现说明性实施例的分布式数据处理环境的示例性框图；

[0013] 图 2 是可以在其中可实现说明性实施例的数据处理系统的示例性框图；

[0014] 图 3 示出可在其中实践本公开的非限定性实施例中的技术的、已知的本地数据库访问控制系统 (LDACS) 的高级操作；

[0015] 图 4 例示一计算实体的存储器，其中可以实现本公开的导入 (spooling) 机制的功能；

[0016] 图 5 例示按照本发明的处理第一类（例如“读”）系统调用的处理流程；

[0017] 图 6 例示按照本发明的处理第二类（例如“写”）系统调用的处理流程。

具体实施方式

[0018] 现在参照各附图、特别是参照图 1-2，图中示出可以在其中实现本发明的示例性实施例的数据处理环境的示意图。应该明白，图 1-2 仅仅是示范性的，并非是要对本发明主题的各方面或实施例可以在其中实现的环境设置或暗示任何限制。在不脱离本发明的范围的情况下可以对所示出的环境作出许多修改。

[0019] 现在参看各附图。图 1 描述了一个典型的分布式数据处理系统的图形表示，其中可以实现说明性实施方案的各个方面。分布式数据处理系统 100 可以包括一个计算机网络，在该网络中，可以实现说明性实施例的各方面。分布式数据处理系统 100 包含至少一个

网络 102，这是用于提供在分布式数据处理系统 100 中连接在一起的各种设备和计算机之间的通信链路的介质。网络 102 可以包括连接，如有线、无线通信链路或光纤电缆

[0020] 在所示的例子中，服务器 104 和服务器 106 与存储单元 102 一起连接到网络 108。此外，客户端 110、112 和 114 也连接到网络 102。这些客户端 110、112 和 114 例如可以是个人电脑、网络计算机，等等。在所示的例子中，服务器 104 提供数据，例如引导文件、操作系统映像和应用程序给客户端 110、112 和 114。在所示的例子中，客户端 110、112 和 114 是服务器 104 的客户端。分布式数据处理系统 100 可以包括其他服务器、客户端和未予示出的其它设备。

[0021] 在所示的例子中，分布式数据处理系统 100 是互联网，网络 102 代表世界范围的网络连接和网关，它们利用传输控制协议 / 互联网协议 (TCP/IP) 的协议组彼此通信。在互联网的核心，是由成千上万的商业、政府、教育和其他路由数据和消息的计算机系统组成的主要节点或主机之间的高速数据通信线路的骨干。当然，分布式数据处理系统 100 也可以实现包括一些不同类型的网络，例如内部网、局域网 (LAN)、广域网 (LAN) 等等。如上所述，图 1 旨在作为一个例子，绝非对所公开的主题的不同实施方式的结构性限制，因此，图 1 中所示的特定元素，不应该视为是对本发明的说明性实施例可在其中实施的环境的限制。

[0022] 现在参看图 2，所示的是可以在其中实现说明性实施例的各方面的示例性数据处理系统的框图。数据处理系统 200 是计算机的一个例子，诸如图 1 中的客户端 110，其中可以安置实现本公开的说明性实施例的过程的计算机可用代码或指令。

[0023] 现在参看图 2，所示的是可以在其中实现说明性实施例的各方面的示例性数据处理系统的框图。数据处理系统 200 是计算机的一个例子，诸如图 1 中的服务器 104 和客户端 110，其中可以安置实现本公开的说明性实施例的过程的计算机可用代码或指令。在这个说明性例子中，数据处理系统 200 包括通信组织 202，它提供处理器单元 204、存储器 206、持久性存储器 208、通信单元 210、输入 / 输出 (I/O) 单元 212 和显示器 214 之间的通信。

[0024] 处理器单元 204 用于执行可加载到存储器 206 的软件的指令。处理器单元 204 可以是一个或多个处理器的集合，或者可以是一个多处理器核，具体取决于特定的实现。此外，处理器单元 204 可以用一个或多个异构处理器系统来实现，其中在单一芯片上存在一个主处理器与一个二级处理器。作为另一个示例性例子，处理器 204 可以是包含多个同类型的处理器系统的对称多处理器 (SMP)。

[0025] 存储器 206 和持久性存储器 208 是存储设备的例子。存储设备是任何一件能够临时地和 / 或永久地存储信息的硬件。在这些例子中，存储器可以是例如随机存取存储器或任何其他合适的易失性或非易失性储存设备。持久性存储器 208 可以根据特定的实现而采取各种形式。例如，持久性存储器 208 可以包含一个或多个组件或设备。例如，持久性存储器 208 可以是硬盘、闪存、可重写光盘、可重写磁带，或上述的某种组合。持久性存储器 208 使用的介质也可以是可移动的。例如，可移动硬盘驱动器可用于持久性存储器 208。

[0026] 在这些例子中，通信单元 210 提供与其他数据处理系统或设备的通信。在这些例子中，通信单元 210 是一个网络接口卡。通信单元 210 可以通过使用物理和无线通信链路的一种或二者而提供通信。

[0027] 输入 / 输出单元 212 允许对可以连接到数据处理系统 200 其他设备进行数据输入和输出。例如，输入 / 输出单元 212 可以通过键盘和鼠标为用户输入提供连接。此外，输入

/ 输出单元 212 可以发送输出到打印机。显示器 214 提供了一种向用户显示信息的机制。

[0028] 操作系统和应用程序或程序的指令位于持久性存储器 208 上。这些指令可以被加载到存储器 206 中，用于由处理器单元 204 执行。不同实施例的过程可以由处理器单元 204 用计算机实现的指令执行。计算机实现的指令可以位于存储器中，例如存储器 206 中。这些指令称为程序代码、计算机可用程序代码或计算机可读程序代码，它们可以被处理器单元 204 中的处理器读取并执行。不同的实施例中的程序代码，可以包含在不同的物理的或有形的计算机可读介质上，诸如存储器 206 或持久性存储器 208 上。

[0029] 程序代码 216 以功能形式位于可选择为移动式的计算机可读介质 218 上，并可以被加载到或传输到数据处理系统 200 供由处理器 204 执行。程序代码 216 和计算机可读介质 218 构成这些例子中的计算机程序产品 220。在一个例子中，计算机可读介质 218 可以是有形的形式，诸如插入或放置到作为持久性存储 208 一部分的驱动器或其他设备的光盘或磁盘，用于传输到诸如作为持久性存储 208 一部分的硬盘的存储设备。在有形的形式中，计算机可读介质 218 也可采取持久性存储器的形式，诸如连接到数据处理系统 200 的硬驱、U 盘或闪存。计算机可读介质 218 的有形的形式也被称为计算机可记录介质。在某些情况下，计算机可记录介质 218 可能不是可移动的。

