[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H03M 7/38 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200380104909.1

[43] 公开日 2006 年 2 月 15 日

[11] 公开号 CN 1736030A

[22] 申请日 2003.10.28
[21] 申请号 200380104909.1
[30] 优先权
[32] 2002.10.30 [33] US [31] 10/285,330
[86] 国际申请 PCT/US2003/034327 2003.10.28
[87] 国际公布 WO2004/042930 英 2004.5.21
[85] 进入国家阶段日期 2005.6.2
[71] 申请人 河床技术股份有限公司
地址 美国加利福尼亚州
[72] 发明人 S·麦克坎尼 M·J·德玛

[74] 专利代理机构 上海专利商标事务所有限公司
代理人 钱慰民

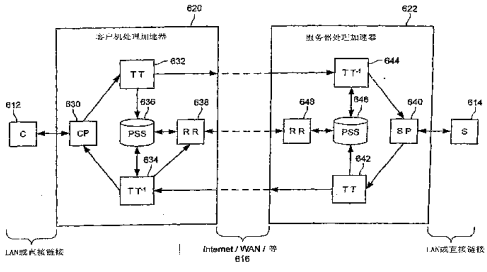
权利要求书 2 页 说明书 19 页 附图 9 页

[54] 发明名称

用于存储器中数据压缩的基于内容的分段模式及包括等级分段表示的传输

[57] 摘要

在一编码系统中，对系统中的输入数据编码。该输入数据可包括在输入数据中重复的或出现在系统中编码的其它输入数据中的符号序列。 该编码包括确定一目标段尺寸，确定一窗口尺寸，在输入数据的偏移处标识符号窗口中的指纹，确定是否要将偏移指定成切割点并按该组切割点所指定的对输入数据分段。 对这样标识的各段，编码器确定该段是否会是引用的段或未引用的段，必要时，用引用标签替换各引用的段的分段数据并将引用绑定存储在用于各引用的段的连续段存储器中。 按等级，可以通过将引用分成组，用分组标签替换分组的引用，存储一个分组的引用和分组标签之间的绑定来重复该过程，如果一个没有呈现，并重复该过程。 等级的级数可以预先固定或从编码的内容确定。



1. 一种解码系统中的输入数据的方法，所述输入数据可包括在输入数据中重复的或出现在系统中解码的其它输入数据中的符号序列，其特征在于，该方法包括：

在多个由偏移和窗口尺寸定义的连续的输入数据符号中标识多个连续的输入数据符号的指纹表示；

从指纹表示确定是否要将所述偏移指定成切割点；

重复以上标识和确定的步骤以到达一组切割点；

通过该切割点组所指定的对输入数据分段；

为各段确定该段是否要成为引用的段或未引用的段；

对各引用的段，用引用标签替换引用的段的分段数据；

对未在连续段存储器中呈现的各引用的段，在连续段存储器中存储一个引用绑定，确定是否任何段序列被分组为索引组；

对每个索引组用分组标签替换分组索引；

对没有呈现在连续段存储器的第一索引组，存储绑定在连续段存储器的分组索引，其中，分组索引绑定使索引组索引与分组标签发生关联。

2. 如权利要求 1 所述的方法，其特征在于，还包括：

将标签组循环地标识成更高级的组，其中标签组是一个或一个以上引用标签组和分组标签的组；

对各更高级的组，用一个分组标签替换更高级组；和

对没有在连续段存储器中呈现的各更高级组，在连续段存储器中存储一个组引用绑定用于更高级组。

3. 如权利要求 1 所述的方法，其特征在于，所述输入数据包括在客户机－服务器网络中的客户机和服务器之间的消息的有效载荷。

4. 如权利要求 1 所述的方法，其特征在于，所述输入数据包括在线备份系统中的部分文件，还包括在在线备份系统中将文件表示成引用标签和分组标签中至少一个的序列并存储连续段存储器的内容，作为部分在线备份系统。

5. 如权利要求 1 所述的方法，其特征在于，所述输入数据包括在一文件系统中的部分文件，还包括将该文件系统中的文件表示成引用标签和分组标签中至少

一的序列和段存储器。

6. 如权利要求 1 所述的方法，其特征在于，所述输入数据包括要用于一文件系统的部分文件，该方法还包括：

当将一文件存储到文件系统中时，用表示成在连续段存储器中引用的一个段的文件的至少一个段将其编码；并

当从所述文件系统中检索一个文件时，将该文件作为解码的文件缓存至局部文件存储器，其中各引用标签和各分组标签由来自连续段存储器的相应分段数据取代。

用于存储器中数据压缩的基于内容的分段模式及包括等级分段表示的传输

相关申请的交叉引用

美国专利申请号：10/285,315，名称：“用于客户机－服务器通信系统的处理加速器”[代理人编号：021647-00010US](以下为“McCanne I”)和本申请同日递交，通过参考包括在此。

背景技术

本发明通常涉及数据压缩，特别是涉及用于压缩的分段。

数据压缩可用于更有效地存储和传输数据。数据压缩是将输入数据表示成压缩数据的过程，使压缩数据包括比输入数据更少的位或符号且可以将该压缩数据解压缩成至少是原始输入数据的合适的近似值。压缩允许更有效的数据传输，因为需要发送给接收方用来恢复原始的那组位(正好或近似地)的位更少了，且压缩允许更有效的存储，因为需要存储的位更少了。

“压缩比”指原始数据中位或符号数与压缩数据中位或符号数的比。例如：如果 100 字节的数据序列可由 5 字节的数据表示，则在该例中的压缩比为 20:1。如果不需要确切地恢复输入数据，则可采用所谓的"有损压缩"，通常导致比“无损压缩”更大的压缩比。在压缩会是透明的典型应用中，压缩应为无损的。

基于结构和输入内容的统计的压缩数据是常见的。典型的压缩器参考输入中的符号值、输入中特定符号值的位置、输入中各符号值之间的关系以及输入数据源的预期属性来接收输入数据流或数据块并生成压缩的数据流或数据块。例如：当预计输入数据为英文文本时，很有可能接着（句点）符号的源的输出为“”（空格）符号。该源的此属性可由压缩器采用。例如：空格在压缩数据中可以根本不用符号来表示，从而将数据减少至一个符号。当然，为了能无损地解压缩已压缩数据，压缩器必须为后接空格的句点的各情况编码专用符号。然而，鉴于它们经常出现，预计可以比专用符号有更多省略，因为整个结果为净压缩。

一种使用可能包括重复的输入字符的序列的源的压缩方法是字典法。采用此方法，建立一个符号序列的字典并用对字典的引用代替字典中每次出现的符号序列之一。当压缩器和解压缩器可以访问同一字典时，解压缩器能通过用相应的条目代替各字典的引用来对压缩数据解压缩。通常，字典压缩假定可以将输入流分成段，且所述段将会在输入流中再次出现。

当然，为了使字典法工作，解压缩器必须具有压缩器使用的字典的副本。当压缩是为了减少传输工作时，一般用传输通路来分隔压缩器和解压缩器，在该传输通路上的工作减少了，但如果通过该通路发送字典的话，则加在该通路上的负载会增加。当将压缩用于减少存储时出现类似的问题，因为需要存储字典使解压缩器可以对其进行访问，而这增加的存储工作。在一些模式中，字典为固定字典，这样就可以将其分摊至许多压缩中，以将每次压缩字典的成本减少到可忽略的程度。在其它模式中，字典为自适应的，但，是可以从解压缩器已经可用的数据重建的，但是是作为预先解压缩的符号。

压缩适用于被带宽约束限制了网络通信量的网络。一个例子是诸如因特网之类的广域网(WAN)，它每次使用比诸如专用局域网(LAN)或专用广域网 WAN 之类的其它网络更少的空闲带宽。也于成本的原因，许多人愿意使用非专用 WAN 而不仅仅依靠 LAN 或增加专用 WAN，但他们受到非专用 WAN 的性能的限制。压缩可能可以允许将低带宽链接用于高带宽应用，因为它减少了表示更大输入序列所需的实际位数。相似地，通过减少表示系统中所有文件所需的位数，压缩能可能提高文件系统的性能和容量。

通常，通过企业体系和网络存储和传送的数据具有高度信息冗余表示。例如：向一公司的大量接收者发送的电子邮件消息和附件在存储系统中生成许多消息数据的冗余副本并导致要通过网络发送的冗余通信量。同样，在一企业中的许多电子文件享有极高度的公共性，因为不同的雇员在不同的设置中用类似的共同信息件工作。

如果压缩该数据，就会改善网络性能并增加有效存储容量。传统的压缩模式可以通过检测在一输入符号流中的统计相关并根据该统计相关在尽量少的位中编码流的符号来利用这些冗余中的一些。一些基于字典的压缩模式被称为“通用代码”，因为在包括了输入符号符合平稳随机处理的假设在内的各种假设下它们收敛到最佳压缩模式(香农极限)。这就会暗示仅通过布署一个对网络中的网络通信量或存储系统中的文件数据执行最佳压缩的通用编码系统就可以实现最佳性能。

