[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/445 (2006.01)
H04L 12/00 (2006.01)

[12] 发明专利说明书

专利号 ZL 200410058924.3

[45] 授权公告日 2008 年 12 月 31 日

[11] 授权公告号 CN 100447740C

[22] 申请日 2004.7.21
[21] 申请号 200410058924.3
[30] 优先权
[32] 2003.7.21 [33] US [31] 10/633,375
[73] 专利权人 微软公司
地址 美国华盛顿州
[72] 发明人 A·V·佩特罗夫 M·H·申德
M·V·斯里格 T·麦克古里
[56] 参考文献
US6216175B1 2001.4.10
. .

Cluster – Based Delta Compression of a Collection of Files. Zan Ouyang, Nasir Memon, Torsten Suel, DimitreTrendafilov. PROCEEDING OF THE 3RD INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS ENGINEERING. 2002
审查员 王 可
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 谢喜堂

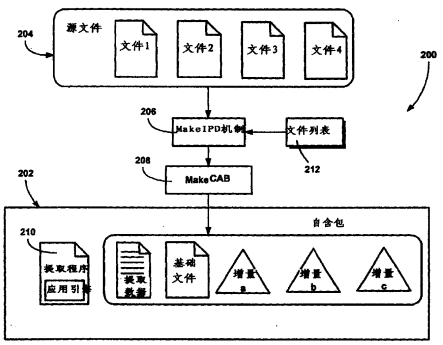
权利要求书 2 页 说明书 19 页 附图 7 页

[54] 发明名称

数据的包内增量压缩的系统和方法

[57] 摘要

一种用于以自含包提供文件数据，如一组用于更新计算机系统的文件的系统和方法，其中，通过增量压缩显著地减小了包的大小。 构建机制检查要分发的文件，并生成包含文件和增量的自含包。 为此目的，从各种基础文件大小和增量文件可能性构建有向图，并且最小生成树计算选择导致最小包的文件。 可以向基础文件应用多个增量来合成多个文件，并且任一基础文件其自身可以是先前从另一基础文件和增量合成的。 与包一起可任选地提供的客户提取机制如清单所指引地与包的内容一起工作，以从基础文件和所包含的增量合成目标文件。



1．在计算环境中的一种方法，其特征在于，它包括：

接收与多个源文件相对应的信息；

选择第一源文件作为基础文件；

从所述第一源文件和第二源文件生成一增量；

基于选择第一源文件作为基础文件并且基于生成一增量，生成一清单文件，所述清单文件包括执行多个源文件的提取所需的指令；以及

将所述基础文件、所述增量和所述清单文件封装为自含包。

2．如权利要求1所述的方法，其特征在于，它还包括，封装数据以指引客户提取程序从所述基础文件和所述增量合成与所述第二源文件相应的目标文件。

3．如权利要求1所述的方法，其特征在于，它还包括，设置至少一个文件名，客户提取程序可通过它从所述基础文件和所述增量合成与所述第二源文件相对应的目标文件。

4．如权利要求1所述的方法，其特征在于，所述第一源文件和所述第二源文件不是同一文件的不同版本。

5．如权利要求1所述的方法，其特征在于，所述第一源文件和所述第二源文件不是同一文件的不同语言翻译。

6．如权利要求1所述的方法，其特征在于，所述第一源文件和所述第二源文件是同一文件的不同语言翻译。

7．如权利要求1所述的方法，其特征在于，选择第一源文件作为基础文件包括基于对包大小的考虑选择所述源文件。

8．如权利要求7所述的方法，其特征在于，它还包括基于多个可能的源文件配对来构造文件大小的有向图，并基于所述有向图中的信息选择所述第一源文件。

9．如权利要求8所述的方法，其特征在于，选择第一源文件作为基础文件包括向所述有向图应用最小生成树或类似的算法。

10．如权利要求1所述的方法，其特征在于，选择第一源文件作为基础文件包括计算可能的增量的尺寸并基于所述尺寸选择第一源文件。

11．如权利要求1所述的方法，其特征在于，它还包括，向接收方提供所述包，所述接收方向所述第一源文件应用所述增量来合成所述第二源文件。

12．在计算环境中的一种方法，其特征在于，它包括：

接收包括至少一个基础文件和多个增量的包；

生成一包括执行多个源文件的提取所需的指令在内的清单文件；以及

向基础文件应用包内的增量并且基于清单文件来合成目标文件。

13．如权利要求 12 所述的方法，其特征在于，向基础文件应用增量包括向包括在所述包内的基础文件应用所述增量。

14．如权利要求 12 所述的方法，其特征在于，向基础文件应用增量包括向从另一增量和另一基础文件合成的基础文件应用所述增量。

15．如权利要求 12 所述的方法，其特征在于，它还包括解释数据文件来确定每一增量应用到哪一基础文件。

16．如权利要求 13 所述的方法，其特征在于，所述数据文件包括一组指令，包括标识向其应用特定增量文件的特定基础文件的指令。

17．如权利要求 12 所述的方法，其特征在于，它还包括，执行设置程序。

18．如权利要求 17 所述的方法，其特征在于，在每一增量应用到相应的基础文件之后执行所述设置程序。

19．如权利要求 12 所述的方法，其特征在于，它还包括，从临时目录中删除所述增量。

20．如权利要求 12 所述的方法，其特征在于，它还包括，向所述合成的目标文件应用另一增量来合成另一目标文件。

21．如权利要求 12 所述的方法，其特征在于，它还包括，向一公用基础文件应用至少两个增量来合成至少两个目标文件。

22．在计算环境中的一种系统，其特征在于，它包括：

用于选择第一源文件作为基础文件的装置，通过应用增量可以从所述基础文件衍生第二源文件；

基于选择第一源文件作为基础文件并且基于增量，生成一清单文件，所述清单文件包括执行多个源文件的提取所需的指令；以及

用于将所述基础文件、所述增量和所述清单文件封装为自含包的装置。

数据的包内增量压缩的系统和方法

技术领域

本发明一般涉及计算机系统，尤其涉及封装的计算机文件。

背景技术

当软件销售商想要向其顾客提供一组一个或多个文件时，如新产品发布或相对较大的升级，可能将一个或多个文件合并为一个档案，形成相关内容的单个包，其中，包一般是作为一组来使用的数据文件的某一集合。通过添加可执行代码，档案经常被制成自提取档案，当执行该可执行代码时，将包的内容提取回到先前所合并的文件组。通常通过执行刚提取的文件之一，自提取代码也可以启动设置程序，从而将文件复制到顾客的计算机上的适当的位置中。完成设置程序之后，自提取代码删除提取的文件，然后终止。在大多数情况下，这允许整个产品特点或更新被检索作为单个文件对象检索，可被直接执行来访问或安装产品的内容。

归档进程通常使用某种数据压缩来减小档案的大小，从而减小了分发和检索的成本，尤其是对大档案而言。一个这类压缩技术个别地压缩文件，向顾客提供所需要的任一个别文件的访问。这类包的大小一般是每一所包含的文件的经压缩大小的总和，加上提取代码的大小。在执行时，包将每一压缩的文件提取到一个临时位置，用户可以从该位置将每一文件复制到系统目录中的适当位置。

对于不需要个别文件访问的包，如当自动运行设置过程来安装所提取的文件时，通过使用机箱（cabinet）（或 CAB）文件可以进一步提高包压缩，在机箱文件中，在压缩之前本质上将一个文件加到另一个之后（连接的）。这使用基于 LZ 的编码器（由完成始发的工作的 Lempel 和 Ziv 命名的众所周知的字典编码器类型）提高了编码效率，因为采用 LZ 编码，输入数据流的压缩取决于输入数据流的以前的部分（称为历史），并且文件的连接增加了可用的历史数据量。注意，使用压缩文件，在提取过程中解压所压缩的数据，使在运行设置程序以在文件上操作之前文件处于其原始的形式。

即使采用压缩技术，例如，相对于可以方便地通过网络传输的数据量而言，

包仍较大。对于不具有宽带网络接入的顾客，包的大尺寸令其变得不实用，或至少下载这类包变得很不方便。一些顾客必须支付长距离或连接时间费用来下载数据，其它顾客可能具有能下载的数据量的定额和/或会话的连接时间的限制。其它顾客仅仅不因通过调制解调器下载大文件而感到烦扰。大文件下载对终止会话的网络连接问题还是易受攻击的。对于这类顾客，大的包的分发是一个问题。

包的销售商也具有关于他们所提供的下载的大小的成本。例如，分发大文件需要相当大量的网络服务器设备，这是昂贵的。经常由销售商付钱来为一些顾客制作可用的 CD-ROM。即使通过因特网分发也具有可变的成本，当发送大的包时该成本增加。

一种减少需要发送的数据量的提供更新的改进的方法在美国专利号 6,493,871 中有描述。在该方法中，客户（顾客）计算机首先从设置服务器获取包括设置程序和安装该软件产品所需要的文件的列表的初始设置包。客户计算机上的设置程序然后确定安装所需要的文件的当前或较早的版本是否已在客户计算机上存在，并编译更新客户计算机所需要的文件的请求列表。客户计算机向下载服务器发送该请求列表，下载服务器维护更新文件和补丁的集合，并通过向客户发送更新所需要的一组合适的文件来响应该请求列表。一个或多个文件可能以补丁的形式，其中，补丁是从文件的较早版本和该文件的较新版本衍生的小数据文件。补丁可以应用到客户计算机上已存在的较早文件版本的副本来产生新版本，消除了下载完整新版本的需要。