[0030] 另外，程序代码 216 也可以从计算机可读介质 218 通过至通信单元 210 的通信链路和 / 或通过至输入 / 输出单元 212 的连接被传送到数据处理系统 200。在示例性例子中，该通信链路和 / 或该连接可以是物理的或无线的。计算机可读介质也可以采取非有形的介质的形式，诸如包含程序代码的通信链路或无线传输。所示出的用于数据处理系统 200 的不同组件，不意味着对可以实现的不同实施例的方式提出结构性限制。不同的说明性实施例可以在包括为数据处理系统 200 例示的那些组件以外的组件或替代为数据处理系统 200 例示的那些组件的组件的数据处理系统中实现。图 2 所示的其他组件可能与所示的示例性例子有所不同。作为一个例子，数据处理系统 200 中的存储设备是可以存储数据的任何硬件设备。存储器 206、持久性存储器 208 和计算机可读介质 218 是有形的形式的存储设备的例子。

[0031] 在另一个例子中，可以用一个总线系统来实现通信组织 202，并且总线系统可以由诸如系统总线或输入 / 输出总线的一个或多个总线组成。当然，总线系统可以使用提供在附接到总线系统的不同组件或设备之间传输数据的任何合适类型的架构来实现。此外，通信单元还可以包括用于发送和接收数据的一个或多个设备，诸如调制解调器或网络适配器。此外，存储器例如可以是诸如可能存在于通信组织 202 中的接口和内存控制器集线器 (hub) 中的存储器或高速缓存。

[0032] 用于执行本发明的操作的计算机程序代码，可以用一种或多种编程语言的任意组合编写，编程语言包括面向对象的编程语言，诸如 Java(Java 和基于 Java 的商标和标识是甲骨文公司和 / 或其子公司的商标或注册商标)、Smalltalk、C++ 等，也包括传统的过程化编程语言，诸如“C”编程语言或类似的编程语言。程序代码可以完全地在用户的计算机上执行，部分地在用户的计算机上执行，作为独立的软件包执行，部分地在用户的计算机上并且部分地在远程计算机上执行，或完全在远程计算机或服务器上执行。在后一种情况下，远程计算机可以通过任何类型的网络—包括局域网 (LAN) 或广域网 (WAN)—连接到用户的计算机，或者 (例如利用因特网服务供应商而通过因特网) 连接到外部计算机。

[0033] 本领域的普通技术人员将明白，图 1-2 中的硬件，可以依具体实现而有所不同。其他内部硬件或外围设备，诸如闪存、等同的非易失性存储器或光盘驱动器等，可在图 1-2 所示的硬件以外使用，或者替代图 1-2 所示的硬件而使用。此外，在不偏离所公开的主题的范围的情况下，示例性实施例的过程也可应用于与先前所提到的非对称多处理 (SMP) 系统不同的多处理器数据处理系统。

[0034] 正如将要看到的，本文中所描述的技术可以在诸如图 1 所示标准的客户机 - 服务器范例中进行操作，其中的客户机与在一个或多个机器构成的集合上执行的可访问因特网的基于网络的门户网站进行通信。终端用户操作能够访问门户网站和并能与门户网站互动的可连接因特网的设备（例如台式计算机、笔记本电脑、能上网的移动设备，等等）。通常情况下，每个客户机或服务器计算机都是一个如图 2 所示的、包含硬件和软件的数据处理系统，这些实体通过网络，诸如互联网、内联网、外联网、专用网的网络或任何其他通信介质或链接，互相交流。数据处理系统通常包括一个或多个处理器、一个操作系统、一个或多个应用程序，以及一个或多个公用程序 (utilities)。数据处理系统上的应用程序提供对 Web 服务本地支持，包括但不限于对 HTTP、SOAP、XML、WSDL、UDDI、和 WSFL 等等的支持。有关 SOAP、WSDL、UDDI 和 WSFL 的资料可以从万维网联盟 (W3C) 获得，后者负责开发和维护这些标准；关于 HTTP 和 XML 的进一步信息可以从因特网工程任务组 (IETF) 获得。假设熟悉这些标准。

[0035] 在一个代表性但非限制性的实现中，这里的技术是在一个包含为响应 HTTP 和 Web 服务客户端用户服务或事务请求而协同工作的分布式主机组件 (mainframe components) 的事务处理系统或环境的上下文中描述的。这样的系统或环境通常包含以分布式的方式配置的多个组件。更大的多组件事务处理环境的分布式组件通常至少包含计算机、操作系统平台、应用程序、网络和相关安全引擎，其提供分布式事务处理功能—如与客户端用户的网络相互作用，以及在 HTTP 和 Web 服务的情况中的识别和认证功能。这种类型的事务处理系统或环境通常也包括一个主机组件，其至少包含计算机、操作系统平台、应用程序、网络和提供高性能的后端事务处理和大型数据库功能的相关安全引擎。

[0036] 虽然并非是限制性的，本发明的技术（下面会更详细地描述）可以在已知的数据库访问控制系统 (LDACS) 架构中实现。LDACS 也便于针对通过网络可访问的数据库进行审计和记录 (logging) 操作。在这个方法中，通常将进程间通信 (IPC) 截获 (intercept) 定义为确定本地和远程数据库 (DB) 公用接入点的访问尝试。对数据库的本地访问尝试被截获并传送到一个可用于访问尝试的网络监视的数据安全设备。因为该数据安全设备是远程的，它有时被称为“外部安全设备”（或“ESD”）。IPC 截获通过一个最小足迹实现对象来执行本地访问尝试的截获，以减少资源开销。以这种方式，远程网络数据安全设备观察到通过数据库主机 (DB host) 处的截获的本地访问尝试和被截获的对该数据安全装置的访问尝试的传输，以及通过网络的远程访问尝试，由此整合通过该数据安全设备对数据库资源的数据访问尝试的分析和记录。

[0037] 图 3 示出可在其中实践本公开的技术的 LDACS 体系结构和操作。参考图 3，环境 300 提供远程用户 302 以用于数据存储和检索操作 (DB 操作) 的数据库 (DB) 主机 304。用户 302 的通过接入网络 306 连接到主机 304，接入网络 306 可以是任何合适的网络基础设施，如局域网、内联网、外联网或因特网。数据库主机 304 包括一个连接到数据库 310 的数

据库服务器 308，数据库 310 通常是一个磁盘阵列或一组大容量存储设备，如磁盘驱动器。数据库 308 包括一个数据库访问网关 312，它以应用程序编程接口 (API) 的形式操作，用于用户 302 通过各种访问方法进行访问。

[0038] 用户以用户请求 314 的形式发起对数据库的访问，用户请求 314 经过网络 306，作为输入请求 316 发送到数据库访问网关 312。数据安全装置 320 通过从网络 306 到主机 304 的路径上的交换机 e22 或其他连接设备—如抽头连接 (tap)、路由器或桥接器 (bridge)—而连接。数据安全装置 320 包括用于接收通过交换机 322 发送的用户请求 314 的数据库监控器 324。数据库监控器接收并分析以抽头访问尝试 318 的形式输入的用户请求 314，数据库监控器 324 根据预定的安全或访问策略分析之。数据安全设备 320 然后把抽头访问尝试 318 作为输入请求 116 的形式传送到访问网关 (AG)312。