然而，这一方法不必适用于实践。例如：众所周知在路由器的网络接口上使能压缩会提高性能，但仅仅是少量地(30%是典型的，但这取决于基础通信量)。传统通信编码模式的一个问题在于如果基础数据输入具有非静态统计则它们不必收

敛于最佳比。另外，如果基础统计是静态的，但它们显示“远程依赖”(LRD),则通信代码与最优性的收敛比率可以不切实际地慢（可能指数规律地慢）。这具有重要的意义，因为许多研究提供了网络通信量显示 LRD 的证据，而实际上有关基础数据处理的最佳模式是否为 LRD 随机处理或非静态处理还有待商榷。其它研究显示文件统计(同样大小分布等)也显示 LRD。简而言之，这都意味着通信编码的传统方法不一定是最佳实际解决方法，而利用典型数据源的远程依赖的技术可能会更好。

检测远程相关的一种强力方法是运用基于字典的压缩模式，用大宽度在数据源（文件、通信流等）中搜索重复的图形，用名称或标签表示那些图形并将相应的数据存储在与名称或标签相关联的表格或数据库中。使用 LRD 可以保留一个很大的数据窗口使系统远溯到输入(或时间)地任意对等，以检测远程依赖图形。该简单的模式直观地匹配一个企业中的信息结构。即，许多相似的信息源即随时间慢变化又出现在不同的语境中（电子邮件、文件系统、Web 等）。随着基础技术的改进(例如：盘和存储器变得越来越便宜)，此方法变得更实用了。然而，强力方法仍有缺点。

一个缺点是在一个位流中搜索匹配数据的任意图形在计算上很贵且在出现的 LRD 统计中快速地有效地找到最佳解的通常问题尚未完全解决。另选的方法的放弃找到最佳解的想法而关注预先形成好的根据 LRD 并实用且可行的近似解或启发式解。

一个在此结构中证实有用的工具是通过根据输入内容本身而不是一些外部施加的方框或框架模式分段数据找到数据中重复的图形的建议的启发式解。见，例如，Muthitacharoen, A.etc.,"A Low-Bandwidth Network File System", in Poceedings of the 18th ACM Symposium on Operating Systems Principles(SOSP '01), pp.174-187(Chateau Lake Louise, Banff, Canada, October 2001)(in vol. 35, 5 of ACM SIGOPS Operating Systems Review, ACM Press)。在本文所述的 LBFS 系统中，散列取代部分传送的文件，且接收方用该散列重建在文件系统上哪个文件的哪一部分对应于替换的数据。在文件的匹配部分的语境中对根据输入内容分段的另一范例进行了说明，由 Manber 所描述的，"Finding Similar Files in a Large File System", USENIX Proceedings, San Francisco 1994(可用作

University of Arizona Dept. of Comp. Sci.Technical Report TR93-33)。

另一尝试是在网络层上应用了其它通过字典型压缩技术减少网络通信量。一个此种技术包括用标号表示部分网络通信量并在连接的各端维持标记的表格。见，例如，Spring,N.,et al.,"A Protocol-Independent Technique for Eliminating Redundant Netwok Traffic", in the Porceeding of ACM SIGCOMM（August 2000）。如该文献中所述，通过标识重复字符串并用要从连接的各端共享的表格中解出的标号替换该重复的字符串包含冗余的网络通信量能减少。因为它仅作用于单个分组，由此方法产生的性能增益受限于分组有效载荷大小与分组标题的比(因为分组标题通常不能用所述技术压缩)。同样，因为在分组级实施的机制，它仅应用于通信通路的两端配置了设备的区域。在某环境下，此配置即使不是不可行，也很难实现。同样，通过用先进先出的替换策略的较小的基于存储器的表格引用网络分组(没有例如大的基于盘的备份存储的帮助)，该方法的有效性局限于检测和使用及时地相当局部化的通信冗余，即，该方法不能采用基础数据流的 LRD 特性。

另一方法是减少包括高速缓存在内的网络通信量，其中如果数据的副本不在局部高速缓冲存储器中就不通过网络发送对数据的请求。如在这里所使用的，术语“近”、“远”、“局部”和“远程”可指物理距离，但更常指有效距离。两个计算机、计算设备、服务器、客户机、外设等之间的有效距离至少约是两个计算机之间取得数据的难度的量度。

当高速缓存对不变的数据块适用并在不同名字下的相似形式中找不到时，在许多情况下仍需要改进。在文件高速缓存中，高速缓存的单元通常是一个文件的一块或整个文件。如果相同的数据出现在不同文件中，或两个文件仅有很小不同，高速缓存不会去除冗余或利用它们减少通信成本。即使将数据对象分割成许多块，且各块分开高速缓存，最终结果仍旧无效，因为在基础对象中小量插入数据删除会导致数据通过许多块（如果不是所有块）移位，从而无效高速缓存的益处。这是因为块是在输入流中任意强加的，所以不可能检测到仅对基础数据作出小的改变。

由于以上，可以对网络环境中、存储系统中和其它地方中压缩数据作出改进。

发明内容

根据本发明的一个实施例的编码系统，对一个系统中的输入数据进行编码。

该输入数据可包括在输入数据中重复的或在编码于系统中的其它输入数据中出现的符号序列。该编码包括确定一个或多个目标段大小，确定一个或多个窗口尺寸、在输入数据中偏移处的符号窗口中识别指纹，确定是否要将该偏移指定成割点并按由该组割点指定的那样分割输入数据。对于这样标识的各段，编码器确定是否该段会成为引用段或未引用段，用引用标签代替各引用段的分段数据并如果必要，在不变段存储器中为各引用分段存储一个引用绑定。按等级，可以通过将引用标签字符串分割成组，用分组标签替代分组的引用，存储分组的引用和分组标签之间的绑定来重复该过程，如果有一个没出现，则重复该过程。可以预先固定等级的级数或从编码的内容确定它。

参照以下详细说明和较佳实施例，本发明的其它特征和优点将会很明显。

附图说明

图 1 为根据内容和分段索引用分段来编码数据流或块的编码器的方框图。

图 2 为用来自不变段存储器的分段绑定来解码数据流或块的解码器的方框图。

图 3 为更详细地示出编码过程的图。

图 4 为更详细地示出解码过程的图。

图 5 为示出分级编码的过程的图。

图 6 为具有多级段存储器的不变段存储器的示例图。

图 7 为动态地示出使用太少编码级的问题的示例图。

图 8 为动态地示出使用太少编码级的问题的示例图。

图 9 为组织成保持任意深度的引用的不变段存储器的示例图。

图 10 示出使用图 9 的不变段存储器的解码过程。

图 11 示出分级内容导出分段。

图 12 为联网的客户机－服务器对的方框图，其中通过客户机端处理加速器("CTA")和服务器端处理加速器("STA")路由客户机和服务器之间的通信量。

图 13 示出使用分级内容导入分段的文件系统。

图 14 示出近线文件系统(NLFS)和文件服务器前端。

本发明的详细说明

如阅读此公开的后所显见的本发明有许多应用。在说明根据本发明的压缩系统的实施例时，仅仅描述了很少几个可能的变形。其它应用和变形对本领域的普通技术人员来说将很明显，所以不应将本发明理解成仅限于范例，而应根据所附的权利要求对应进行理解。

可以用几个不同的设备进行编码(编码和解码)，所述设备可包括硬件、软件或两者都包括。可以用计算机、计算设备、外围设备、电子设备等和/或使用由这些元件执行或控制的应用程序来编码。编码对诸如 McCanneI 中所述的传输过程是难免的。使用这里所述的编码装置和过程，可以通过减少在临界时间在临界信道所需传送的位数来改进通过网络处理的响应度。同样，可以将编码系统集成至单机存储系统以优化容量。在此环境中，可以增加存储系统的有效容量，因为这里所述的编码系统提取数据的共同序列并将它们仅仅存储一次，即使该序列可能在可能的许多不同文件中出现多次。

图 1 示出编码器可能有的特定输入。如其中所示，编码器具有用于接收输入数据的输入和参数的输入，例如：目标段大小、窗口尺寸和其它控制参数以及输出数据的输出和在编码过程中生成的绑定，它们按照需要被存储不变段存储器(PSS)210 中。在运行中，编码器 140 会处理输入数据、标识数据分段，用引用代替分段的数据，以绑定的形式向 PSS142 提供分段数据和分段引用并输出编码的数据。编码的输出数据可包括未引用的分段数据、嵌入的绑定和/或引用的分段的引用标签。在最简单的情况中，输出数据全部是引用标签。

编码器利用目标分段大小参数来控制段的平均大小。通常，段是可变长度且如下所述在该大小的选择中有设计折衷。

图 2 示出解码器 200 和 PSS310，它们合在一起执行解码，该解码是编码器 200 编码的逆过程。如上所述，编码的数据可包括引用、绑定和未引用的冗余数据。当解码器 150 在收到的数据中遇到绑定时，它可以使用该绑定中的分段数据来重建原始数据，并且它还能将绑定存储在其 PSS 中。当解码器 150 遇到没有绑定的引用时，它能用该引用从 PSS 152 获取分段数据来重建该段。如果在 PSS152 中没有找到分段引用，则解码器 150 能发送一个对分段数据的请求。