尽管这一数据压缩能够显著地减少用户必须下载的数据量，然而这一技术同样具有很多缺点。一个缺点是，这类二进制修补，也称为增量压缩，仅当销售商知道(或可以安全地假定)文件的哪一表示已在给定的客户计算机上可用时才起作用。这并不总是可能的，如采用 CD-ROM 或其它固定的分发模式时。注意，可以通过在每一不同版本的档案中包括多个文件来使单个一般档案更新销售商的顾客可能正在使用的文件的不同版本，其中一个版本可以应用到特定客户可能具有的文件的任一给定版本。然而，这还不是有效的，并且在有大量文件（如，类似于上百甚至上千）需要通过一个包来更新的情况下不实用或不易管理。由于必须处理大量文件的多个版本，通过增量压缩而达到的节省的大多数都被丢失。

总之，常规压缩成本较高和/或对许多用户和销售商来说不适当，因为所得的压缩包的大小还是太大，无法容易地分发。同时，增量压缩迄今不能对需要或想要

使用不需要对每一顾客在服务器进行动态定制的自含包（self-contained package）的顾客和/销售商较好地起作用。需要一种方法来提供高度有效的软件产品数据，而同时在包内是充分自含的。

发明内容

简言之，本发明提供了一种以自含包提供数据的系统和方法，其中通过增量压缩显著地减少了数据量。为此目的，以普通的（压缩）形式封装一组文件（可以是任一组相关数据），而第二组被表示为从第一或第二组的输出衍生的增量。由此，封装的包包含应用到同一包之内的或先前从同一包衍生出的其它文件的增量的集合。

本发明的包内增量系统和方法包括两个初级机制，一个构建机制和一个客户机制。一般而言，构建机制检查要分发的一组文件（目标文件）来生成优化的自含包内增量包，而客户组件与包的内容一起从所包含的增量合成目标文件。

销售商方的包内增量机制充分利用了包内文件之间的相似性来减小总的包大小。这对更新来说尤其能较好地起作用，因为更新通常具有一个以上的二进制文件，并且这些二进制文件由于具有一些共享的公用源代码或库而频繁地相互关联。更新也经常为不同的情形提供等效文件，如用于不同语言的各种等效文件。

在一个实现中，排列自含增量压缩包，使大多数文件从一个基础文件和一个增量合成，单个基础文件能够具有向其应用的多个增量来合成多个文件，并且/或者其中，任一基础文件其自身可能是先前从另一基础文件和增量合成而来。由此，通过包内增量机制的包构建可能具有其自身可能以某一方式压缩的单个基础文件，加上能够变换该基础文件的副本来合成其它文件的任意数量的增量。增量也能够应用到先前合成输出的文件，允许对每一目标文件的最优源选择。如果可以对已在接收端可用的文件作出任一假定，可以将现有文件的副本或应用到副本的增量添加到目标文件组来重新创建完整的包，进一步减小了包的大小。

在一个实现中，包内增量机制自动生成包内增量压缩包的增量。为此目的，给定需要通过包提供的目标文件的列表，则该机制探查合成每一文件的各种可能性，创建可能的增量，并检查所得的文件大小来确定哪一基础文件和增量将得到最小的包大小，而在客户端随后的提取时能完全重新创建目标文件。例如，执行提取（如以适当的顺序）的所需要的指令被保存在清单文件中，而最终化封装所需要的

信息被保存在伪指令文件中，从而能够生成包括基础文件、增量和清单文件的机箱文件以及任一其它所需要的文件，如客户将运行来执行提取的可执行提取工具。

在客户端，包内增量自提取程序框架包括可执行提取工具，它创建临时目录，与常规包一样扩充包的内容，但是在开始设置程序之前，依照本发明的一个方面执行额外的处理。为此目的，提取工具解释清单文件来应用增量以从基础文件或曾在包内的文件或从其本身先前被合成的基础文件合成（一些）目标文件。提取工具可以在开始设置程序之前丢弃增量文件，使设置程序仅看见解压和/或合成的目标文件的完整的组。可以容易地理解，将包内的一些文件作为（压缩的）基础文件，其它携带的文件作为基于压缩的基础文件的增量，能够显著地减小包的大小。注意，使用包内增量的包的形式和应用从现有的组件的观点来看与常规自含包相同，即使顾客仅提取包的内容而不开始设置进程。这是由于包内增量内容的增量处理由自提取可执行代码透明地执行。

结合附图阅读以下详细描述，可以清楚其它优点，附图中：

附图说明

图 1 是一般表示可以结合本发明的计算机系统的结构图；

图 2 是一般表示依照本发明的一个方面使用包内增量压缩生成自含包的结构图；

图 3 是一般表示依照本发明的一个方面使用包内增量压缩生成自含包的包生产源环境（如软件销售商）处的组件的结构图；

图 4 是一般表示依照本发明的一个方面如何选择包括在包内增量压缩包中的文件和/或增量的结构图；

图 5 是一般表示依照本发明的一个方面从包内增量压缩包提取文件的顾客目标环境（如客户计算机）中的组件的结构图；

图 6 是一般表示依照本发明的一个方面通过向单个基础文件应用多个增量来提取多个文件的结构图；

图 7 是一般表示依照本发明的一个方面通过向基础文件应用增量并通过向其自身为通过增量解压合成的基础文件应用增量来提取多个文件的结构图。

具体实施方式

示例性操作环境

图 1 说明了适合在其中实现本发明的计算系统环境 100 的一个示例。计算系统环境 100 仅为合适的计算环境的一个示例，并非试图提出对本发明的使用或功能的范围的限制。也不应将计算环境 100 解释为关于示例性操作环境 100 中说明的任一组件或其组合具有任何依赖或需求。

本发明可以使用众多其它通用或专用计算系统环境或配置来操作。适合使用本发明的众所周知的计算系统、环境和/或配置包括但不限于：个人计算机、服务器计算机、手持式或膝上设备、输入板设备、多处理器系统、基于微处理器的系统、机顶盒、视频游戏、蜂窝或其它电话产品、可编程顾客电子设备、网络 PC、小型机、大型机、包括任一上述系统或设备的分布式计算环境等等。

本发明可在计算机可执行指令的一般语境下描述，计算机可执行指令如程序模块，由个人计算机执行。一般而言，程序模块包括例程、程序、对象、组件、数据结构等等，执行特定的任务或实现特定的抽象数据类型。本发明也可以在分布式计算环境中实践，其中，任务由通过通信网络连接的远程处理设备来执行。在分布式计算环境中，程序模块可以位于本地和/或远程计算机存储媒质中，包括存储器存储设备。

参考图 1，用于实现本发明的示例系统包括以计算机 110 形式的通用计算设备。计算机 110 的组件可包括但不限于，处理单元 120、系统存储器 130 以及将各类系统组件包括系统存储器耦合至处理单元 120 的系统总线 121。系统总线 121 可以是若干种总线结构类型的任一种，包括存储器总线或存储器控制器、外围总线以及使用各类总线结构的本地总线。作为示例而非局限，这类结构包括工业标准体系结构（ISA）总线、微通道体系结构（MCA）总线、增强 ISA（EISA）总线、视频电子标准协会（VESA）本地总线以及外围部件互连（PCI）总线，也称为 Mezzanine 总线。

计算机 110 通常包括各种计算机可读媒质。计算机可读媒质可以是可由计算机 110 访问的任一可用媒质，包括易失和非易失媒质、可移动和不可移动媒质。作为示例而非局限，计算机可读媒质包括计算机存储媒质和通信媒质。计算机存储媒质包括以用于储存信息的任一方法或技术实现的易失和非易失，可移动和不可移动媒质，信息如计算机可读指令、数据结构、程序模块或其它数据。计算机存储媒质包括但不限于，RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、数

字多功能盘（DVD）或其它光盘存储、磁盒、磁带、磁盘存储或其它磁存储设备、或可以用来储存所期望的信息并可由计算机 110 访问的任一其它媒质。通信媒质通常在诸如载波或其它传输机制的已调制数据信号中包含计算机可读指令、数据结构、程序模块或其它数据，并包括任一信息传送媒质。术语“已调制数据信号”指以对信号中的信息进行编码的方式设置或改变其一个或多个特征的信号。作为示例而非局限，通信媒质包括有线媒质，如有线网络或直接连线连接，以及无线媒质，如声学、RF、红外和其它无线媒质。上述任一的组合也应当包括在计算机可读媒质的范围之内。

系统存储器 130 包括以易失和/或非易失存储器形式的计算机存储媒质，如只读存储器（ROM）131 和随机存取存储器（RAM）132。基本输入/输出系统 133（BIOS）包括如在启动时帮助在计算机 110 内的元件之间传输信息的基础例程，通常储存在 ROM 131 中。RAM 132 通常包含处理单元 120 立即可访问或者当前正在操作的数据和/或程序模块。作为示例而非局限，图 1 说明了操作系统 134、应用程序 135、其它程序模块 136 和程序数据 137。

计算机 110 也可包括其它可移动/不可移动、易失/非易失计算机存储媒质。仅作示例，图 1 说明了对不可移动、非易失磁媒质进行读写的硬盘驱动器 141、对可移动、非易失磁盘 152 进行读写的磁盘驱动器 151 以及对可移动、非易失光盘 156，如 CD ROM 或其它光媒质进行读写的光盘驱动器 155。可以在示例性操作环境中使用的其它可移动/不可移动、易失/非易失计算机存储媒质包括但不限于，磁带盒、闪存卡、数字多功能盘、数字视频带、固态 RAM、固态 ROM 等等。硬盘驱动器 141 通常通过不可移动存储器接口，如接口 140 连接到系统总线 121，磁盘驱动器 151 和光盘驱动器 155 通常通过可移动存储器接口，如接口 150 连接到系统总线 121。