[0039] 一般来说，数据库服务器 308 预计相当一部分的数据库流量 (用户请求 314) 将通过网络 306 远程地到达，因此要通过数据安全设备 320 的检查。然而，一部分的数据库访问尝试是从主机 304 上执行的本地客户端 330 以本地访问尝试 332 的形式发出的。本地访问尝试 332 通过进程间通信 (IPC) 机制 334 到达访问网关 312。这种本地访问尝试 332 不经过交换机 322，因此可能会另行地规避数据安全装置 320 的检查。为了解决这个问题，一个已知的 IDACS 解决方案采用 IPC 截获 (intercept)340 来截获本地访问尝试 332 和把所截获的访问尝试 342 传输到本地代理 350。本地代理 350 通过询问 IPC 机制 334 确定对应该本地访问尝试 332 的数据库指令 352。本地代理 350 然后将所确定的数据库指令 352 发送到数据安全装置 320，用于由数据库监控器 324 进行分析和进一步操作。以这种方式，数据安全装置 320 接收对数据库服务器 308 的所有本地和远程访问尝试，以更充分地分析、监测和防范可能是不可取的访问尝试。虽然上面描述的配置是优选的，代理 350 不必是本地的，而是可以被定位在与数据库主机或系统关联的其他位置或配置中。

[0040] 在典型的数据库主机 304 中，本地客户端 330 可采用多种 IPC 机制 334 来发送本地访问尝试 332 至数据库服务器 308。IPC 通常是不安全的。替代的配置可以采用其他通信机制，如加密的远程方法调用。

[0041] 如上所述，LCADS 的描述，并非旨在是限制性的。现在所描述的技术，可以实现在其他类型的解决方案中，例如审计和合规系统、变更控制解决方案、漏洞管理解决方案、防欺诈解决方案、数据库泄漏预防解决方案等中。

[0042] 更一般来说，此处所描述的技术可以在这样的任何计算环境中实现：其中一个程序或进程例如使用系统调用来请求操作系统的内核的一个服务 (例如读、写或类似的服务)，并且需要在与该服务相关联的数据的逻辑边界上实行某种行动。

[0043] 导入系统调用数据

[0044] 图 4 例示一个具有代表性的计算实体的系统存储结构，在其中可以实现本文所述的技术。该计算实体可以是上面参照图 2 所述的类型。它包括一个处理器 (未予示出)，和一个存储器 400，存储器 400 被组织得包括一个虚拟存储器 402，虚拟存储器又被分隔成内核空间 404 和用户空间 406。内核支持一个操作系统内核 408。进程或程序 410 在用户空间 406 中执行。系统调用诸如系统调用 412，提供在用户空间 406 中执行的程序或进程 410 与在内核空间 404 中执行的操作系统内核 408 之间的接口。

[0045] 按照优选实施例中，提供一种机制 414，用于导入系统调用数据，例如用于诸如分

析和编辑的数据变换。导入机制 414 提供被截获系统调用的“时移”，以便能实现被截获系统调用与实际系统调用的一对多 (1:n) 或多对一 (n:1) 映射。在这种方法中，需要在系统调用截获时出现的数据的逻辑边界（而不是物理边界）上实行的任何行动，在采取该行动前先导入（缓存）数据，完成时再导出 (unspools) 行动的结果。该行动可能是相当不同的，例如检查数据、编辑数据、更改数据、限制数据、处理数据、更新数据，等等，并且，该行动可以本地地或远程地执行。远程处理方法的一个示例性实施例是在以上参照图 3 描述的方法。

[0046] 如前文所述，该方法假定系统调用 414 具有相关的源和目的地。在“读”的情况下，源是原始系统调用本身，而目的地是用户空间 406。在“写”的情况下，源是用户空间，而目的地是原始系统调用。导入机制 414 在系统调用 412 相关联的数据上设置一个“保持”。特别地，在读的情况下，导入机制 414 导入该数据，在所有的数据已被读取之前不返回到用户空间。在写的情况下，导入机制 414 导入该数据，直到所有将要被写入的数据都被从用户空间 406 读取。以这种方式保持数据，使得能对调用被截获系统调用的用户进程 410 透明地实行变换。此外，使用这种方法，使得能够远程地进行数据的处理和实行该变换，以例如避免存储器限制，从而使不同的数据集合可以互相关联。

[0047] 导入机制 414 通过挂钩 (hooking) 系统调用而操作。如下将更详细地描述的那样，该机制包括一个分析器功能 (function)。当一个系统调用 412 被挂钩时，导入机制功能的地址被插在原始功能的地址的位置，而（该原始功能的）的那个地址则由操作系统保存。然后，当需要调用（返回到）原始系统调用时，在挂钩过程中被保存的原始系统调用的地址就被使用。为了解释的目的，一个或多个数据构造或结构，随着本公开的时移方法的执行而被缓冲机制维持和处理。这些数据结构有时被称为“缓冲区”(buffers)，因此，这里所使用的“原始缓冲区”，指的是最初经过操作系统内核 408 的、并且通过一个或多个系统调用被发现的数据。可能有多个原始缓冲区。这个数据之所以被认为是经过，是因为在读的意义上以及从系统调用挂钩的角度来看，该数据源于内核的原始系统调用，而在写的意义上，该数据源于用户。“改变的数据缓冲区”指的是来自一个或多个原始缓冲区的、已经实行了变换的数据。“剩余计数器”指的是留在原始缓冲区中的、仍然需要被转移到无论是原始系统调用（对于读而言）还是或用户（对于写而言）的数据的量。“导入”(spooling) 的意思是缓存 (buffering)。因此，在导入机制的运行中，数据（原始缓冲区）被收集，连接在一起（至工作缓冲区中），实行变换，将结果存储到改变的数据缓冲区，然后，改变的数据缓冲区被抽排回（或“导出”）到原始缓冲区原本要去的地方（即，对于读而言的原始系统调用，对于写而言的用户）。

[0048] 在被截获系统调用中，感兴趣的数据是与调用相关联的数据。对于一个读系统调用（例如，recv、read 等）来说，该数据只有在原始系统调用被调用之后才是可用的。对一个写系统调用（例如，send、write 等）来说，该数据被作为参数传入。在这两个情况下，都没有办法控制向系统调用提供的缓冲区的大小。因为这个数据以不同的方式存在，读和写系统调用必须以不同的方式进行处理，尽管一般性概念是相似的。特别地，在这两种情况下，导入 / 导出方法的应用都非常相似，主要差别在于发生（被截获系统调用到实际系统调用的）(1:n) 映射之处。在读系统调用的情形中，整个逻辑数据包（例如数据库查询）需要（在将任何数据返回到用户之前）被收集，以便能实行任何变换。特别地，调用该系统调用