图 3 为更详细地示出编码过程的图。如其中所示，将要编码的输入数据存储在由分段分隔符 222 分割的高速缓冲存储器 220 中，该存储器为 PSS210 生成输出引用和绑定。将所需的输出引用、诸如段 224 之类的未引用的段和绑定提供给输出缓冲器 330 以形成编码器的输出。

图 4 为更详细地示出解码过程的图。如其中所示，将已编码的数据缓存至输入缓冲器 320 中。从缓冲器的内容提取绑定并存储在 PSS310 中。将引用标签提供给换装器 325，用引用分段数据代替它们并将该数据放入输出缓冲器 330 中。将未引用的分段数据直接输出到输出缓冲器 330，用于作为重建数据的最终输出。

如上述附图所示，编码过程的一个方面是输入数据的分段。在分段的过程中，标识“分割点”（例如在一段结束下一段开始处的输入数据的偏移）等于将该输入数据分成分离的数据结构等。

如果分段模式设计得合适，分段界线应该一直出现在相同数据序列的相同位置，不管数据出现的语境如何。如果是该种情况，应该对各重复数据用同样的方式分割经常重复的数据图形并可以将系统设计成有效地识别这些重复的图形。例如：在较大文件（例如：文字处理文件、幻灯片简报文件或在 Web 页上）的许多不同位置中出现的特定数据序列（例如：广泛使用的 GIF 图象或表示常用图标的位图）会一直被找到并以同样的方式分割，不管什么数据围绕它。

为了实现这一特性，这里的分段模式利用在数据本身中的信息而不是用外部施加的诸如块大小、处理边界等之类的参数引导分割过程。当输入数据被编码过程消耗时，输入符号的各种值和结构引导分段过程(如下所述)。因此，系统具有“自同步”特性，其中如果将相似的数据图形呈现给其输入，则检测到相同的段边界，导出与过去所见到的相同段的信息。另外，这意味着该系统是耐插入、删除或其它基础输入数据中的变化的，因为一旦找到先前呈现的段边界，新的段会与从该点处开始的现存段相匹配(假设从该点处始的数据图形在过去见到过)。

因为此模式由输入内容引导的，这里将其称为"内容－引导分段"。通过用跨极大时标(即，显示 LRD 统计)的重复的图形将内容－引导分段应用于输

入流，产生相同的段而不必留意在过去见到过的数据图形的整个历史。即，分段过程可以根据输入本身而不必搜索已找到的现有数据来简单地将输入分段。虽然通常这种方法不产生最佳分段(即，最大化段尺寸的同时最大化找到的重复段的数量)，它在复杂性和性能之间提供了好的拆衷－该模式在利用某种类型的 LRD 方面有效，同时导致有效的实用的实施。

内容－引导分段

诸如图 3 中的分段分隔符 222 的分段器作用于一组输入数据的一部分。该分段器提取一定数量的输入符号(由当前偏移和窗口尺寸参数所决定的)并计算窗口上的散列函数。这被称为窗口的指纹。确定性的指纹指示器函数为每个指纹返回一个布尔值，指示是否要将该偏移认作切割点，从而定义一个段边界。最好是窗口不伸出应用数据单元(ADU)的两端,从而各窗口成分都完全来自一个 ADU。因此，最好将输入数据流中一个 ADU 结束且另一 ADU 开始处的指示提供给编码器。然而，该指示不一定用于系统的正确操作。相反，这些指示允许编码器在流中 ADU 边界结束的地方强制一个段边界，从而防止诸如等待可能永远不会到达的数据的额外延迟之类的实际无效（例如：换成依赖实施超时来强制段边界）。另外，将段边界限制在单个 ADU 中防止系统检测和存储将来不太可能重复的段，而这种情况在段跨两个 ADU 时可能会发生。

当指纹指示器函数的值为 1 和 0 时，1 可表示切割点而 0 表示切割点以外的。因此，在一个约定下，当函数为具有给定偏移和窗口的给定指纹求的值为 1 时，在由输入数据的第一符号定义的符号处生成一个新的段(按输入数据的次序)。在该约定下，如果函数求的值为 0，则多消耗一个输入符号且窗口预先覆盖此新符号(且将最近的符号从该窗口中去除，但保留在当前的段中)。因此可以将目标段的大小控制成指纹指示器函数的一个参数。

因为选得好的指纹函数会趋向于在指纹中生成随机位图形，可以将由目标段大小参数化的随机变量的伯努利分布函数用于将散列值映射至指标值。例如：可以选择参数，使平均起来，假设指纹输入的随机分布每 M 次有一次，则函数值为真。这样，平均起来，段的大小会是 M+W 字节，其中 W 为使用的窗口尺寸(因为段至少有 W 字节大)。因此，M 的值确定目标段的大小。

变形的分段过程允许窗口定义段边界的开始而不是边界的结尾。这允许

段任意地小，因为用于定义指纹窗口的数据的图形不一定要作为段的一部分消耗。

在另一实施例中，不需要清楚的指纹窗口。而是在可变数量的输入符号上计算散列函数。当各输入符号被消耗时，在消耗的符号的整个缓冲器上重新计算指纹函数。又，下一个用于指纹的指示器函数决定何时和是否要在输入中的当前点处插入一个新的段边界。这一方法通常没有先一种方法有效率，因为在段的起始处附近的数据中的变化影响以后的分段过程。因此，错过了否则会产生并与先前的段相匹配的段边界，因为可以很远地从希望的段边界去除的数据中的变化。早先描写的窗口方法正是完成这一目标。

可以通过使用在读取输入符号时能增量式更新的散列函数来使指纹散列模式有效率。用于增量地更新窗口的散列函数的有效产生的例子是已知，这里就不再详细说明了。见，例如，Rabin,"Fingerprinting by Random Polynomials",Technical Report TR-15-81,Dept. of Comp Sci.,Harvard University（1981）。

当定义了新的段时(通过找到新的段边界)，将该段与所有存储在 PSS 中的现有段比较。可以通过维持由对分段数据（或是整个分段内容或是部分分段内容、分段指纹值或其组合）计算的散列函数连接的段的索引使该查找过程有效率。对于各散列值，该索引包括呈现在散列至该值的 PSS 中的那组段。因此，要确定段是否在 PSS 中，编码器计算有问题的段的散列并用所述散列查找段索引。如果查找失败，则该段不可能在 PSS 中。如果查找成功，则编码器可以将由查找返回的各段与有问题的段进行比较以核查出完全匹配。这处理将段散列乘以同一散列索引的罕见情况。

继续编码过程的说明，如果没有呈现可接受的段(通常为一相同段，但在特定情况下，不需要完全相同)，将新的唯一名称分配给新的段而将绑定(引用标签、分段数据)输入 PSS 中。在需要时，可以压缩绑定，但在不限制 PSS 的最简单的情况中，可以不将绑定存储为数据库中的记录，各记录中的字段用于存储表示引用标签的字符串而数据用于表示段数据。另外，可以用有效地利用任何分段数据中的短程依赖的传统压缩算法对该绑定中的段数据自身进行压缩。这在无终端段(下述)中特别有用，该无终端段包括标签字符串，

该字符串根据分段命名模式的具体内容可以具有高度冗余。

如果呈现可比较段，则将其先前定义的引用标签用于新的段且不生成新的绑定。然后输出该引用标签而不是实际分段数据，如图 3 中的序列 226 所示。因此，原始输入数据表示成绑定存储在 PPS 中的数据对象的一序列引用标签。某些数据可以不被引用标签替换，例如确定用引用标签替换特定段(例如：图 3 中的段 224)不会在实质上改善性能的地方。然而，在某些实施例中，所有段都由引用标签表示。

引用标签可能没有被简洁地表示，导致将超过必要的位用于序列 226。如果是这样，则可以用传统压缩方法（例如：引用标签的差分编码法，后面是运转周期编码法和或霍夫曼编码法或其它方法）来压缩引用标签。

图 4 示出解码过程，它是图 3 所示的反过程。将已编码的数据（引用标签和可能还有绑定以及未引用的分段数据）接收至输入缓冲器 320 中。从该输入缓冲数据提取绑定并将其存储在 PSS310 中并将未引用的分段数据移入输出缓冲器 330 中。利用引用标签在 PSS310 中定位绑定，将在编码数据中表示成引用标签(或压缩的引用标签)的引用的段通过换装器 325 取代它们的分段数据，从而获得相应的分段数据。因为引用标签对唯一数据是唯一的，总可以提供正确的分段数据。如果引用的绑定没有呈现在 PSS310 中，可以从例如 PSS210 请求它们。当为了存储的目的编码时，PSS210 和 PSS310 可能是一个由编码器写由解码器读取的数据结构。