图 1 讨论并说明的驱动器及其关联的计算机存储媒质为计算机 110 提供了计算机可读指令、数据结构、程序模块和其它数据的存储。例如，在图 1 中，说明硬盘驱动器 141 储存操作系统 144、应用程序 145、其它程序模块 146 和程序数据 147。注意，这些组件可以与操作系统 134、应用程序 135、其它程序模块 136 和程序数据 137 相同，也可以与它们不同。这里对操作系统 144、应用程序 145、其它程序模块 146 和程序数据 147 给予不同的序号来说明至少它们是不同的副本。用户可以通过输入设备，如输入板或电子数字化仪 164、麦克风 163、键盘 162 和指向设备

161（通常指鼠标、轨迹球或触摸板）向计算机 110 输入命令和信息。图 1 未示出的其它输入设备可包括操纵杆、游戏垫、圆盘式卫星天线、扫描仪等等。这些和其它输入设备通常通过耦合至系统总线的用户输入接口 160 连接至处理单元 120，但是也可以通过其它接口和总线结构连接，如并行端口、游戏端口或通用串行总线（USB）。监视器 191 或其它类型的显示设备也通过接口，如视频接口 190 连接至系统总线 121。监视器 191 也可以与触摸屏面板或其类似物组合。注意，监视器和/或触摸屏面板可以物理地耦合至结合计算设备 110 的外壳，如平板类型的个人计算机。另外，诸如计算设备 110 的计算机也包括其它外围输出设备，如扬声器 197 和打印机 196，通过输出外围接口 194 或其类似物连接。

计算机 110 可以在使用到一个或多个远程计算机，如远程计算机 180 的逻辑连接的网络化环境中操作。远程计算机 180 可以是个人计算机、服务器、路由器、网络 PC、对等设备或其它公用网络节点，并通常包括许多或所有上述与计算机 110 相关的元件，尽管在图 1 中仅示出了存储器存储设备 181。图 1 描述的逻辑连接包括局域网（LAN）171 和广域网（WAN）173，但是也可以包括其它网络。这类网络环境常见于办公室、企业范围计算机网络、内联网以及因特网。当在 LAN 网络环境中使用时，计算机 110 通过网络接口或适配器 170 连接至 LAN 171。当在 WAN 网络环境中使用时，计算机 110 通常包括调制解调器 172 或其它装置，用于通过 WAN 173，如因特网建立通信。调制解调器 172 可以是内置或外置的，通过用户输入接口 160 或其它合适的机制连接至系统总线 121。在网络化环境中，描述的与计算机 110 相关的程序模块或其部分可储存在远程存储器存储设备中。作为示例而非局限，图 1 示出了远程应用程序 185 驻留在存储器设备 181 上。可以理解，示出的网络连接是示例性的，也可以使用在计算机之间建立通信链路的其它装置。

包内增量压缩

本发明部分地一般针对一种方法和系统，提供文件和增量压缩文件（以下称为增量）的自含包，当提取时，产生安装器或其类似物更新计算机系统所需的文件。由此，这里的许多示例将一般针对提供更新包。然而，可以理解，除更新以外，还有许多对这类产品的使用。例如，依照本发明的一个方面，可以将诸如一套软件应用的全新安装提供为文件和增量压缩文件的自含包。其它数据文件同样可以从本发明的系统和方法中获益，尽管本发明往往在减小包含诸如随情形变化而变化的许多

可执行文件和/很大程度上等效的文件的包的大小方面起到很好的作用。此外，如这里所使用的，术语“文件”或“多个文件”可包括常规地被认为是文件的东西，但是实际上也可包括数据的任一集合，如不必要地安排为常规文件系统文件的股票指数、字节流等等。

此外，尽管这里描述的包指自含的，然而可以容易地理解，包不需要完全自含来从本发明获益。本发明可以与增量的常规使用相组合。可以构造包含将现有文件用作基础文件的增量的混合物。从这一增量合成的文件然后可以用作包内另一增量的基础。例如，可以知道用户在给定计算机上具有的哪些文件，如读取包的内容的自提取程序，可能是通常在用户的计算机系统中存在的操作系统组件，从而该程序不需要作为包的一部分包括在内。同样，可能已知一个给定文件版本在顾客的计算机中存在，如，如果更新是针对当前只能为一个版本的文件。基于这一知识，包生产过程有时候可以避免必须包括特定的数据，从而进一步减小包的大小。

一般而言，如图 2 所示，本发明的一个方面在包生产环境 200 中操作，以试图将包需要包含的数据的大小最小化（至少至一个合理的程度）的方式构造包括文件和增量的自含包内增量压缩包 202。为此目的，提供了一组源文件 204，可包括包想要包括的新文件版本，（但是可能包含生成增量所需要的任一较旧文件版本）。这些源文件 204 一般与客户需要具备的目标文件相应。一般而言，如下所述，依照本发明的一个方面，制作包内增量（IPD）机制 206 首先将源文件 204 处理（如，通过读取其列表）为基础文件和/或增量。然后，MakeCAB（机箱文件）机制，可以是常规进程，将基础文件和/或增量数据压缩为自含包 202。也可以使用其它压缩技术。

注意，当前技术需要文件的较旧版本来生成增量。例如，用于增量压缩的已知技术将原始文件（或其一些较后版本）与新版本一起输入到生成增量文件的增量创建引擎中；随后在客户端将该增量应用到原始版本来重新创建新版本。这些技术以及能获得更好压缩的改进在名为“将比较和压缩执行为单个处理的文件更新（File update performing comparison and compression as single process）”的美国专利号 6,496,974；名为“下载软件安装的更新的方法和系统（Method and system for downloading updates for software installation）”的美国专利号 6,493,871；名为“对补丁生成和压缩预处理参考数据流（Preprocessing a reference data stream for patch generation and compression）”的美国专利号 6,466,999；名为“通过使用不同于解

压辅助数据预初始化压缩器/解压器的文件更新（File update by pre-initializing compressor/decompressor with other than decompression aid data）”的美国专利号6,449,764；以及名为“使用较小补丁文件更新软件的方法和系统（Method and system for updating software with smaller patch files）”的美国专利号6,243,766；以及名为“使用在标准化相应原始安装过程中创建的副本之间的差异之后的同一更新数据来升级原始文件的副本的方法（Method for upgrading copies of and original file with same update data after normalizing differences between copies created during respective original installations）”的美国专利号6,216,175中有描述。

依照本发明的一个方面，并如下所述，一般而言，不需要对用来生成增量的基础文件的身份作出假设，其中，如这里所使用的，基础文件是随后向其应用增量来产生另一文件的任一文件。由此，与当前技术不同，任一文件，不仅是同一文件的较早版本，可以用作基础文件，通过应用增量可以从该基础文件合成另一文件。例如，取代从先前的文件版本和新文件版本生成增量，可以从对人类观察者来说看似为本质上不相关的文件生成增量。例如，结果可能是，对于给定的一组文件，可以将一个电子表格组件文件用作对增量的基础文件，当应用时，能够合成字处理组件文件。此外，可以用多个增量再次使用单个基础文件来合成多个合成文件。

因此，在一个实现中，源文件204不需要包括任一较旧文件版本，因为本发明的系统和方法可以使用仅从（一个或多个）较新文件版本衍生的增量。例如，在图2中，文件1可以是作为包204的一部分作为更新包括在内的新文件版本，并且可能用来从文件2生成增量a，从文件3生成增量b以及从文件4生成增量c。然后，当随后通过向文件1应用这些增量在客户顾客上提取时（如后文参考图6所描述的），四个目标文件，文件1－文件4在客户机器上可由设置程序或其类似物使用。注意，可以令一个包包含一个或多个用来合成其它文件的基础文件，然后丢弃这些文件，并且这些文件实际上不是最终文件组的一部分，并且例如，不被设置程序所使用，如后文所描述的。

依照本发明的另一方面，并且也参考图7在后文描述的，向其应用增量来合成文件的基础文件其自身可以是从先前的增量解压操作合成的。由此，例如，在图7中，文件2可以从文件1和增量a创建。然后，文件2用作向其应用增量b来产生文件3的基础文件。由此，本发明提供了许多新的概念，每一概念运作来提供包内所需要的数据量的减小。

此外，可以将文件版本用作用于生成另一文件版本的增量的基础文件，并在包内包括该基础文件用于如需要时的提取目的，即使该基础文件将会被删除并且不被设置程序所使用。例如，在图 7 中，文件 1 可以是用来创建增量的较旧文件版本，置于包内，并由提取程序 210 用来创建其它文件，但是然后在设置程序之前被删除。

转向附图的图 3，示出了包生产环境 200（图 2）的一个实现中的组件。通常这一环境 200 通过一组一个或多个软件销售商的计算机或与需要生产包的销售商关联的某一第三方来实现。注意，这一生产环境 200 通常不是响应于客户请求而动态操作的，而是可能花费类似几个小时或几天的相对较长的时间来生产自含包 202 的计算上昂贵的进程。

一般而言，MakeIPD 机制（制作 IPD）206 读取提供的文件列表 212 来确定哪些文件加入包内，以及哪里可以找到这些文件。例如，文件列表可包含[Files]段，其每一条目指定了包内的文件的名字，以及到该文件的全路径。文件列表也可以指定已知在用户计算机上存在的特定参考文件，这些文件不需要在当前的包内，但是可以用来构造上述的混合物。这些文件可以被认为是包内任何其它文件的潜在基础。