的用户进程已经提供了要被填充以数据的缓冲区。因为需要在用户接收数据之前对该数据实行一个变换，被读数据被一直保持到能实行该变换。因为不能保证整个逻辑数据包将适合所提供缓冲区（也不能保证变换了的数据也适合），按照本公开，数据要被一直保存到整个原始数据包都被接收，数据被发送出去用于分析和修改（或其他变换），变换了的数据被排出（导出）回至用户。为此，如将在图 5 中描述的那样，通过下述方式完成，即，反复调用被截获系统调用内的原始系统调用，直到（导入机制内的）分析器功能已经检测到该包是完整的。然后，（例如由分析器）实行变换，将改变了的数据返回到被截获的系统调用。将所提供的缓冲区填充以变换了的数据，返回给用户。如果整个变换了的包在缓冲区内不能适合，则将缓冲区填充到容量，由用户进程作出的随后的系统调用将不再次调用原始系统调用，相反只是简单地以来自变换了的包中的更多数据填充所提供的缓冲区，直到该包的全部都被返回，此时再次开始该循环。

[0049] 图 5 详细说明了在截获时被截获读系统的这个过程。该例程在步骤 500 通过测试一个会话 (session) 是否是导出的 (unspooling) 而开始。如果在步骤 500 的测试的结果是否定的，该例程将继续，在步骤 502 分析原始系统调用。在步骤 504 中，原始系统调用中的数据被保存。然后，在步骤 506 执行测试，以确定数据是否完整。如果在步骤 506 的测试的结果为否定的，该例程周期的这个部分返回到步骤 502，以从原始系统调用获得额外的数据。然而，如果在步骤 506 的测试的结果是肯定的，这表明所有的数据是完整的，该例程继续在步骤 508 实行数据变换。这种变换型的性质可能是相当不同的，例如，检查、编辑、改变、限制、处理、更新，等等。对变换的性质和范围没有任何限制，这是一个已知的函数 / 运算。实行变换后，该例程继续在步骤 510 将一个剩余计数器设置到一个改变的数据缓冲区的大小。该例程然后继续在步骤 512 把该会话开始标记为导出的。控制然后在步骤 514 继续，如果在步骤 500 的测试的结果是肯定的，也到达这个步骤。在步骤 514 中，复制将在原始缓冲区安装 (fit) 的尽可能多的剩余的变化了的数据，并递减剩余计数器。然后在步骤 516 进行测试，以确定剩余计数器是否为零。如果是这样的话，在步骤 518 将该会话去除导出标记。该例程然后继续在步骤 520 返回到用户，在步骤 516 的测试的结果为否定的时，也到达这个步骤。这就完成了对截获读系统调用情形的处理。

[0050] 在写系统调用的情况中，在调用原始系统调用之前，需要收集整个逻辑数据包，以便能够实行变换。在这种情况下，调用系统调用的用户进程已经提供了一个具有要被发送的数据的缓冲区。因为在数据被传输之前，需要对数据实行变换，所以，在能进行变换之前，需要把原始数据包全部收集在一起。如上所述，数据包是有可能分散在多个系统调用中的。也有这样的可能，即在实行变换后，新的包将不适合 (fit) 由用户进程提供的缓冲区。因此，直到分析器确定整个数据包已被接收，该机制收集来自用户进程的数据，并从被截获系统调用返回，如同该数据被成功传输一样。分析器然后实行变换，并将修改后的数据发送回原始系统调用（即发送原始包的最后一部分的用户进程的调用）。如果新的包在调用过程所提供的缓冲区内不适合，则调用该系统调用，必要时反复调用，而不返回到用户，直到变换了的包的最后部分已经被发送。

[0051] 图 6 说明了在截获时写系统调用情况的详细处理流程。例程在步骤 600 通过保存和分析数据而开始。在步骤 602，进行测试，以确定数据是否完整。如果在步骤 602 中的测试的结果是肯定的（数据完整），例程继续在步骤 604 去标记该会话为导入。例程继续在步

骤 606 实行一个数据变换，如上所述的那样。在步骤 608，将剩余计数器设置为改变了的数据缓冲区的大小。例程继续在步骤 610 复制将在原始缓冲区安装的尽可能多的剩余的变化了的数据，并递减剩余计数器。步骤 610 后，例程返回到原始系统调用。这是步骤 612。然后在步骤 614 进行测试，以确定剩余计数器是否为零。如果不是，控制在步骤 616 返回到用户。但是，如果剩余的计数器为零，则控制返回到用户 616。如果在步骤 602 的测试的结果是否定的，也到达步骤 616，在这种情况下，在步骤 618 标记该会话标记为导入，然后返回到用户。这就完成了对截获写系统调用情形的处理。

[0052] 换另一种方式来讲，在读的情况下，有一个被截获系统调用到实际系统调用的 (1:n) 映射，其中将一个被截获系统调用转换为多个被调度的 (dispatched) 调用，直到完整的数据集合被检索并且然后被返回（跨 n 个被截获呼叫）而不带对应的调度系统调用。在写的情况下，有一个被截获系统调用到实际系统调用的 (n:1) 映射，其中 n-1 个被截获调用没有对应的被调度的调用，第 n 个被截获调用包含数据集合的最后一部分，m 个被调度的调用在最后的被截获调用返回之前被执行。一旦数据被导入，就可以实行变换（包括不变换）。作为一个例子，假设有一例简单的基于 SQL 的数据库查询“select*from employee where dept = 20”。如果向系统调用提交的数据被分割到含有字符串“select*from employee”和“where dept = 20”的两个缓冲区中，这两个字符串实际是同一查询的分部；进一步假设变换包括一个规则，即“Replace ‘dept = 20’ with ‘dept = 21’ when object = ‘employee’”（当 object = ‘employee 时，用 ‘dept = 21’ 替换 ‘dept = 20’）。另一种可能的变换可能是“在试图访问部门“20”中的对象 ‘employee’ 时放弃一个会话”（“Drop a session when attempting to access object ‘employee’ from dept. ‘20’”）。使用本文的方法，可以对由用户应用程序提供的数据实行这些类型的变换，而没有由传递协议 (delivery protocol) 插入的中断 (breaks) 或其他应用程序的限制。

[0053] 因此，如已经描述的那样，导入机制是用于被截获系统调用的时移装置，具有对发送和接收的原始数据的可选的修改。正如在问题陈述中所描述的，为了在通过操作系统发送和接收的数据上实行变换的目的而截获系统调用时，不能保证截获的数据量包含整个逻辑包。例如，当截获数据库的 TCP 流量 (traffic) 时，单一的查询可能被拆分在多个读调用中。根据本公开的方法，不是执行被截获系统调用到实际系统调用的 1:1 映射，在读期间，被截获的数据被分析，原始系统调用按需要被重新调用，直到所有的数据都被读取；然后，就能通过操作该完整的集合而实行一个变换，然后通过重新填充原始缓冲区而返回原始调用者，可能跨多个系统调用。这使得能对调用被截获系统调用的用户进程透明地实行该变换。

[0054] 此外，为了避免存储器限制，可以远程地处理数据和实行变换，从而使不同的数据集合能与彼此关联。这种方法可以用在图 3 中的 LDACS 解决方案中。因此，举例来说，这种数据关联的一个应用是保护主机数据库不受未经授权的访问。该变换被用来保护用标准的访问控制难以保护或不可能保护的数据元素。例如，通过关联数据，用户的与数据库的整个会话都可以通过从打开到关闭跟踪之而被观察。这个被授权的用户可能需要访问包含未予授权查看或修改的数据子集的表。通过变换查询，受保护的数据可以得到自动保护（例如，不用’select*from table’，而用’select*from table where...’）。然而，众所周知，关联步骤和变换步骤是复杂的，需要保持会话期间中发生的情况的历史。这过于复杂，以致不能