在许多情况下，为了好的性能，需要仔细地选择编码过程的参数。如果将目标段的尺寸选择得很大，则有效压缩比可能会高，因为大量的符号组成段并由表示段的引用标签的较少量的位所取代。然而，大的目标块尺寸可能会错过较细小的数据重复，并增加存储和移动多个近似数据段的开销。另一方面，如果将目标段的尺寸选择得很小，则有效压缩比可能会低，因为表示引用标签所需的位数可能不是充分小于原始分段数据中的位数。通常，依靠底层静态数据，特定分段模式是有效的。这一问题可以通过将分级引入引用标签模式来解决。这里将此方法称为分级内容－引导分段(HCS)。

分级内容－引导分段

为了获得大目标块尺寸的优点(例如高压缩比)和小目标块尺寸的优点(诸

如这样注出和分段细粒重复)，可以利用分级引用。在该系统中，可以用小目标块尺寸分割要编码的输入数据，产生许多引用标签。将该引用标签依次分组并用组标签替换，其中引用绑定(组标签、形成该组的引用标签的序列)存储在类似于 PSS 的结构中。这允许对各种重复数据使用的单个技术，不论重复图形是以细粒或以粗粒出现的。为了捕获细粒重复图形，将目标块的尺寸选择得较小。小目标块尺寸预计会导致更冗长的编码引用流，但分极引用会减少使许多引用标签图形重复的开销，实际上使在最终结果中的各结果标签在能找到重复的条件下表示尽可能大的输入数据跨度。分级引用可在各级作用不同的目标块尺寸，所述各级调整到在该级的引用名称的相对尺寸，或它可以使用同一尺寸。类似地，它可以在分级中的各级使用不同的指纹函数和/或不同的指纹窗口尺寸，或从始至终使用相同的函数。

以下描述了两种该类模式。图 6 示出分级引用编码的一个范例。将要编码的输入数据加载到输入缓冲器中并将该输入数据分段成段 S_A、S_B、S_C、S_D、S_E 和 S_F。在该例中，用引用代替最前面 5 段且该引用碰巧为 R^1_{15}、R^1_{16}、R^1_{17}、R^1_3 和 R^1_8。注意该引用不一定依次，并且此例示出一些引用(例如：R^1_3 和 R^1_8)可以是已遇到过的分段数据，在该情况中没有使用新的段，但引用是对预先有的段。

理想地，编码器会确定例如该经常重现的序列（R^1_{15}、R^1_{16}、R^1_{17}）和（R^1_3、R^1_8），籍此将它们分组并用分组标签(例如：R^2_1、R^2_2)分别替换它们。然而，解决此问题通常很难(在难度上类似于在基础数据上直接解决的相同问题)。因此，编码器将内容－引导分段重新用于引用标签序列，产生与原始方法相似但更高级的优点(即，较低的复杂度和不依赖于移位在输入序列中找到图形和基础数据的局部变化的能力)。因此，如果分隔符决定序列(R^2_1、R^2_2、S_F)对应于在此更高层的新的段（通过重新应用指纹函数、窗口和指示器），则可以用对新段(R^2_1、R^2_2、S_F)的引用表示原始的输入数据。相应地，可以将绑定输入将新引用标签与新的更高级的段相关联的 PSS 中。虽然在本例中没有呈现，最终的序列可以具有来自任何或所有等级的引用标签和分组标签。按照需要，将分段绑定和引用绑定存储至和/或提供给解码器。

固定级 HCS

用固定级 HCS 将 PSS 构筑成一组 N 个(大于一个的不确定整数)绑定表格 PSS^1、PSS^2、...、PSS^N。绑定表格 PSS^1 在引用标签和分段数据之间提供绑定。绑定表格 PSS^2 在索引标签序列和组标签之间提供绑定。其它绑定表格为参照标签组等组提供绑定。这在图 7 中示出。

绑定表格可以为各段和引用标签的一个任意序列存储一个任意的位串。使用该例来自图 6 的分段和表示，PSS^1 可以持有绑定(R^1_{15}、S_A)，(R^1_{16}、S_B)，(R^1_{17}、S_C)、(R^1_3、S_D)和(R^1_8、S_E)而 PSS^2 可以持有绑定(R^2_1、$R^1_{15}+R^1_{16}+R^1_{17}$)，($R^2_2$、$R^1_3+R^1_8$)。

使用该模式，输入缓冲器中的所有数据最终都可以用单个标签 R^N_1 来表示，且如果数据序列再次出现，将在单个引用符号中有效地表示它。同样，如果再次出现了该数据的子集及稍微变化了的数据部分，则类似的部分会有效地由相同紧凑符号串来表示且只有不同处将由区分紧凑符号串来表示。

这一与基于内容的分段相结合的编码过程的分级分解具有局部改变数据的有吸引力的特性，该分段即使任意地大也不影响重复数据的未变化部分。在使用固定大小分段的系统中，例如，如果靠近数据对象开头的几个字节被插入或删除，则每个段都会改变，因为在各固定大小块中的数据都要移位，因此对系统看上去不一样。然而在图 6 的例子中，可以将任意量的数据插入位于由段 S_C 覆盖的点处的对象且仅仅会影响引用 R^1_{17} 和 R^2_1。对于大的数据对象，这种影响的局部化允许由各种在此结构的顶部建立的算法利用的重要性能节约。

可变级 HCS

引用分级可以具有由运行时间确定的可变深度而不是前面例子是的固定的 N 级。这消除了必须选择一个可用于所有类型和大小的数据的 N 的值的问题。

如图 8 所示，如果编码级数太小，则大块数据的编码的引用流仍会需要许多符号，因为许多引用标签仍需要在基础数据中编码。也就是说，如果有更多分级，则可以进一步减少表示在最顶级的基础数据的符号数。然而，如图 9 所示，如果编码级数太大则编码会导致用于小块数据的不必要的开销，因为会不必要地定义和发送引用标签。

为了解决这一问题，图 10 示出组织成保持任意深度的引用的不变段存储器。这里，平等地对待所有引用而不是各引用标签具有专用等级(上述例中的上标)，不论它们是分段数据符号的序列的引用标签、引用标签的序列或是两者的结合。如图 10 所示，引用标签 R_1 绑定分段数据 S_A，引用标签 R_2 绑定分段数据 S_B，引用标签 R_i 绑定一组引用标签(R_1, R_7, R_9)等。

如果编码器能根据输入流弹性地选择合适的编码级，应通知解码器及时在任意给定点出现的分级级数。因此，应该以某种方式在编码位流中传送此信息。在一个实施例中，编码引用流清楚地用特定代码逐字指示出在位流中的级数。在此方案中，当解码器改变使用的级数时，它会发出一个代码来清楚地指示该变化。

在另选的实施例中，可以通过将各段(如绑定中传送的)清楚地标记为即是非终端又是终端(如图 10 中所示)来在编码的输出中传送自适应级结构。这里，终端段表示最终输出数据的字符串而非终端段表示标签串。此信息会作为"叶位"存储在 PSS 中，指示各绑定，不论它终止分级并表示最终输出数据，还是指引用其它绑定的标签序列。一组叶位(例如：一个"1")指示该记录是对终端段且记录的内容中没有任何其它分隔符的分段数据，而清除的叶位(例如：一个"0")指示该记录是引用标签序列。当引用标签为固定长度或具有唯一可解释性时，不需要用间隔来在分组中界定引用。在另一实施例中，对各记录的内容编码，使读出器能在特定叶位不呈现的情况下确定该内容是否包括引用或只不过是分段数据。

为了实施可变级 HCS，当编码器消耗输入数据流时，它生成新的引用并将它们附加至第一级引用标签的序列。每次将新的引用附加到此成长的标签块时，编码器确定是否应用应用于引用标签序列的内容导出分段来定义一个分段边界。从而利用指纹窗口和指纹指示器函数指示在哪里建立一个新的段边界来在标签序列上计算指纹。注意，这些参数不依赖于解码过程且不需要让解码器知道(相反，它们控制在解码器的性能折衷)。如果指纹函数指示新附加的引用不使序列定义段边界，则编码器继续消耗下一输入数据符号。

然而，如果编码器没有在第一级引用序列中检测到新的段边界，则新的段定义为包含引用标签串。与在输入数据中定义新的段时发生的过程相似，

将此新的段与合适 PSS 中现有的所有段相比较。如果没有找到现有的段，则从 PSS 中找回该段的现有引用标签。如果没有找到段，则将新的标签分配给该引用标签块并将新的绑定添加至 PSS(且可能是输出流)。无论哪种情况，现在都可以将第二级标签用于表达第一级引用标签的序列。然后将此新标签添加至成长的第二级引用标签序列。该编码过程检测该第二级引用序列，将内容导出分段应用于第二级引用标签序列，再一次确定是否有分段边界且如果有，为第二级标签的段生成一个第三级引用。增加地为下一分级重复该过程，在各情况下，“冒泡”新的对下一更高级的引用定义。以此方式，通过解码器的大块数据可能会多次通过该递归，然而小块数据不会经历不必要的编码级。