文件列表 212 也可以在如[Options]段中指定的一些处理选项。例如，可以提供“Run（运行）”伪指令（如作为经过清单文件的传递）来指定在提取以后应当执行包内的哪一文件（如果有的话）。可以设置“Verify（核实）”伪指令来引发 MakeIPD 机制对所有包文件生成[Verify]段。“PatchDLL”伪指令标识传递到清单文件的文件，同时引发 MakeIPD 机制 206 在创建机箱文件时使用的脚本中包括该文件，并且认为该文件是包内任何其它文件的潜在基础。

如图 3 所示，在一个实现中，总体生产进程从 GUI 或命令行 304 启动，GUI 或命令行由分析程序 306 解释来确定操作参数。操作参数可包括标识文件列表 212，如上所述，它标识如包含包内所需要的文件的名字的文件名的文本文件列表中要使用的源文件 308。也可以提供路径信息来指定可能包含如下所述的所需要的符号文件的一个或多个目录（如，由分号隔开）。如果未指定，使用的目录为源文件的目录。

同样，可以指定另一操作参数－目录，用于使用在包构造过程中使用的各种中间文件。更特别地，MakeIPD 机制 206 在处理过程中创建许多中间文件，并在指定的工作目录中维护。这类文件包括符号列表、增量文件和清单文件，将解法描

述为一组增量解压指令。注意，如果随后运行 MakeIPD 机制 206 并且指定了同一工作目录，可以使用任何现存的文件来协助分析。例如，如果所有需要的文件仍可用，则 MakeIPD 机制 206 将在如几秒内迅速完成其操作。如果自从先前的构建以来仅一些包文件改变，重新使用未受影响的增量将节省可观的处理时间。

另一操作可指定创建如 CAB 压缩形式的最终包所使用的脚本的名字。其它操作参数可指定现有清单的位置，如下所述，可以导入该清单以从一些先前的运行复制解法。用户也可以指定处理过程中的不同类型的输出（如向用户或向文本文件），如，允许选择简明或详细输出，或禁止输出。

分析文件列表，其中，每一条目提供包内源文件的位置以及该源文件的名字。分析任一输入文件选项，可包括提取之后执行的文件的名字、用作提取程序的增量应用引擎（如 DLL）的名字以及自提取机制是否应当在提取之后核实文件的签名（如 MD5 散列生成），（其中，MD5 指 RSA 数据安全公司的消息摘要算法 5，也称为因特网 RFC 1321。可以使用任一合适的误差检测或完整性核实散列，包括 CRC、MD5、SHA1 等等）。

作为预处理的另一部分，如下所述，向文件列表添加用于清单文件的条目，以及用于增量应用引擎的条目。同样，对源文件 308 计算 MD5 签名，来将任何重复标识为原始发生的副本。如果该文件是可执行文件，则对其符号文件提取细节。

更具体地，上述美国专利号 6,466,999 描述了当使用增量时，对可执行文件（如 EXE、DLL、OCX、SYS 等等）使用符号来获得更优化的尺寸减小。MakeIPD 机制 206 调节了这一技术，并在提供源文件的同一目录中查找符号。可以使用选项来明确地提供额外的符号目录，每一目录由分号隔开。递归地搜索每一指定目录来查找对分析有益的任何符号。扫描符号路径，查找任何标识的符号文件。

对于每一单独的源文件，MakeIPD 机制 206 生成预期增量输入的列表，包括包内每一其它但独源文件的条目。这些列表将总共具有 $N\cdot(N-1)$或接近 N^2 个条目。

依照本发明的一个方面，MakeIPD 机制 206 的迭代程序组件 312 通过将文件和该文件的列表上的每一文件输入到增量创建引擎 314，为每一文件列表上的每一预期增量创建增量。如果用于创建增量的输入都为可执行文件，则使用任一可用的符号信息来优化增量的大小，如上述美国专利号 6,466,999 中所描述的；（注意，一般而言，合适的增量创建引擎 314 在上述美国专利中描述）。这些增量 315 储存在工作目录中，并且每一增量的大小被添加到该列表条目，用于下一计算。

依照本发明的另一方面，从这一大小以及文件身份信息生成有向图316。更具体地，将每一源文件作为顶点添加到有向图 316，并将每一预期增量作为边添加，其加权等于该增量的大小。所包括的还有从“空”顶点到每一源文件顶点的边，其加权等于该源文件的压缩大小。

作为示例，如图 4 所示，考虑从在四个源文件 A、B、C 和 D 上的迭代生成的有向图 416。结合该信息的替代表格表示 417 能够看见，每一文件是一个顶点，每一预期增量是一个尺寸值，如，对使用文件 A 作为基础文件并使用文件 B 作为合成文件，增量的尺寸为 ba，对于尺寸 ab 反之亦然。尺寸 b 仅用于压缩 B，如，使用“空”顶点作为边。

从这一信息，可以在有向图 316 上计算最小生成树 320。为此目的，可以采用各种众所周知的最小生成树计算 318 之一，一些具有几乎线性的运行时间。这类最小生成树计算在计算机科学文献中有描述，并且仅在这里简要描述。

概念地而言，每一文件从最小可用增量衍生。然而，重要的是不形成环型引用，如文件不能被用来合成其自身。排除特定的增量来打破这些循环。全局最优化该进程，使能够没收其它增量来允许使用另一更有希望的增量，来减小总的大小。这一问题映射到称为“有向最小生成树”的问题。可以使用对边加权的增量的节省，并搜索有向最大生成树来构造另一等效解。

然后，使用“空”顶点作为根来枚举生成树。离开根顶点的边对应于在包内简单压缩的文件。离开其它顶点的边对应于使用生成的增量压缩的文件。继续图 4 的示例，可以看见，将最小生成树 420 枚举为结果的解，其中，简单压缩 A，压缩 B 并用作可以从其从合适的增量合成文件 D 的基础文件。进而，从合成的基础文件 D 以及用于从 D 生成 C 的合适的增量合成文件 C。可以理解，当以这一方式使用时，基于尺寸作为加权的最小生成树提供了最小包的可能性。

返回至图 3，枚举 322 本质上标记了每一源文件的解法（压缩或增量），并以在树中找出它们的顺序创建源文件的连接列表 324。从这一连接列表 324（以及上述其它文件信息），清单生成进程 326 如下所述地格式化清单文件 328。本质上，当在客户计算机上运行时，清单文件 332 指引提取程序 210 的操作。注意，操作以特定的顺序在清单文件 326 中列出，并以该顺序执行，来确保提取程序不会运行到一个文件需要被合成来作为用于应用增量的基础文件而尚未存在的情况。例如，在图 4 中，在文件 C 可以从文件 D 合成之前，文件 D 需要从文件 B 合成。

另外，MakeIPD 机制 206 包括伪指令文件生成组件 330，它生成伪指令文件 332，将从该文件创建压缩包。伪指令文件包含临时目录中源文件和增量的位置，以及包内这些文件的名字。压缩程序/封装程序 334 组件，如常规 MakeCAB（制作 CAB）机制 208（图 2），采用该伪指令文件来生成自含增量压缩包 202。参数选项指定要创建的伪指令文件的路径和名字，以及使用该伪指令文件指定同一路径内具有同一基础文件名的 CAB 文件。

尽管图 4 的简化示例仅使用了四个文件，MakeIPD 机制 206 能够花费相对较长的时间来分析包的内容。由此，当可能时，期望避免进程的迭代/分析部分。例如，当需要生成多个包时，每一个包概略地包含同一内容，可以使用先前运行的解法（如，在清单中维护）来更直接地指定当前包的解法。当在优化了第一个包之后为额外的语言生成包时可能出现这一情况。这一情况可用的另一时间是当由于内容中有小变化而重建包时。

为调节构造包的现有解法，MakeIPD 机制 206 能够导入解法。例如，如果指定为参数选项，该参数可以指定能够找到清单文件的目录，或能够指定要使用的文件的完整文件名。

一般而言，当使用时，MakeIPD 机制 206 的导入进程从某一先前创建的清单文件读取[Deltas]段，来看选择了哪些增量。更特别地，当导入解法时，读取导入的清单来生成预期增量输入的列表。引用该包内复制文件的任一条目作为替代被推断为原始文件。导入清单内标识复制的任一条目被推断为预期增量输入，反之亦然。这些列表一般在每一文件中仅有一个条目，或者说接近 N 个总条目。

可以容易地理解，当使用导入操作来指定解法文件时，大多数 MakeIPD 分析被绕过，因为仅使用先前选择的增量。注意，仅需要构造约 N 个增量，而不是 N^2 个。如果新的包包含任一新内容，则不考虑这些文件用于增量。如果原始包已选择了一个特定文件作为增量参考，并且未在新的包中找到该文件，将不考虑这些增量。然而，应当理解，导入操作对具有很类似内容的包能起到最好的作用。

转到图 5 所示的自提取进程的解释，在顾客环境 500，以某一方式，如通过网络传输或在诸如压缩或 DVD-ROM 盘的物理媒质中，接收自含包 502 的一个副本。注意，图 5 示出包生产者提供包，然而应当理解，可以有一个或多个中间物，如第三方批发商、令包对其它机器可用的企业网络等等。