向被截获系统调用插入直接插入码(in-line),因此,该LDACS解决方案使得能够提供单独的收集器/分析器,用于处理数据和把该信息反馈给系统调用。因为这个过程需要对用户透明,因此,本公开的技术被用来“暂停”原始系统调用,直到处理完成。

[0055] 这里描述的机制的用途不止是数据库查询的编辑。这个机制可用于基于数据的内容为数据设置防火墙、选择性地审计,等等。更一般来说,这种方法可用来方便需要在与系统调用截获一起出现的数据的逻辑边界(而不是物理边界)上实行的任何变换—其中在实行该变换之前需要导入数据(并且然后要在结束时导出结果)。

[0056] 本文的技术提供了许多优点。通过创建被截获系统调用到到实际系统调用的一对多(复用的)映射,可以更有效地实行变换,即使被截获数据跨越一个以上的系统调用。在1:n映射中(读情况下),实际系统调用被按需要多次重新调用,然后再返回到用户空间。特别地,该技术一直读到整个包被收集为止,然后实行该变换,然后将修改后的数据(包括修改的包头)返回到用户空间。

[0057] 以上描述的缓冲机制的功能可以以内核修改的形式而实现,以独立方法的形式而实现,或以它们的某种组合的形式而实现。在任一情况下,该变换都可以被实现为由处理器执行的基于软件的功能,或者可以作为一个托管服务(包括通过SOAP/XML接口的网络服务)而提供。本文所描述的特定的硬件和软件实现细节仅用于说明性目的,而不是用来限制所描述的主题的范围。

[0058] 不作为对前述内容的限制,在其中可以使用本发明方法的代表性的LDACS解决方案是IBM® InfoSphere® Guardium(IBM、InfoSphere和Guardium是国际商业机器公司的注册商标)第8.0版。该解决方案包括一个可扩展的架构,包括收集器、聚合器和中央管理设备,以及安装在数据库服务器上的软件开发(S-Tap)代理和与内核相关联地安装的内核开发(K-Tap)代理。K-Tap收集数据,S-Tap将被收集数据传送到实行变换(和接收结果)所在的点。

[0059] 本文中所描述的技术,除了读和写,还可以针对其他类型的系统调用实现。

[0060] 更一般来说,在所公开的发明的上下文中的计算设备各是包含硬件和软件的数据处理系统(如图2所示),这些实体通过诸如互联网、内联网、外联网、专用网或任何其他通信介质或链接的网络互相通信。数据处理系统上的应用提供对Web和其他已知的服务和协议的本地支持,包括但不限于对HTTP、FTP、SMTP、XML、SOAP、WSDL、UDDI、和WSFL等等的支持。有关SOAP、WSDL、UDDI和WSFL的资料可以从万维网联盟(W3C)获得,后者负责开发和维护这些标准;关于HTTP、FTP、SMTP和XML的进一步信息可以从因特网工程任务组(IETF)获得。假设熟悉这些已知的标准和协议。

[0061] 本文中描述的方案可能在或者协同各种服务器端架构实现,包括简单的多层体系结构、门户网站、联邦系统等等。正如所指出的,本文中的技术可以在松散耦合的服务器(包括基于“云”的)环境中实现。安全服务器本身(或其功能,诸如监控器过程)可以在云中被托管。

[0062] 更一般来说,本文中所述的主题可以采取完全硬件实施例的形式、完全软件实施例的形式或者同时包含硬件单元和软件单元的形式。在一个优选的实施例中,功能以软件来实现,该软件包括但不限于固件、驻留软件、微代码等。此外,如上所述,分析引擎功能可以采取可从计算机可用的或计算机可读的介质访问的计算机程序产品的形式,其中该介质

提供用于由计算机或任意指令执行系统使用或者结合计算机或任何指令系统使用的程序代码。出于该描述的目的，计算机可用的或计算机可读的介质可以是任意一种能包含、存储用于由指令执行系统、装置或设备使用或者结合指令执行系统、装置或设备使用的程序的装置。介质可以是电的、磁的、光的、电磁的、红外的或半导体的系统（或装置或设备）或者传播媒介。计算机可读介质的例子包括半导体或固态存储器、磁带、可移动计算机磁盘、随机存取存储器 (RAM)、只读存储器 (ROM)、硬磁盘和光盘。光盘的当前例子包括压缩盘－只读存储器 (CD-ROM)、可读写压缩盘 (CD-R/W) 和 DVD。计算机可读介质是有形的介质。

[0063] 计算机程序产品可以是具有用来实现一个或多个所描述的功能的程序指令（或程序代码）的产品。这些指令或代码可以在通过网络被从远程数据处理系统下载后存储在数据处理系统中的计算机可读存储介质中。或者，这些指令或代码可以存储在服务器数据处理系统中的计算机可读存储介质中，并适于通过网络被下载到远程数据系统以用于远程系统内的计算机可读存储介质中。

[0064] 在一个典型实施例中，导入机制的组件是在专用计算机中实现的，优选地在由一个或多个处理器上执行的软件中实现。该软件被维护在与一个或多个处理器相关的一个或多个数据存储库或存储器中，并且该软件可以被作为一个或多个计算机程序来实现。总的来说，在一个实施例中这个专用硬件和软件包含上文所述的导入机制。

[0065] 尽管上文描述了本发明的某些实施例执行的操作特定顺序，应该理解，这样的顺序是示例性的，因为替代实施例可以以不同的顺序执行这些操作，组合某些操作，重叠某些操作，等等。在说明书中对一个给定的实施例中的引用表明所描述的实施例包括一个特定的特征、结构或特点，但是每一个实施例可能不一定非要包括该特定的特征、结构或特点。

[0066] 最后，尽管已经单独地描述了给定的系统组件，本领域普通技术人员应当理解，在给定的指令、程序序列、代码部分等中有些功能可以被组合或共享。

[0067] 本文所披露的技术并不局限于多组件事务处理环境，但这将是一个典型的实现。如上所述，上述功能可用于其中的服务器设置的会话管理数据可能被（不同会话中的原始用户或另一个用户）通过相同的客户端浏览器重新使用的任何系统、设备、门户、网站等等。