图 11 示出采用图 10 的不变段存储器的解码过程，其中各引用标签由其绑定的内容取代。如果该段的内容包括引用序列，则依次替换这些引用，且重复该过程直至所有的引用标签指向终端节点，然后输出该分段数据。即，解码器循环分解引用块，并当其到达数据分段时终止。在此实施例中，编码流等价于引用的分级树，其中标记树叶处的段(通过终端标记)来指示解码遍历应停在何处。

该分段模式的一个优点是可以有效地识别和压缩单个输入流中的共用图形。图 11 示出一个这样的例子。输入流包括分割成三段：S_1、S_2 和 S_3 的初始数据字节序列。向这些段分配了引用 R_1、R_2 和 R_3。在此例中，在其它符号中，余下的输入流包括重复两次以上的相同数据。因为将分段切割点定义成内容的函数，会在 PSS 中检测并找到到相同的段边界，因此会输出同一引用序列。类似地，假定分级编码确定引用<R_1、R_2、R_3>的序列为新的段定义一个切割点标示的 R_{10}，则可以将单个标签 R_{10} 再次用于标识输入流中的以后的重复序列。

HCS 使能客户机－服务器传输代理

图 12 示出可能用到这里所述的 HCS 编码的系统的例子。如其中所示，客户机 612 经过客户机端处理加速器("CTA")620 和服务器端处理加速器("STA")622 通过网络 616 耦合至服务器 614. 虽然仅示出一个客户机和一个服务器，但应理解可以用多个客户机和/或服务器很好地运行 CTA 和 STA。

客户机 612 与 CTA620 的客户机代理 630 相耦合。图 12 中所示的 CTA 620 的其它元件包括处理变换器(TT)632、反处理变换器(TT^{-1})634、不变段存储器(PSS)636 和引用分解器（RR）638。服务器 614 与 STA622 的服务器代理 640 相耦合，所示的代理包括与 CTA620 的那些相类似的元件(诸如处理变换器 642、反处理变换器(TT^{-1})644、不变段存储器(PSS)646 和引用分解器（RR）648)。

客户机 612 与客户机代理 630 耦合，该代理与 TT632 及 TT^{-1}634 相耦合。TT632 与 PSS636 相耦合并与 CTA620 和 STA622 之间的网络相耦合。TT^{-1}634 与 PSS636、客户机代理 630、RR638 相耦合并与 CTA620 和 STA622 之间的网络相耦合。所示的 RR638 也与 PSS636 相耦合并与 CTA620 和 STA622 之间的网络相耦合。

在图的另一边，服务器 614 与服务器代理 640 相耦合，该代理与 TT642 及 TT^{-1}644 相耦合。TT642 与 PSS646 相耦合并与 CTA620 和 STA622 之间的网络相耦合。TT^{-1}644 与 PSS646、服务器代理 640、RR648 相耦合并与 CTA620 和 STA622 之间的网络相耦合。所示的 RR648 也与 PSS646 相耦合并与 CTA620 和 STA622 之间的网络相耦合。

在所示的连接中，箭头指示信息流的最常见的一个或几个方向，但信息也可以流向其它方向且在单一方向上流动的信息也可能包括在相反方向上流动的数据。例如：TT632 通常在 TT^{-1}644 方向上发送信息，但它可能包括诸如确认、握手等从 TT^{-1}644 流向 TT632 的数据。

在操作中，CTA 和 STA 将它们的处理的有效载荷分段，其中担保的和存储/缓存字符串或其它用能不依赖处理的唯一命名模式从那些有效载荷中提取的数据序列("段")并当将有效载荷从一个 TA 发送至另一 TA 时，引用段替换分段数据，这时分段数据使发送器可预计接收器会具有该唯一命名的分段数据，或因为它出现在更早的处理中或通过其它处理到发送接收器。

例如：客户机－服务器处理可包括从服务器至客户机的文件数据的传送，虽然这只是服务器－服务器面向处理应用的一个特定情况。这里，客户机 612 可通过向服务器 614 发送文件打开请求从服务器 614 请求多个文件。这些请求会由客户机代理 630 代理，该客户机代理会与 STA622 相互作用，而 STA622 又通过其代理 640 代理对服务器 614 的请求。以此方式，可以以对其它实体

透明的方式(除了性能改善之外)加速客户机－服务器通信。客户机代理 630 将请求路由至 TT632，它作为文件打开请求很可能不会被编码。假定从服务器 614 的响应包含一个包括了请求文件的一部分的有效载荷。在一些系统中，当客户机对打开文件作出文件读取请求时，服务器首先响应具有短响应的文件请求消息并仅仅包括有效载荷。这里，我们假定服务器返回文件内容作为有效载荷。

当服务器代理 640 接收到响应消息及其有效载荷时，它将该数据传送至 TT642，TT642 如下所述对响应消息(或仅仅对其有效载荷)编码并当将其建立的任何绑定添加至 PSS646 中，如果它们不是现有的，并将编码的消息发送至 $TT^{-1}634$，$TT^{-1}634$ 重建该消息并将其通过客户机代理 630 中继至客户机 612。$TT^{-1}634$ 能重建消息，是因为它具有绑定以替换消息中的段引用。那些绑定来自 PSS636，且当它们不在 PSS636 中时，引用分解器 638 能从引用分解器 648 得到绑定。以此方式，可以编码消息且解码对消息发送器和接收器透明。

当各 TT 建立绑定时，它分配通用唯一引用标签，使引用标签的接收方一直能用明确的分段数据替换它。如 McCanne I 中所述，可以使用几种模式来确保唯一名称。

HCS 使能文件系统

本发明的另一实施例使用文件系统中的 HCS。通过合适地将 HCS 应用于文件系统的设计，可以通过识别文件系统中所有文件(各数据结构)中的重复数据的图形并相应地将任何给定数据图形仅仅在盘上存一次，可以实现容量的实质性的收益。一种以此形式杠杆作用文件系统中的 HCS 的方法是合并 HCS 编码器和 HCS 解码器作为本地的文件系统的本地元件。例如：一旦文件系统读取或写入盘存储块时，可以将 HCS 应用于这些盘存储块。然而，这一直接方法可引发性能问题，因为应用需要高性能文件 I/O 而 HCS 编码过程会引起计算消耗。又，当呈现了大的相邻的应用程序或文件数据区且文件文件系统的实施通常用较小的固定大小的块操作文件时 HCS 的工作最有效。

更好的方法是将 HCS 应用于采用整个文件访问而不是基于块的访问的文件系统。图 13 所示的系统用整个文件接口和杠杆作用 HCS 实施文件系统。即，客户机存储、检索并删除整个文件而不是文件的单个文件。当此方法通常不

支持标准文件系统接口时，它很适合其它文件系统的应用，例如：作为备份系统的基础。基于盘的在线备份系统，在存取比本地的文件系统慢但比传统备份系统(例如：基于磁带)快时，常称为“近线”存储器，以描述高性能在线文件系统和低性能离线备份系统之间的折衷。因此，将图 13 中所描述的文件系统称为“近线文件系统（NLFS）”。

在所示的模式中，客户机能存储、检索和删除整个文件。NLFS 运用 HCS 编码器、PSS 和 HCS 解码器来有效地存储文件，这样仅仅需要将各公共段的单个副本存储一次。另外，NLFS 运用数据库表格(称为"文件映射数据库(FMDB)")来将文件名映射至表示特定文件的一个或几个顶级标签。在一个实施例中，FMDB 包括一个时戳字段，使文件系统可包含同一文件的多个副本，在时间的不同点处表示该文件。

系统按以下工作。客户机通过将向接有文件名和文件数据的文件系统发送一个“存储”命令来存储或更新文件。然后文件系统将所有的文件数据发送至 HCS 解码器，该解码器返回一个或一个以上表示该存储的文件的标签。然后将那些标签输入 FMDB 作为表示新的或更新过的文件的元组的一部分。此元组中所包括的是相应的文件名和当前时戳(例如：从时钟提取的)。

为了检索一个文件，客户机将一个“检索”命令和文件名和可选择时戳一起发送到文件系统。该文件系统又用文件名作为关键字向 FMDB 发出一查询，检索所有包括所述文件名的元组。如果没有该元组，则向客户机返回一个错误。然后文件系统选择具有小于或等于指定时戳的最大时戳的元组。如果在命令中没有指定时戳，则选择具有最大时戳的元组。这具有选择存在请求的时间存储在系统中的文件的版本的效果。接着，文件系统将来自选择的元组的标签串发送至 HCS 解码器。HCS 解码器又将该标签解码成文件数据并将该数据返回至文件系统。然后文件系统将该文件数据返回至客户机。当然，可以使该过程流水线化，使文件系统能在解码器将解码的文件传送至文件系统时将文件数据传送至客户机。

为了从文件系统中删除一个文件，客户机将一个"删除"命令连同文件名和一可选择时间范围发送至文件系统，告知删除所有相对指定的时间范围存储的文件。相应地，NLFS 从 FMDB 查找并去除相应的条目并从 PSS 删除相应的标