一般而言，当以某一方式，无论通过 GUI、网络脚本、命令行等等，执行可

执行提取文件（如，在自含包中，但是可能已经在顾客机器上）时开始提取机制504。如需要，可以有指定的参数，并分析这些参数。

作为第一预处理操作，可执行程序可在本地硬盘，如具有最大空闲空间的硬盘上创建随机命名的临时目录506。另选地，可以如通过可任选的参数指定一个临时工作目录。

每一文件从机箱文件提取到临时目录506。作为这一提取的一部分，解压当创建机箱时所压缩的文件。注意，可能有在创建机箱文件之前已压缩的其它文件，并且在提取的第一部分中不解压这些文件。

此时，目录506包含一个或多个基础文件508、一个或多个增量510以及任何其它文件512，如当发现能够比使用增量压缩更有效地压缩时简单压缩的文件（如图4的示例中的文件A和B）。仍为压缩文件的任何文件可以在需要时被解压。本示例中，清单514、应用引擎516（如可能包括在包内的DLL）以及设置程序518（如果其自身不是合成的）也在目录506中可用。注意，可能不存在清单文件，这可能是文件的常规机箱压缩的情况，即，不使用包内增量压缩。

如在图5的示例中，在存在清单文件514时的情况，提取机制504处理清单514中列出的每一指令。对于其中所列出的每一增量510，将指定的增量510应用到指定的输入基础文件（如，基础文件508之一或先前合成的文件520）来创建新文件，在图5中由合成文件520表示。注意，应用引擎本质上是增量创建引擎的倒转，将基础文件作为一个输入，增量作为另一输入，并合成目标文件作为输出。与增量创建引擎类似，合适的应用引擎在上述美国专利申请中有描述。同样，如下所述，对于每一复制的文件，使用新名字作出指定文件的副本，而对于清单514中指定的每一删除模式，提取机制504删除临时目录506中匹配该模式的任何文件。这通常被用来在合成了所有文件之后丢弃增量。

作为示例，图6示出了通过具有单个基础文件的自含包所得的文件，对该基础文件应用三个增量a、b、c来合成三个文件，文件2、文件3和文件4。基础文件文件1可用作更新，尽管如上所述并非所需。一般而言，以标注的顺序跟随箭头，可以看到提取程序读取提取数据（如，清单），如需要，将基础文件解压到设置程序将用来安装这些文件的目录。然后结合每一增量a、b和c使用基础文件来分别合成文件文件2、文件3和文件4。

图7示出了一个稍微不同的示例，在该示例中，文件2从文件1和增量a合

成，然后用作基础文件，与增量 b 一起合成文件 3。文件 4 从该基础文件和增量 c 合成。

返回到图 5，一旦目标文件完全可用，如果通过 RUN 伪指令将程序标记为设置程序 518，则自动执行该程序。完成设置程序之后，提取机制检索并保存其返回代码。然后，提取机制 504 删除临时目录中创建的任何剩余文件，包括任何目录条目，并使用保存的设置程序返回代码退出。如需要，也可以删除提取机制 504。

清单文件信息

在一个实现中，包内增量特点由清单文件启用。然而，可以容易地理解，指引提取机制的其它方法也是可行的。例如，可以使用其它合适的提取数据。另选地，例如，可以以某一方式对增量排序（如，以在机箱解压时放置在临时目录中的顺序等等），并以该顺序将增量应用到基础文件，其文件名从增量衍生或通过一组重命名操作改变。然而，如下所述，清单文件提供了一种直接且有效的方法来指引提取机制。

在这一特定的实现中，在内嵌的机箱文件中该文件被命名为“_sfx_manifest_”。当将机箱提取到临时目标目录时，提取机制寻找这一文件名，并使用其内容来指定完成提取之后要执行的各种处理。_sfx_manifest_文件不添加到目标目录，以不干扰设置程序。在完成这一额外的处理之后，开始设置程序。

在一个实现中，清单是一文本文件，组织为由方括号指示的段，尽管可以容易地理解，诸如标记语言格式的其它格式是等效的。每一段包含表示操作细节的行。定义了若干段名，包括[Deltas]、[Copy]、[Verify]和[Delete]。每一段包含对段特定的条目，依照它们所属的段来解释。段中的某些条目可以以预定的关键字开始，后者称为伪指令。

可以以任一顺序指定清单中的段，但是以[Deltas]、[Copy]、[Verify]然后[Delete]的顺序处理。每一段内，顺序地处理条目。由此，通过名字而不是它们在文件中的位置来找出支持的段。

在这一格式中，文件中的每一段从由括号包围的段名开始，并在新段的开始或在文件的末端结束。如果多于一个段具有同一名字，它们被逻辑地合并为单个段。段名、条目和伪指令是对情况不敏感的。其它规则是，段中的每一条目和伪指令以新行字符（十六进制的 0x0A）结束或在文件的末端结束。注释以分号(;)字符开始，

并在新行字符或文件末端结束。注释可以在行中自己出现，或在行中位于条目之后。使用逗号来分隔段的条目和伪指令中提供的值，并且可能需要等号来将“关键”值与其它参数分隔。

文件名条目与包的根相关，尽管在替换实现中可以支持绝对路径。包内子目录中的文件用相对路径来表示，如，驻留在“update”子目录的名为“update.exe”的文件被引用为 update\update.exe。空行、行中的前导空白（空格、跳格等等）被忽略，好像引号外的空白。如果空格字符、逗号或等号是串的一部分，则该串在引号中。不在引号中的分号字符用于注释，其中，到该行的末尾的所有东西都被忽略。

需要包含空格或其它隔断字符的文件名或相对路径由双引号包含。不可以对段名加引号。以下是遵循上述规则的示例清单：

```
; Sample _sfx_manifest_ for Q326863
; This package contains eight closely-related files

[Options]
     run = xpsp1hfm.exe

[Deltas]
     sp2\ntkrnlpa.exe = sp2_ntkrnlpa_exe._p, sp2\ntoskrnl.exe
     sp2\ntkrpamp.exe = sp2_ntkrpamp_exe._p, sp2\ntkrnlpa.exe
     sp2\ntkrnlmp.exe = sp2_ntkrnlmp_exe._p, sp2\ntkrpamp.exe
     sp1\ntoskrnl.exe = sp1_ntoskrnl_exe._p, sp2\ntoskrnl.exe
     sp1\ntkrpamp.exe = sp1_ntkrpamp_exe._p, sp2\ntkrpamp.exe
     sp1\ntkrnlpa.exe = sp1_ntkrnlpa_exe._p, sp2\ntkrnlpa.exe
     sp1\ntkrnlmp.exe = sp1_ntkrnlmp_exe._p, sp2\ntkrnlmp.exe

[Delete]
     *._p

[Verify]
     sp1\ntkrnlmp.exe = 1D575A38471CB066CC23925AEFCD9A49
     sp1\ntkrnlpa.exe = 89A0875AEA13E021C9E63F2EB6446327
     sp1\ntkrpamp.exe = 934AAC402BA1F8D1C9319AA0DB849E6F
     sp1\ntoskrnl.exe = C78CA71C81A051DF25A79102C867BB10
     sp2\ntkrnlmp.exe = E62EA04019BC4AE785855DA0EE36D231
     sp2\ntkrnlpa.exe = 6C1BD8121224A83DC0FD9E36BFCF2AD9
     sp2\ntkrpamp.exe = 64F5029190445488347B204DF6A53A6C
     sp2\ntoskrnl.exe = A7379A2180D3AA4F64D804D8B5CDD659
```

[Deltas]段描述了包内增量的核心特点，即如何从包内的其它文件合成设置所需要的一些文件。[Deltas]段中的条目的语法包括：

[Deltas]

{目标文件名} = {增量文件名}[, {参考}]

其中，{目标文件名}是要生产的文件的名字，{增量文件名}是包内的增量文件的名字，{参考}是用作基础文件的现有文件的名字（如果有的话）。提取程序将增量文件应用到参考文件的一个副本，创建目标文件。

这一语法的一个示例包括：

[Deltas]

file2 = file1_to_file2.delta, file1

在该示例中，file1 需要包括在实际包内，或从某一先前的[Deltas]条目合成。可见，file1_to_file2.delta 将应用到 file1 的一个副本，创建 file2，而 file1 保持不变。通常，file1_to_file2.delta 也包括在[Delete]段内，以在设置程序开始之前将其删除。

注意，可能具有“增量”而没有参考文件，这等效于基于零长度参考文件的增量。这类增量的条目仅省略了{参考}。

[Copy]段允许文件在包内复制。例如，如果设置进程需要三个具有不同名字的相同文件，可以在包内包括文件的一个副本，在设置之前复制其它两个。[Copy]段中的条目的语法包括：

[Copy]

{目标文件名} = {源文件名}

其中，{目标文件名}是要生产的文件的名字，{源文件名}是具有相同内容的现有文件的名字。

示例包括：

[Copy]

file3 = file2

在该示例中，file2 包括在包内，或从先前的[Deltas]或[Copy]条目合成。可见，file2 复制到 file3，并且 file2 保持不变。

[Delete]段允许在设置程序开始之前删除设置不需要的文件。一个常见的使用是删除用来合成所需要的文件的任何增量文件。[Delete]段中的条目的语法为

[Delete]

{目标文件名}

其中，{目标文件名}是要删除的文件的名字。指定的{目标文件名}可能包含通配符，在这一情况下，匹配该模式的任何文件都被删除。如果没有文件匹配给定的名字或模式，则报告无错误。文件删除不是递归的；必须明确地命名子目录。

一个示例包括：

[Delete]