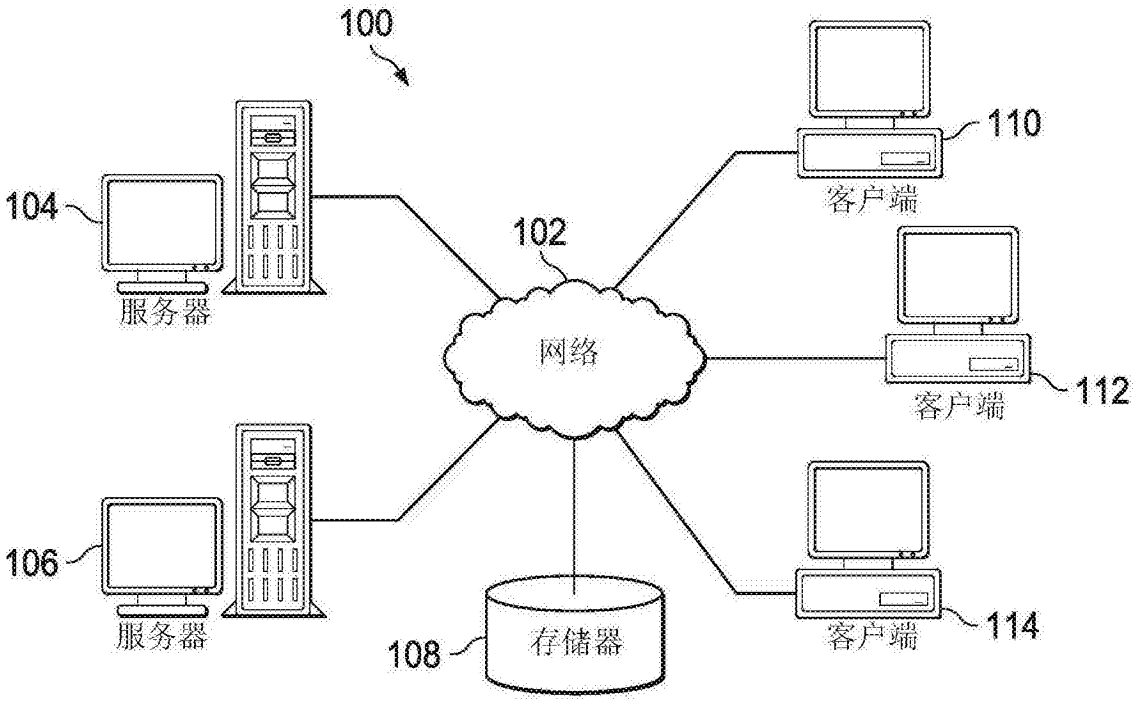


图 1

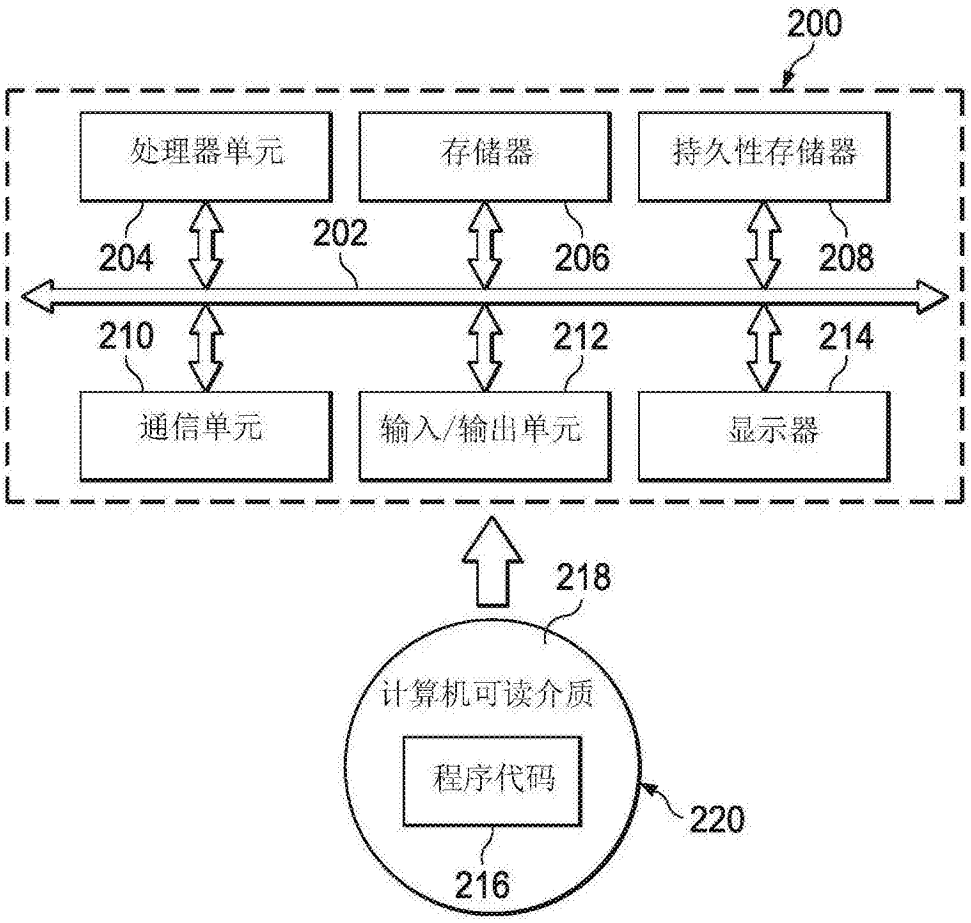

200
处理器单元
存储器
持久性存储器
204
202
206
208
210
212
214
通信单元
输入/输出单元
显示器
218
计算机可读介质
程序代码
220
216