签。

在一个实施例中，NLFS 元件布署成标准备份结构与通过 NDMP 之类的标准协议与客户机希望备份服务接口。

在另一实施例中，通过将上述 NLFS 与文件系统前端相结合，可以将 HCS 用作更标准的基于操作系统的文件系统的互补元件。此方法在图 14 中示出，图 14 示出 NLFS 和文件服务器前端。在此方法中，当一个或一个以上客户机打开一个特定文件时，文件服务器从 NLFS 系统读取该文件并将该文件复制到局部文件系统中。该文件被有效地缓存到局部前端文件系统中，该系统允许文件服务器采用普通文件系统协议和用于管理打开文件的算法。通过确保仅仅前端文件服务器与 NLFS 通信并通过采用普通文件系统锁定和一致性机制，存储在 NLFS 中的文件不可能与文件服务器的局部文件系统中的文件不一致。当所有客户机关闭该文件时，文件服务器将该文件写回到 NLFS 并释放局部文件服务器中的相应存储器。

根据熟知的文件系统设计和整个文件缓存的技术有许多建立前端文件服务器的方法，但在一个实施例中，将熟知的 unix 文件系统（UFS）扩展至与 NLFS 接口。例如：UFS(以及其它文件系统)利用目录将文件名映射至磁盘数据结构。在 UFS 中，文件点的目录项指向一个引用节点（或引用节）。因此可以用新的包括至 NLFS 中文件的引用的节点引用节类型扩展 UFS。在此模式下，当客户打开一个现有文件，UFS 检查新的引用节类型和当找到时从 NLFS 读取文件。然后 UFS 将返回的文件数据写入局部文件系统中，有效地用普通引用节点数据结构将格式化局部盘上的该文件数据。实际上，指向 NLFS 文件的单个引用节点由多个引用节点和磁盘存储块取代，以局部地表示实际文件。

在该点，客户机打开调用结束。此后，所有客户机文件动作一致进入传统 UFS 模式(包括多个客户机同时打开同一文件的情况)并且所有文件更新或改变都反映在局部文件副本中。当关闭文件时(由所有客户机)，修改的 UFS 从局部文件系统读取文件数据(使用正常引用节点数据结构)并将该文件写回到 NLFS。当连续地写入文件时，可以释放表示该文件数据的引用节点并用用指向 NLFS 的新的类型的单个节点将其替换。另选地，可以用该形式将该文件从 NLFS 复制出来而不需要释放文件的本地副本，从而未来对该文件的访问不需

要等待 NLFS。然后，当局部文件系统开始装满容量时，UFS 可以用上述方法通过应用多个熟知缓冲替换策略(例如：最近使用的、最常访问的等)来释放文件中的任意个。

上述说明是示例性的而非限定性的。在读过此公开后，许多本发明的变化将对那些本技术领域的技术人员变得很明显。因此，本发明的范围不应对照以上说明来确定而应对照所附权利要求及它们的完全等效物来确定。

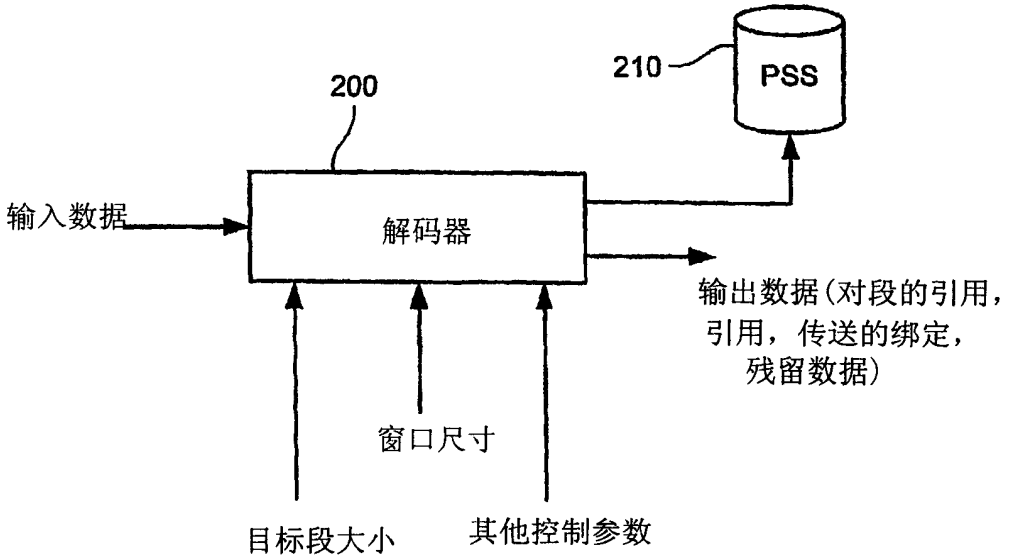

210
PSS
200
输入数据
解码器
输出数据(对段的引用，
引用，传送的绑定，
残留数据)
窗口尺寸
目标段大小
其他控制参数

图 1

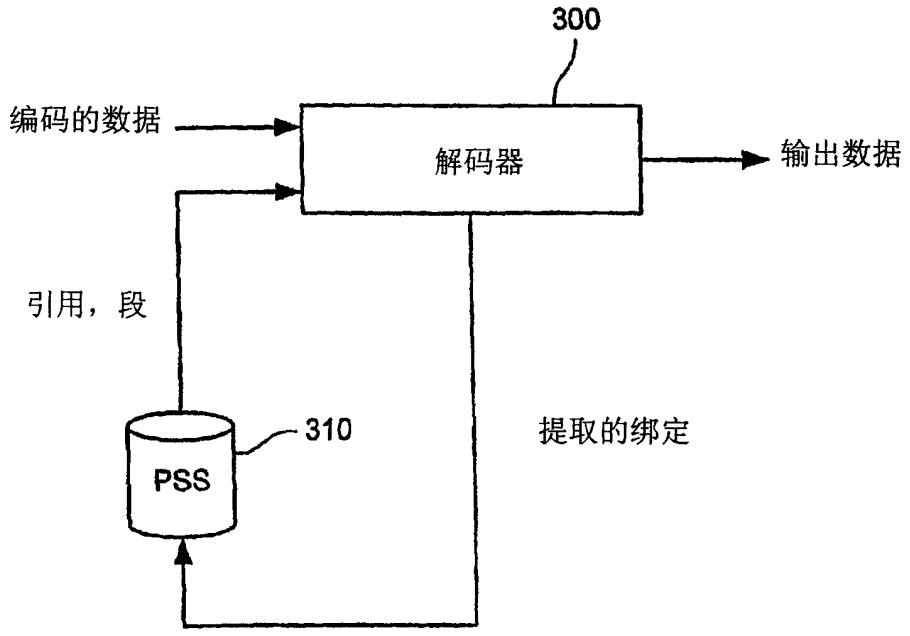

300
编码的数据
解码器
输出数据
引用，段
310
PSS
提取的绑定

图 2

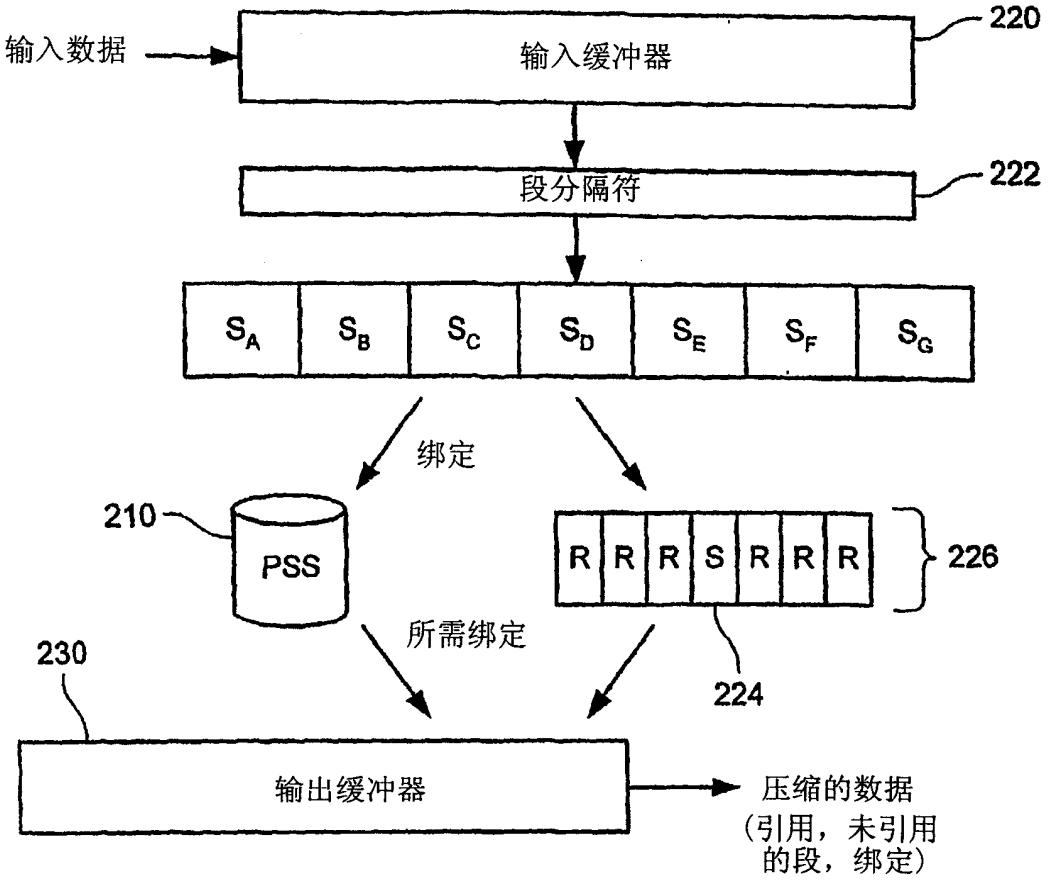

输入数据
输入缓冲器
220
段分隔符
222
S_A S_B S_C S_D S_E S_F S_G
绑定
210
PSS
R R R S R R R
226
所需绑定
224
230
输出缓冲器
压缩的数据
(引用，未引用的段，绑定)