*.delta

在该示例中，匹配“*.delta”的文件被删除。

[Verify]段指定了某些文件要被校验看其是否损坏。该段中的每一条目命名要核实的单个文件，以及对该文件的预期 MD5 签名。如果不能核实该段中的任何文件，则安装失败。[Verify]段中的条目的语法包括：

[Verify]

{目标文件名} = {MD5 签名}

其中，{目标文件名}是预期在包内的文件的名字，{MD5 签名}是该文件的 MD5 签名的十六进制表示。

一个示例包括

[Verify]

file1 = 3D2EDAF98C77086F18925193E471C1C8

file2 = CCF3719A65DB9637864A4340A74575DE

file 3 = 7BEB665C45858982E58D496C3A474CB2

在该示例中，计算每一文件 file1、file2 和 file3 的签名，并与指定的值比较。如果任一签名不匹配，或任一文件丢失，则安装失败。

有一些可以使用[Options]段中的特定伪指令控制的 IPD 选项。忽略未定义的伪指令。

[Options]

Run = update\update.exe ;要执行的程序

PatchDLL = update\mspatcha.dll ;更新的增量核心

Run = 伪指令定义了要执行的设置程序的名字。提取程序已允许将程序标记为当创建机箱文件时使用伪指令文件中的/RUN 选项执行。然而，要运行的程序可能不是机箱中的文件之一，但是可能作为基于其它文件之一的增量被封装。在这一情况下，在机箱内没有文件可标记。Run = 伪指令标识要运行的程序，并且在功能上等效于在机箱内标记该文件。Run = 伪指令覆盖了机箱内标记的文件（如果有的话）。

PatchDLL = 伪指令定义了用来应用增量的 DLL 的名字。提取程序缺省使用

Windows 系统目录的 mspatcha.dll，但是这一伪指令可以明确地命名来自包的一个替换文件。由于在清单中指定的文件与目标目录相关，这一 DLL 是原始包中的文件之一。[Delete]段通常包含一条目以在设置程序开始之前丢弃该 DLL。

传统地，仅执行机箱包，其中扩充其内容、运行设置程序并清除其内容。由于各种原因，也可能期望扩充包的内容，但是不运行设置或清除操作。例如，销售商可能需要获取包中找到的一个或多个文件，但是不希望实际在这一计算机上安装该包。参数选项提供了这一特点，当使用时，提示用户目标目录（或使用/X:目标目录来明确给出）而内容简单地在那里扩充。

本发明的包内增量系统和方法也支持这一概念。当使用这一参数选项（或/X:目标目录）时，提取程序将提取内容而不运行设置程序。然而，当内容包括任何增量文件时，这些增量一般将不以其当前的形式使用的，因此包内增量处理仍按缺省执行，以将内容恢复其自然形式。

一些操作系统作出允许文件关联，其中，特定的程序可以与特定类型的文件关联。一些允许通过文件名后缀（“扩展名”）来关联，其它可能使用文件的其它属性。可以看见，这里描述的自提取包能够仅使用诸如机箱文件的文件集合来实现，当文件被激活时，依赖于文件关联来开始自提取进程。由此，自提取特点的可执行代码不需要必须是包的一部分。

从上述详细描述中可以看到，提供了一种可以在自含包中使用增量压缩来提供数据的方法和系统。显著地减小了包的大小，而自含包的优点对销售商和客户顾客可用。因此，该方法和系统提供了当代计算中所需要的优点和利益。

尽管本发明对各种修改和替换构造敏感，在附图中示出了其特定的说明实施例并在上文详细描述。然而，应当理解，并不意味着将本发明限制在所解释的特定形式上，而是相反，本发明覆盖了其精神和范围之内的所有修改、替换构造和等效物。

这里所引用的每一专利转让给本发明的受让人，并通过引用结合于此。

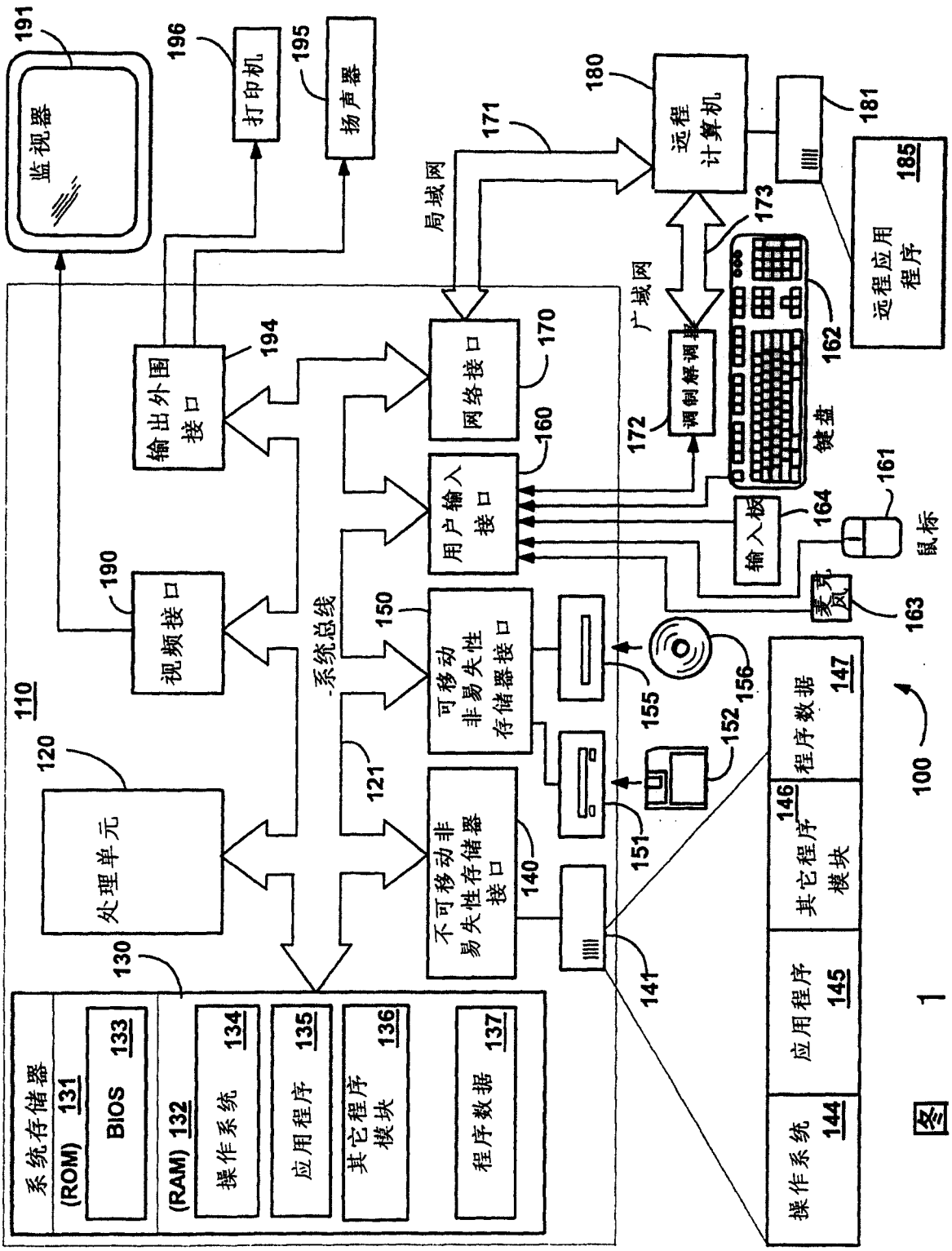

监视器
191
打印机
196
扬声器
195
局域网
171
远程计算机
180
远程应用程序
185
输出外围接口
194
网络接口
170
广域网
调制解调器
172
键盘
162
用户输入接口
160
输入板
164
鼠标
161
麦克风
163
视频接口
190
系统总线
150
可移动非易失性存储器接口
155
156
152
151
不可移动非易失性存储器接口
140
141
处理单元
120
110
130
系统存储器
(ROM) 131
BIOS 133
(RAM) 132
操作系统 134
应用程序 135
其它程序模块 136
程序数据 137
操作系统 144
应用程序 145
其它程序模块 146
程序数据 147
100
121
173
181