图 2

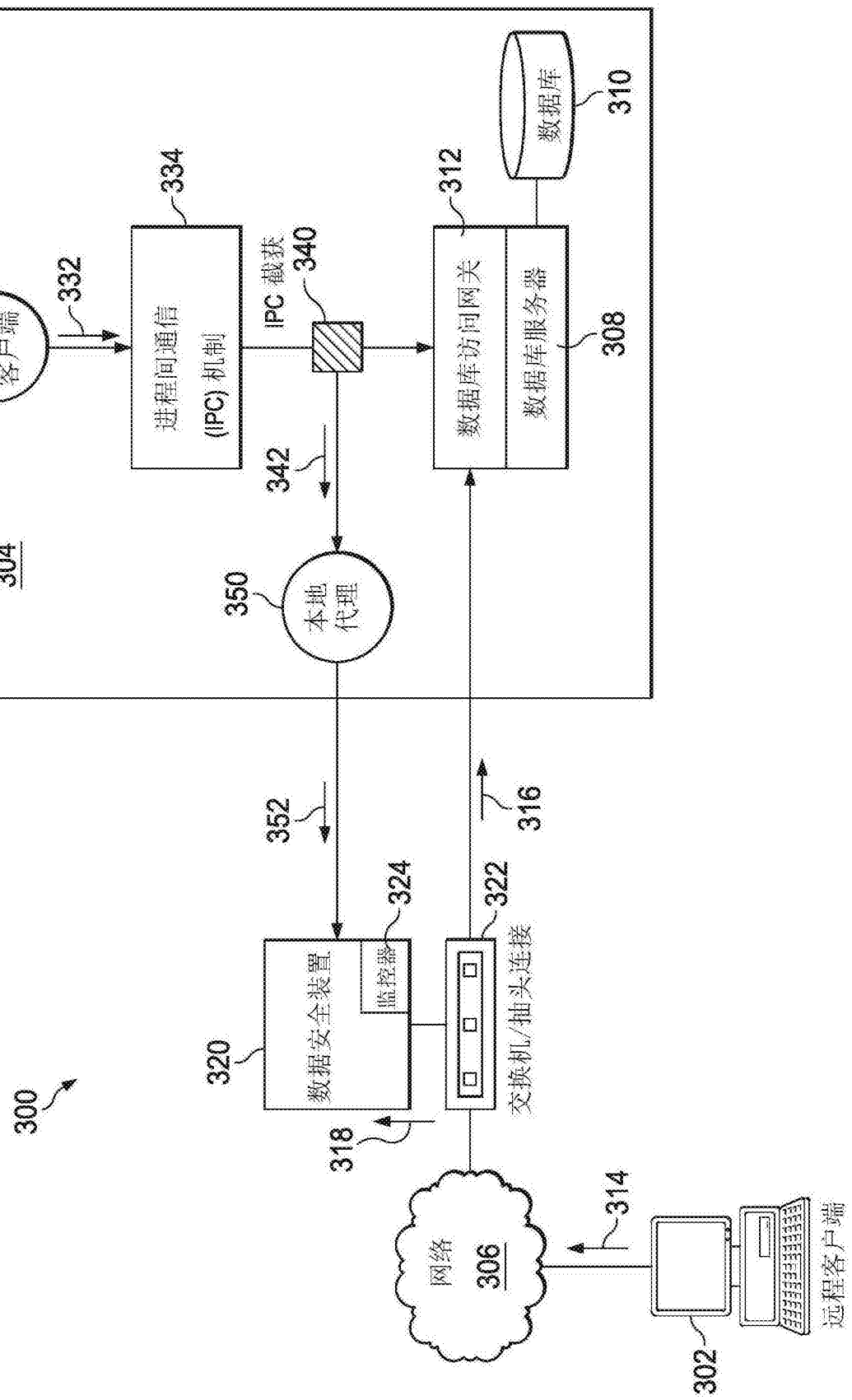

数据库主机
304
本地客户端
330
332
进程间通信
(IPC) 机制
334
IPC 截获
340
数据库访问网关
312
数据库服务器
308
数据库
310
342
本地代理
350
352
316
数据安全装置
320
监控器
324
交换机/抽头连接
322
318
网络
306
314
远程客户端
302
300