图 3

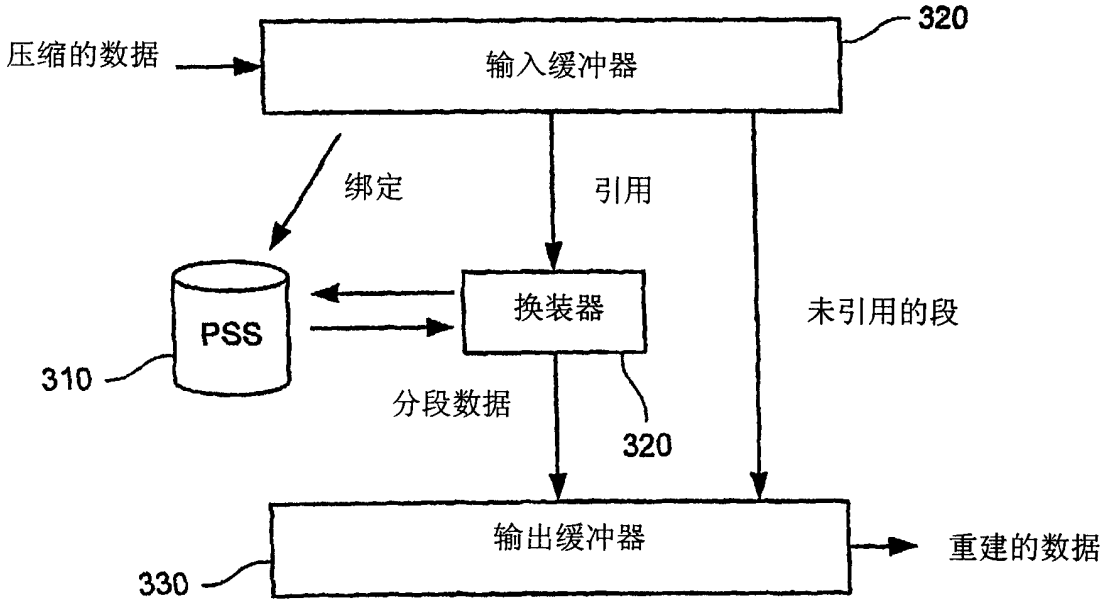

压缩的数据
输入缓冲器
320
绑定
引用
PSS
310
换装器
未引用的段
分段数据
320
输出缓冲器
330
重建的数据

图 4

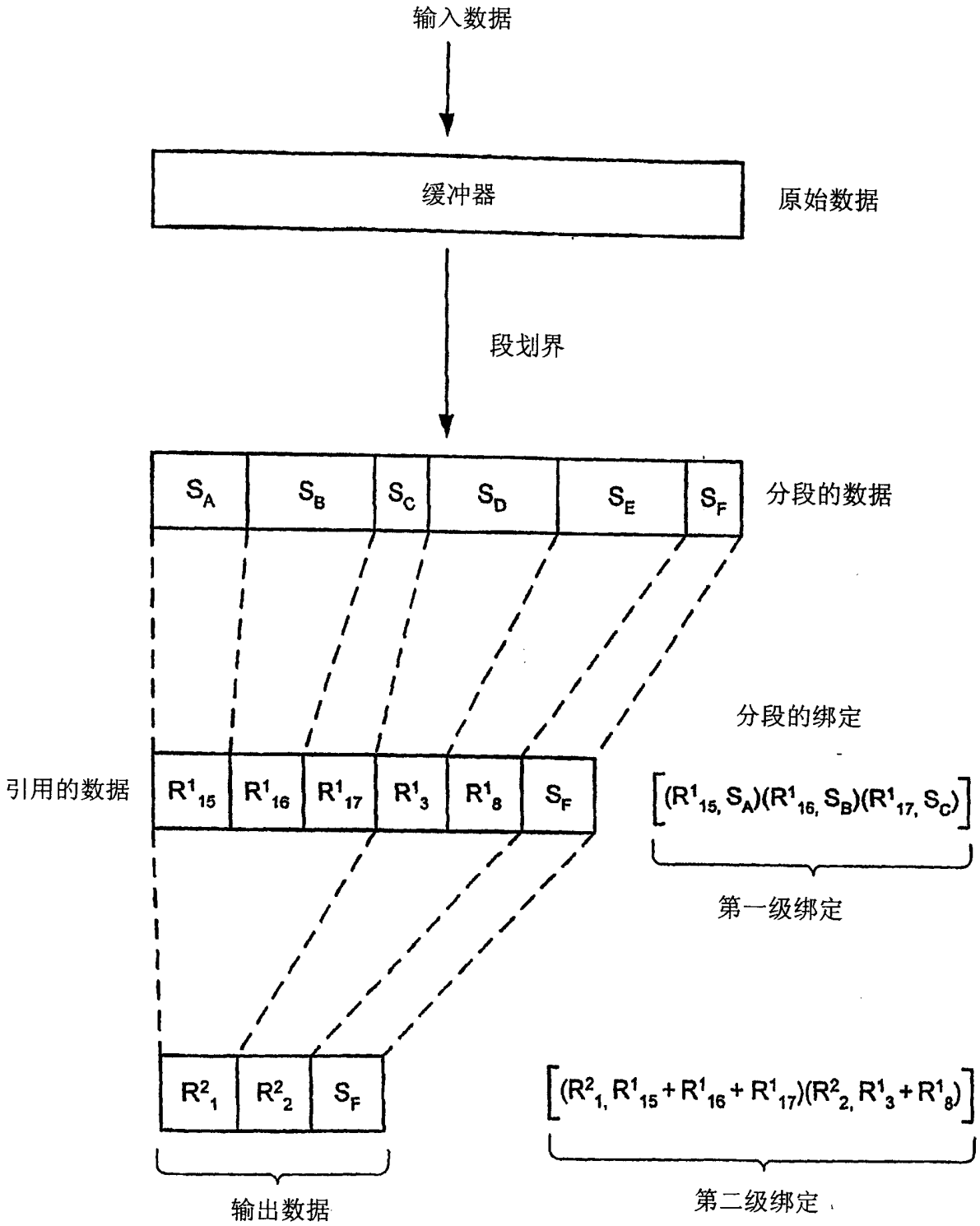

输入数据
缓冲器
原始数据
段划界
S_A S_B S_C S_D S_E S_F
分段的数据
分段的绑定
引用的数据
R^1_{15} R^1_{16} R^1_{17} R^1_3 R^1_8 S_F
$[(R^1_{15}, S_A)(R^1_{16}, S_B)(R^1_{17}, S_C)]$
第一级绑定
R^2_1 R^2_2 S_F
$[(R^2_1, R^1_{15}+R^1_{16}+R^1_{17})(R^2_2, R^1_3+R^1_8)]$
输出数据
第二级绑定