图 1

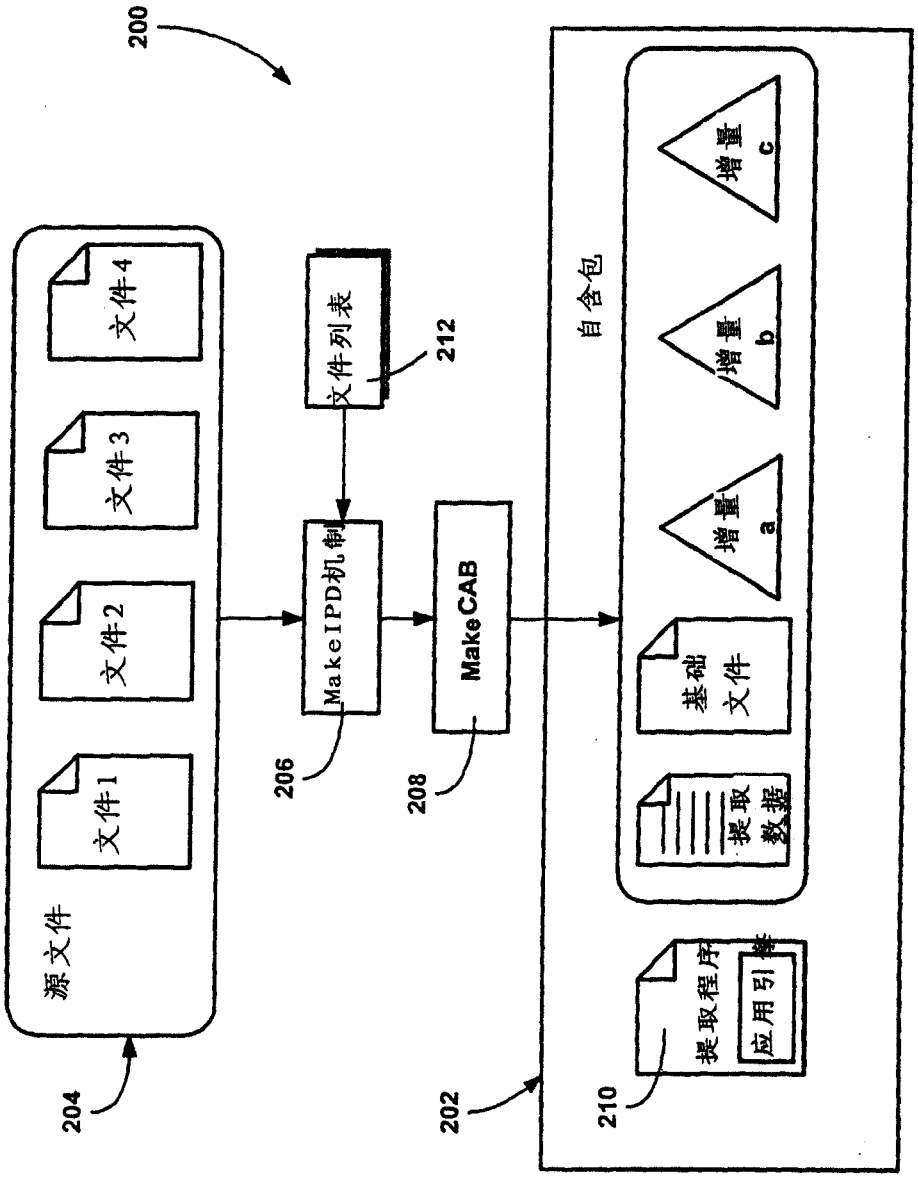

200
源文件
文件1
文件2
文件3
文件4
204
MakeIPD机制
206
文件列表
212
MakeCAB
208
自含包
基础文件
增量a
增量b
增量c
提取数据
提取程序
应用引擎
210
202

图 2

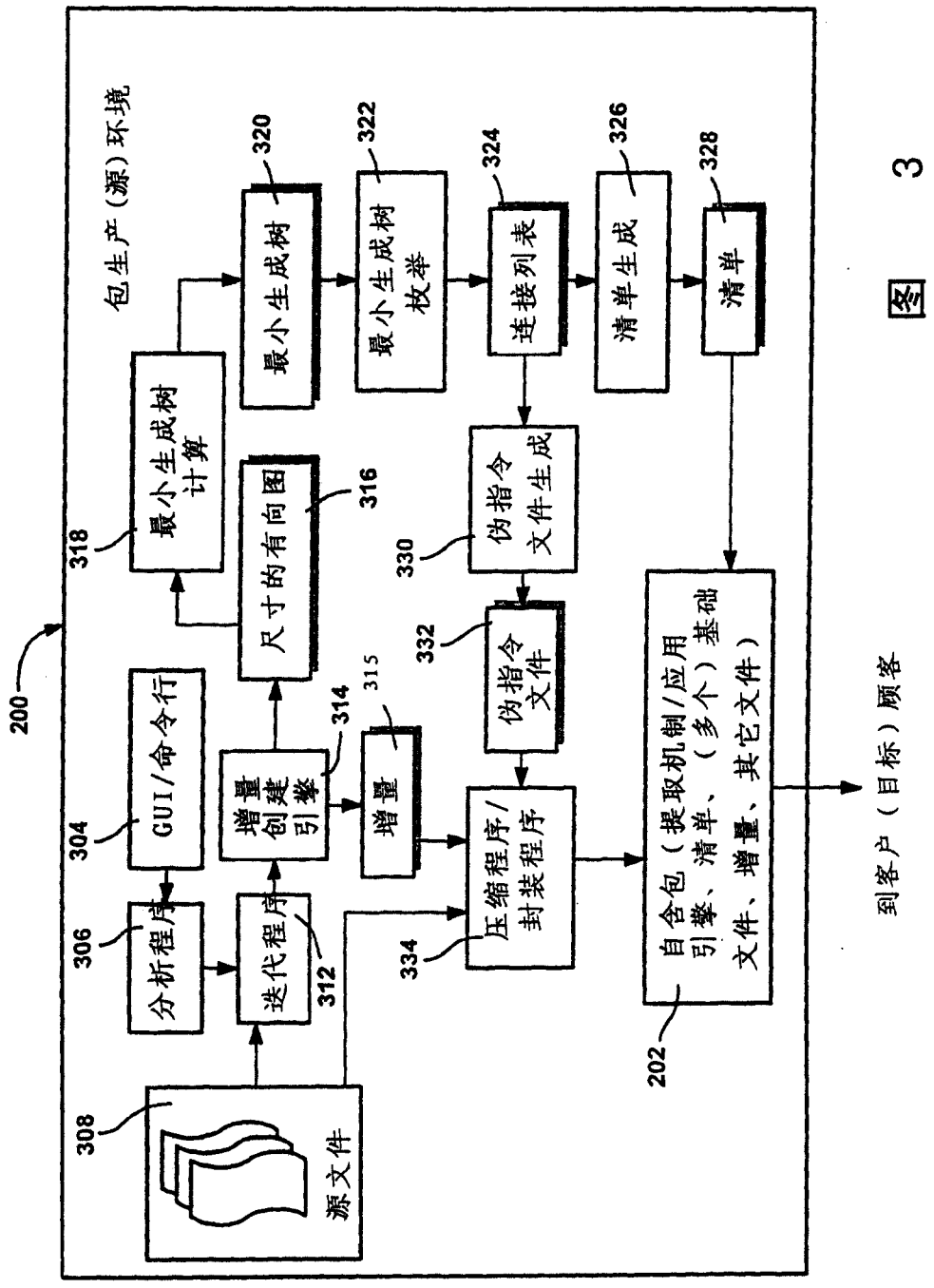

包生产（源）环境
200
308
源文件
306
分析程序
304
GUI/命令行
312
迭代程序
314
增量创建引擎
315
增量
316
尺寸的有向图
318
最小生成树计算
320
最小生成树
322
最小生成树枚举
324
连接列表
326
清单生成
328
清单
330
伪指令文件生成
332
伪指令文件
334
压缩程序/封装程序
202
自含包（提取机制/应用引擎、清单、（多个）基础文件、增量、其它文件）
到客户（目标）顾客

图 3

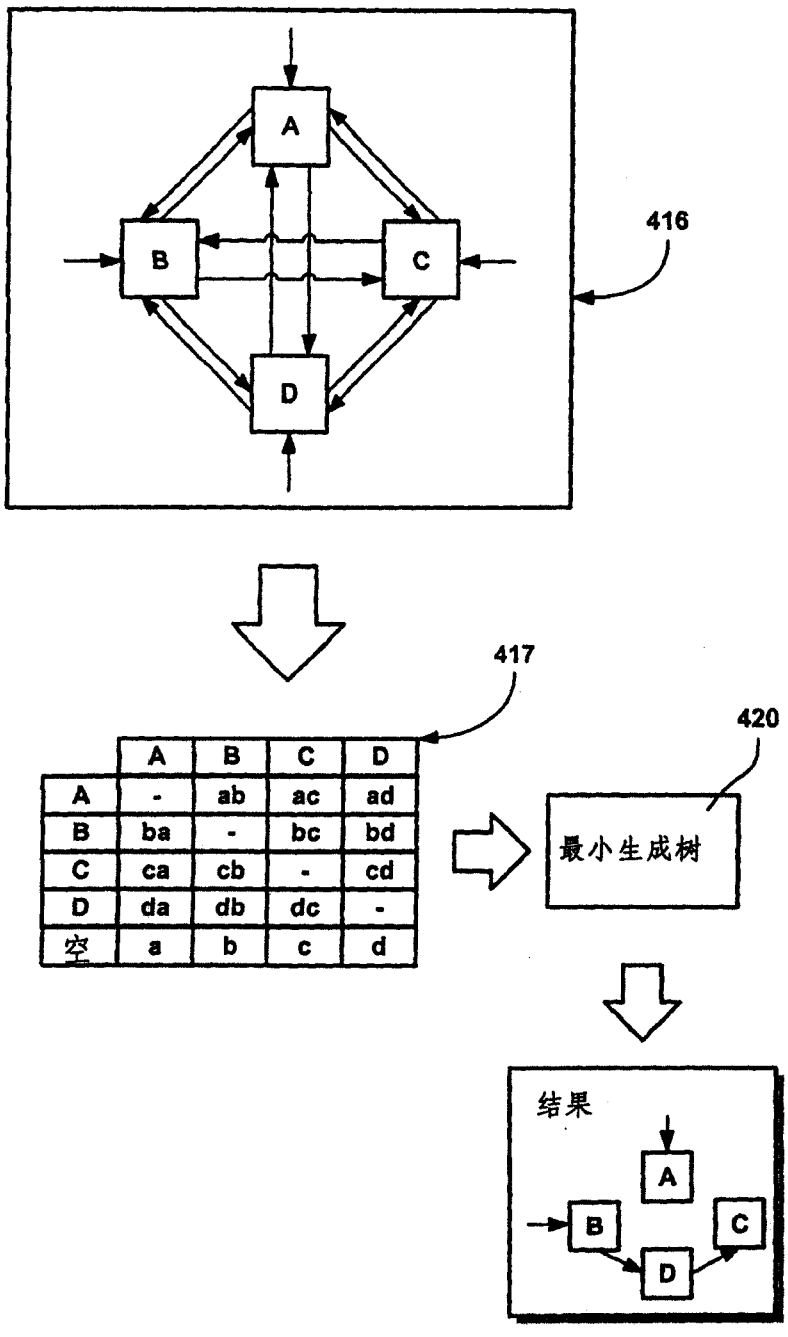

A
B
C
D
416
417
A B C D
A - ab ac ad
B ba - bc bd
C ca cb - cd
D da db dc -
空 a b c d
420
最小生成树
结果
A
B
C
D