图 3

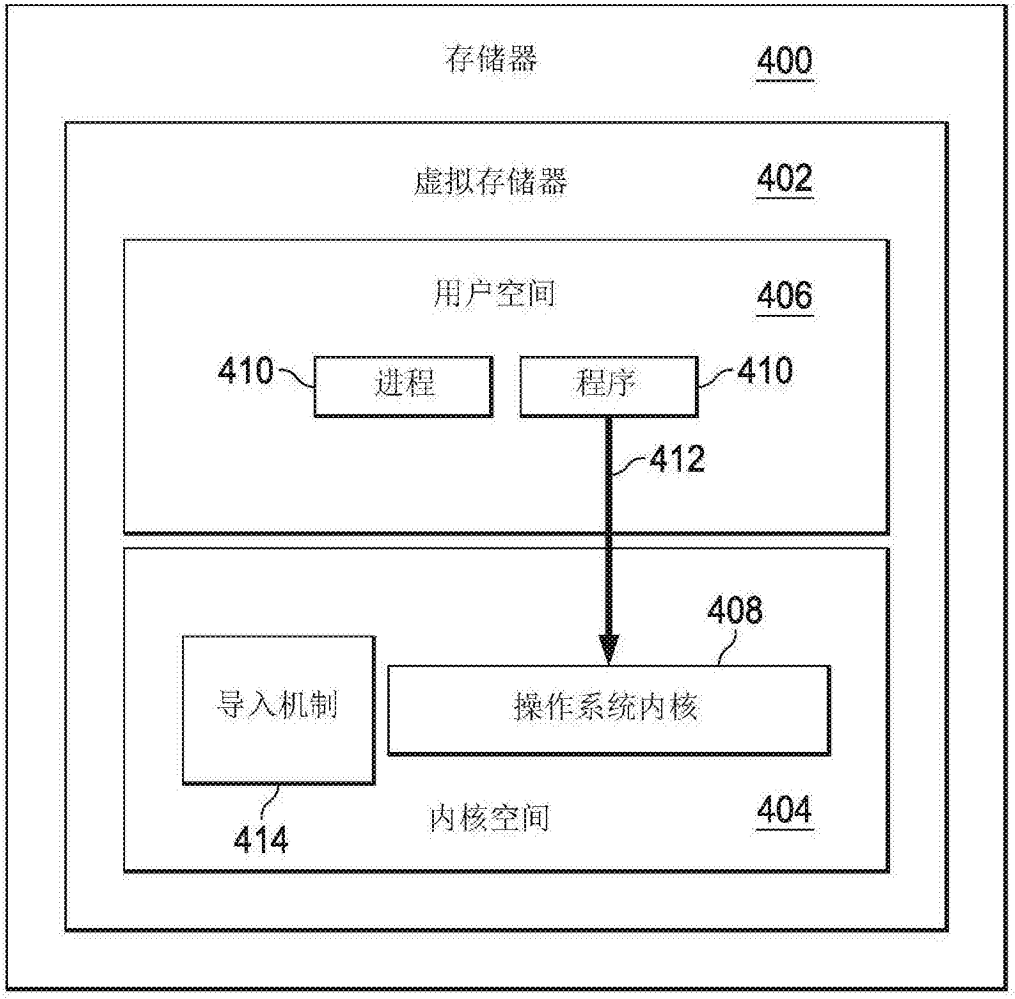

存储器
400
虚拟存储器
402
用户空间
406
410
进程
程序
410
412
408
导入机制
操作系统内核
414
内核空间
404

图 4

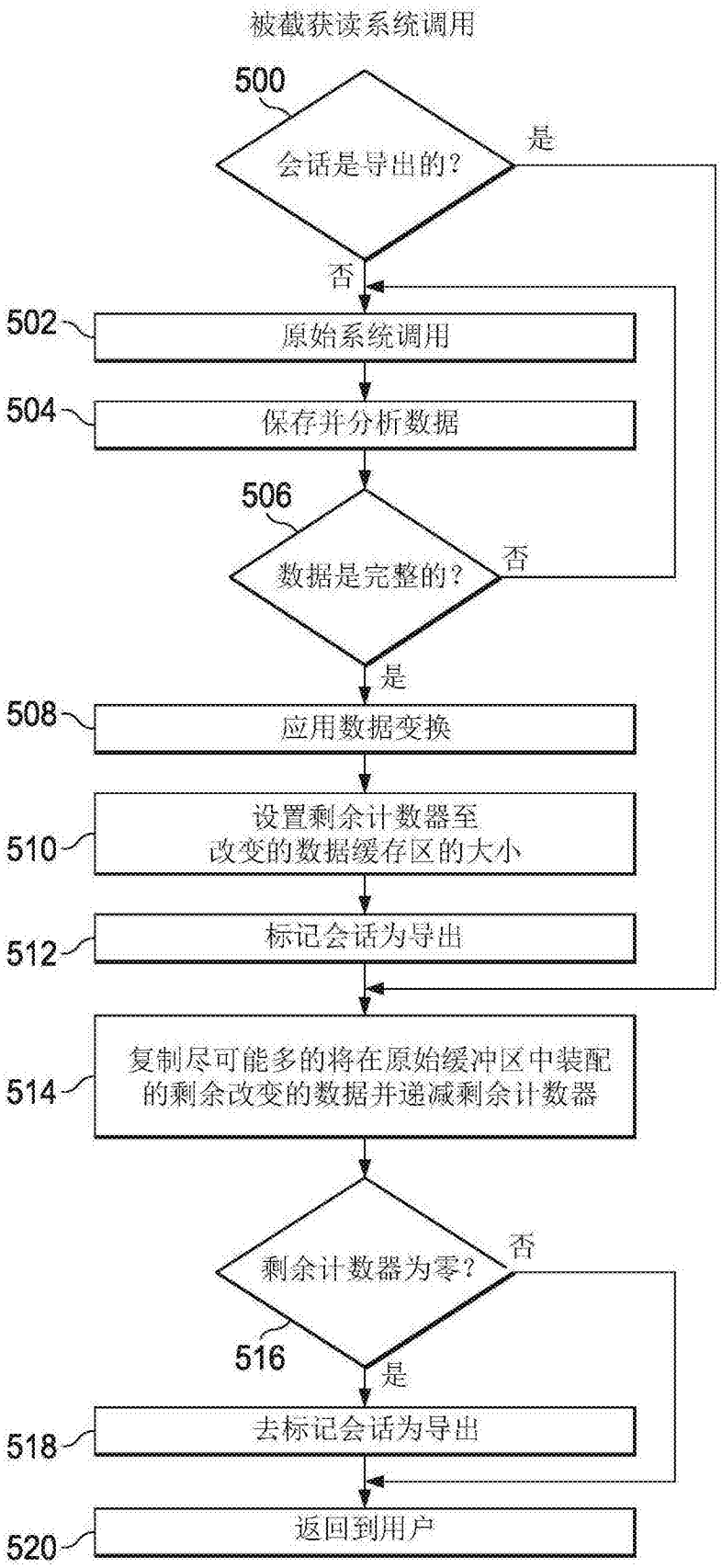

被截获读系统调用
500
会话是导出的?
是
否
502
原始系统调用
504
保存并分析数据
506
数据是完整的?
否
是
508
应用数据变换
510
设置剩余计数器至
改变的数据缓存区的大小
512
标记会话为导出
514
复制尽可能多的将在原始缓冲区中装配
的剩余改变的数据并递减剩余计数器
剩余计数器为零?
否
516
是
518
去标记会话为导出
520
返回到用户

图 5

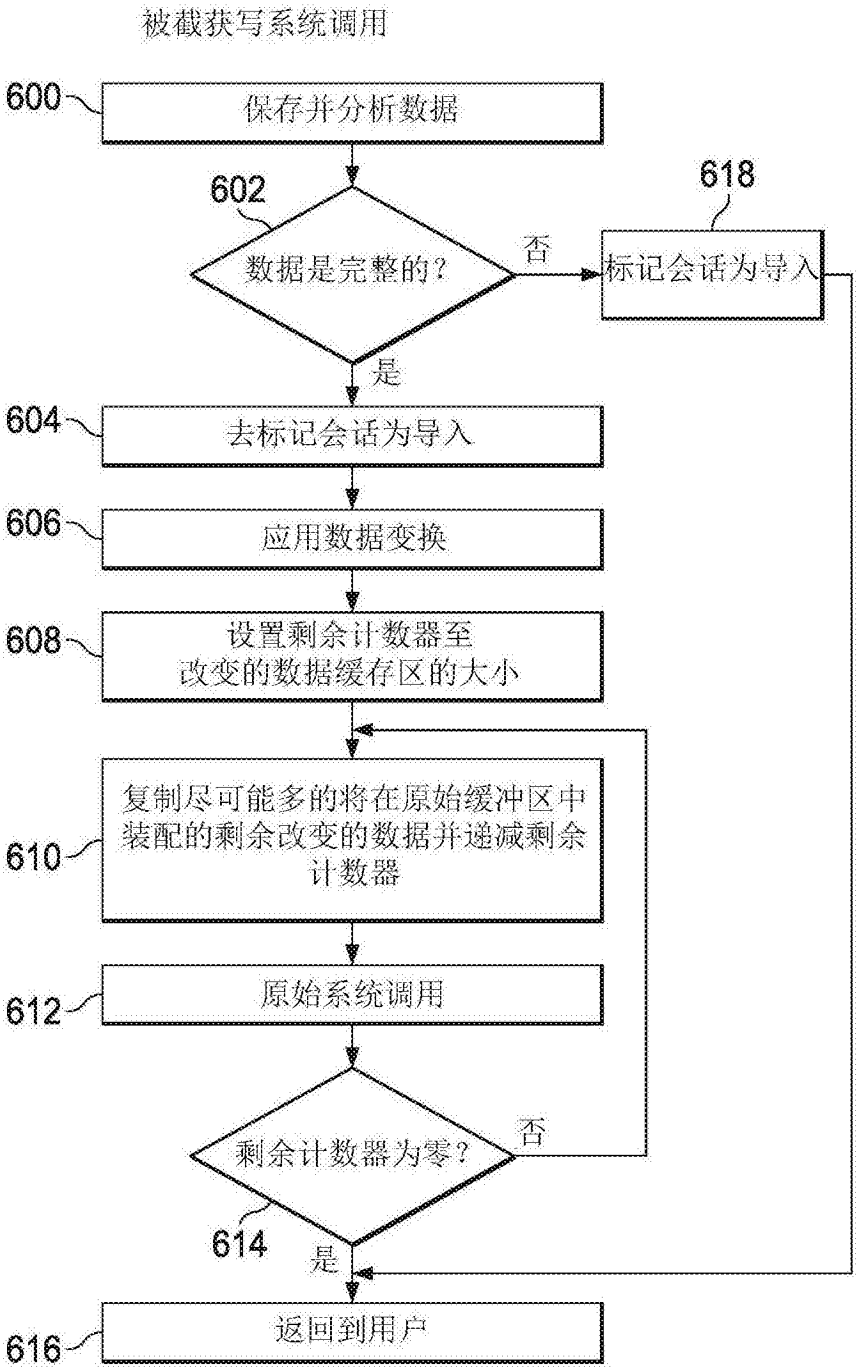

被截获写系统调用
600
保存并分析数据
602
数据是完整的？
否
618
标记会话为导入
是
604
去标记会话为导入
606
应用数据变换
608
设置剩余计数器至
改变的数据缓存区的大小
610
复制尽可能多的将在原始缓冲区中
装配的剩余改变的数据并递减剩余
计数器
612
原始系统调用
剩余计数器为零？
否
614
是
616
返回到用户

图 6