图 4

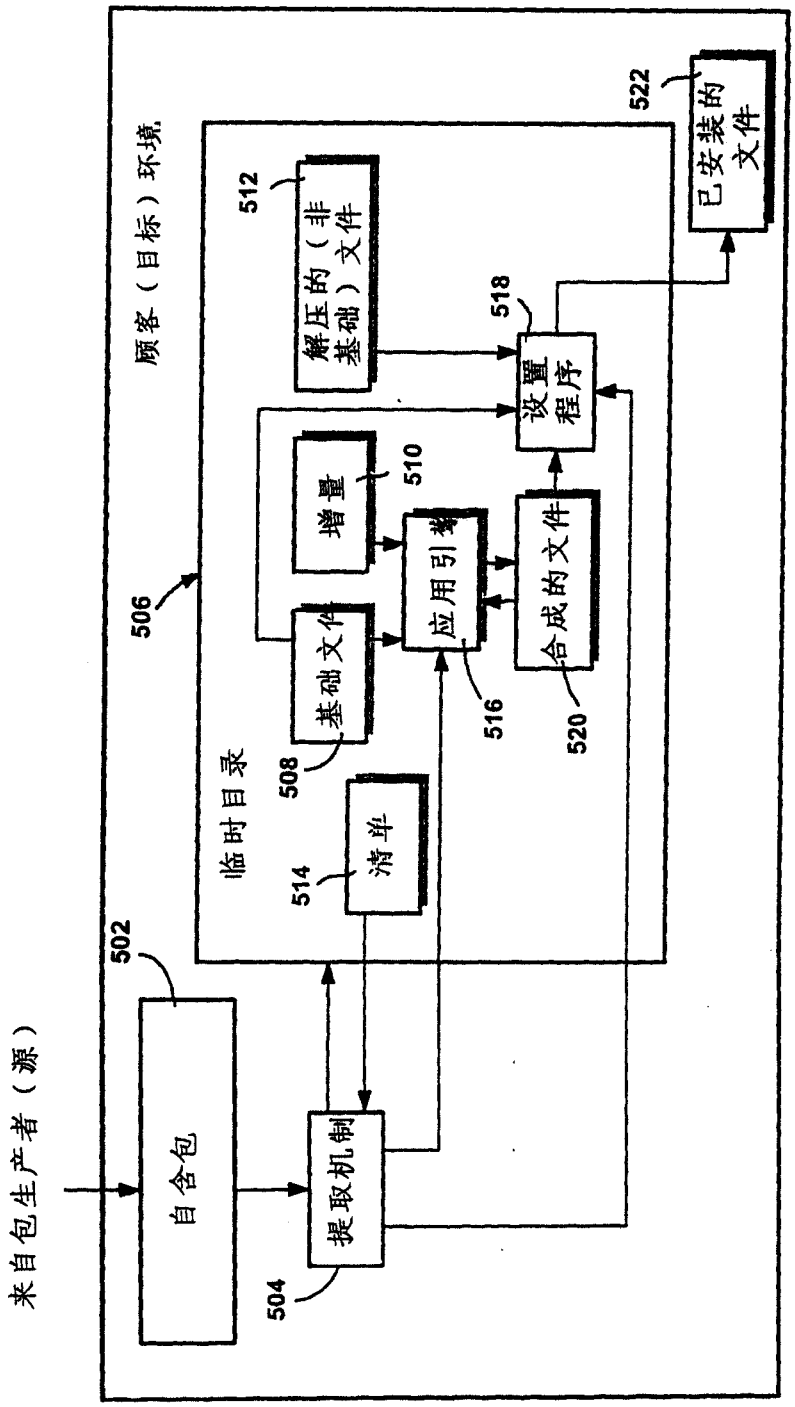

来自包生产者（源）
自含包
502
提取机制
504
顾客（目标）环境
506
临时目录
清单
514
基础文件
508
增量
510
应用引擎
516
合成的文件
520
解压的（非基础）文件
512
设置程序
518
已安装的文件
522

图 5

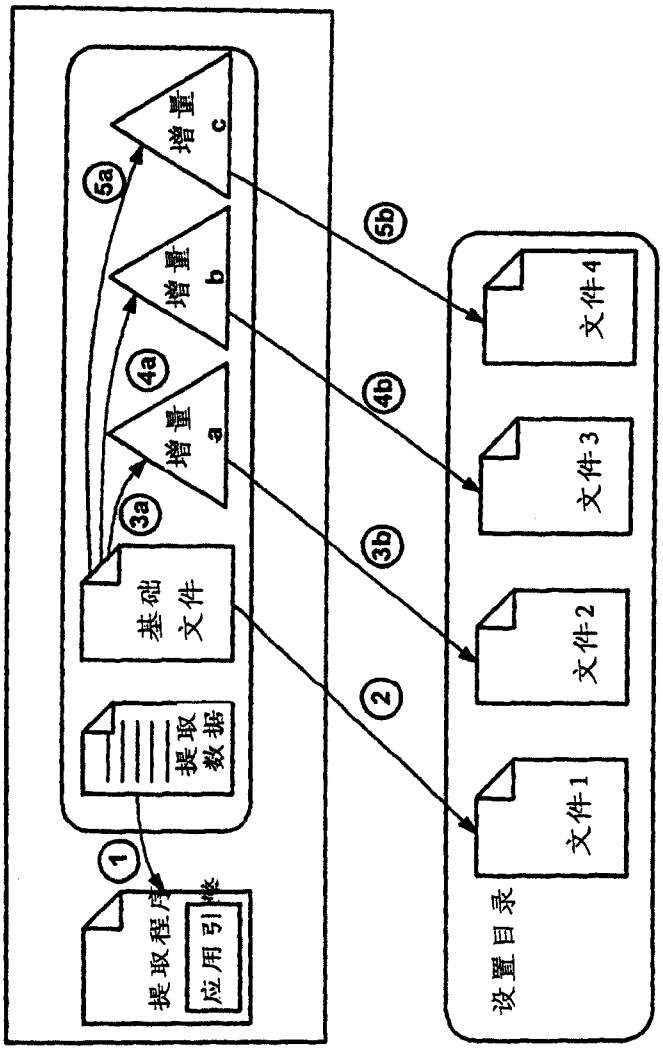

提取程序
应用引擎
提取数据
基础文件
增量a
增量b
增量c
设置目录
文件1
文件2
文件3
文件4
①
②
3a
3b
4a
4b
5a
5b

图 6

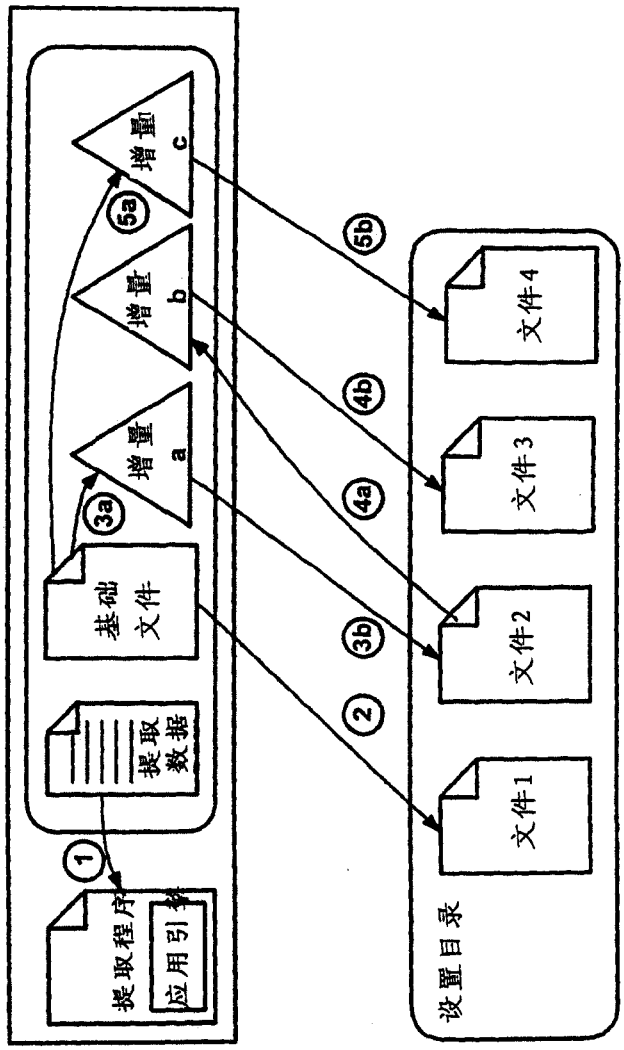

提取程序
应用引擎
①
提取数据
基础文件
3a
增量a
增量b
5a
增量c
设置目录
文件1
②
3b
文件2
4a
文件3
4b
文件4
5b

图 7