图 5

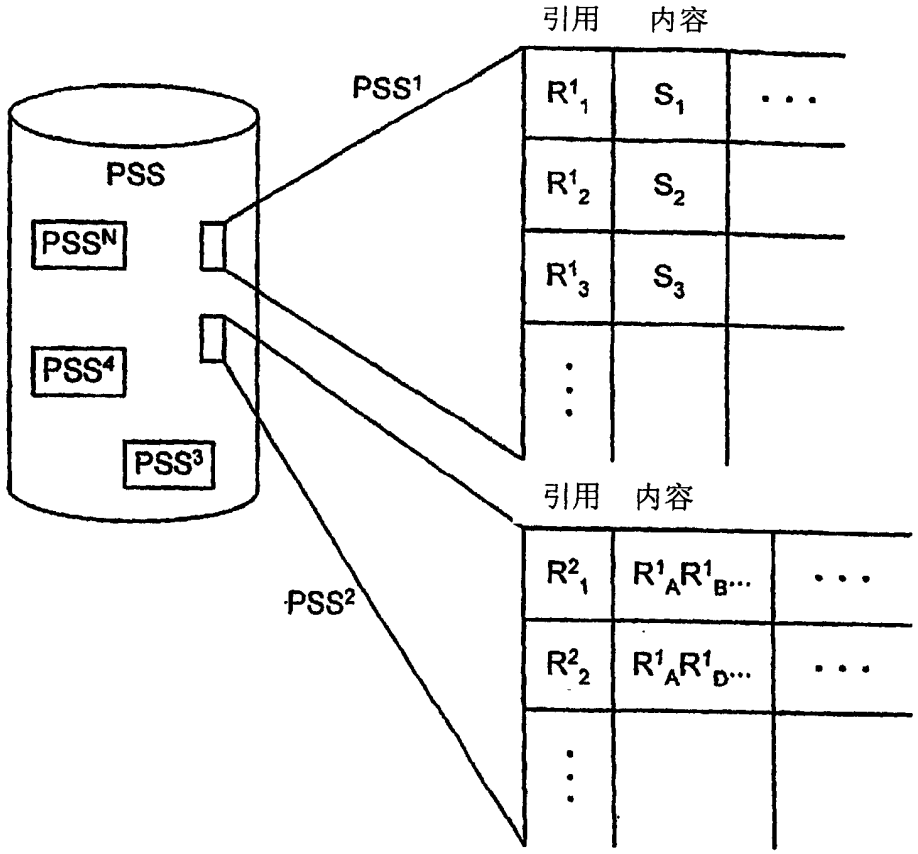

引用 内容
PSS1
PSS
PSSN
PSS4
PSS3
R1 1
S1
R1 2
S2
R1 3
S3
引用 内容
PSS2
R2 1
R1 AR1 B...
R2 2
R1 AR1 D...

图 6

太少的编码级

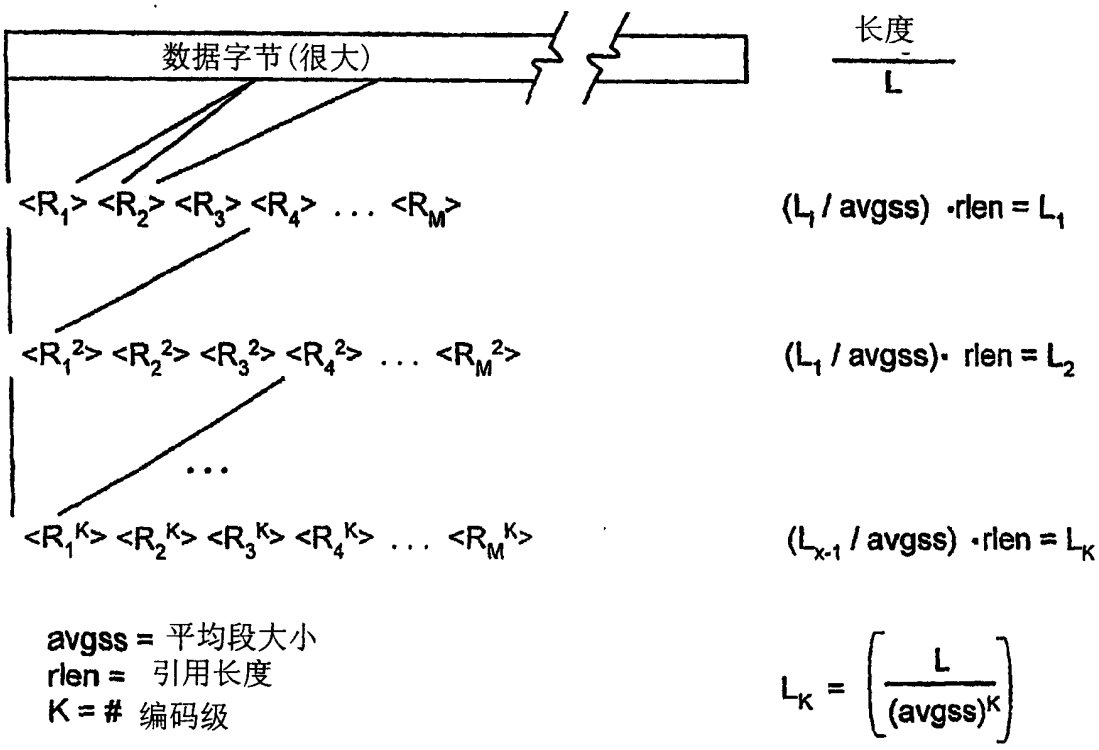

数据字节(很大)
长度
L
<R1> <R2> <R3> <R4> ... <RM>
(L1 / avgss) ·rlen = L1
<R1 2> <R2 2> <R3 2> <R4 2> ... <RM 2>
(L1 / avgss)· rlen = L2
<R1 K> <R2 K> <R3 K> <R4 K> ... <RM K>
(Lx-1 / avgss) ·rlen = LK
avgss = 平均段大小
rlen = 引用长度
K = # 编码级
LK = ⌈ L / (avgss)K ⌉

图 7

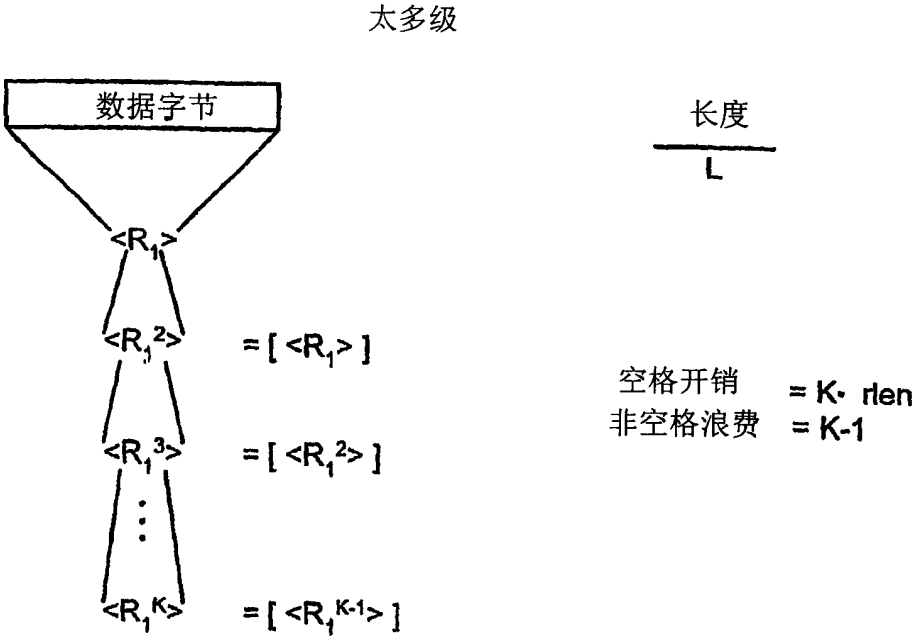

太多级
数据字节
长度
L
<R1>
<R1 2>
= [<R1>]
<R1 3>
= [<R1 2>]
<R1 K>
= [<R1 K-1>]
空格开销 = K· rlen
非空格浪费 = K-1

图 8

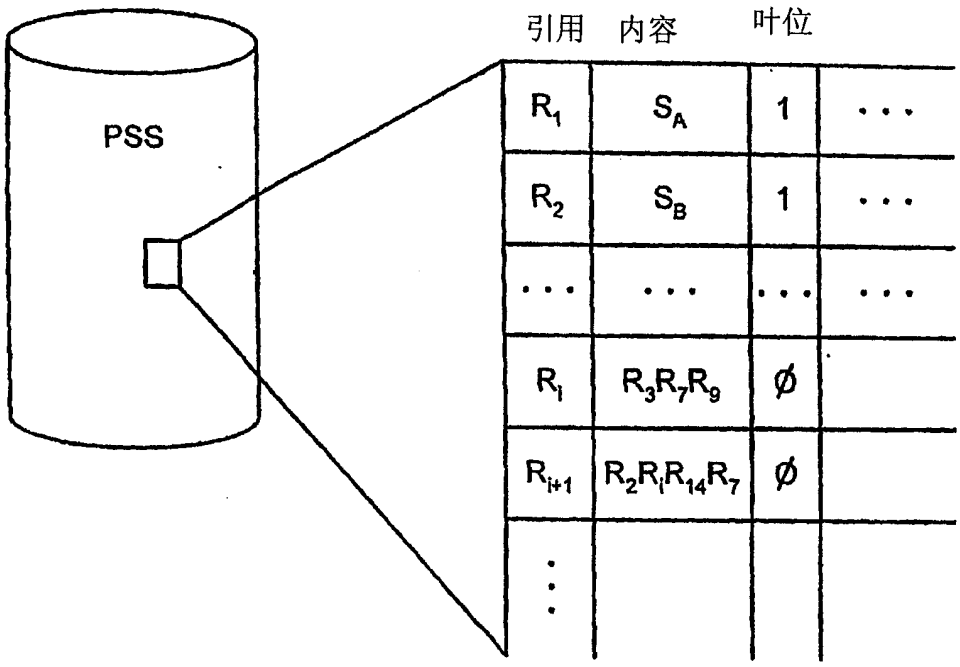

PSS
引用
内容
叶位
R1
SA
1
R2
SB
1
Ri
R3R7R9
Ø
Ri+1
R2RiR14R7
Ø

图 9

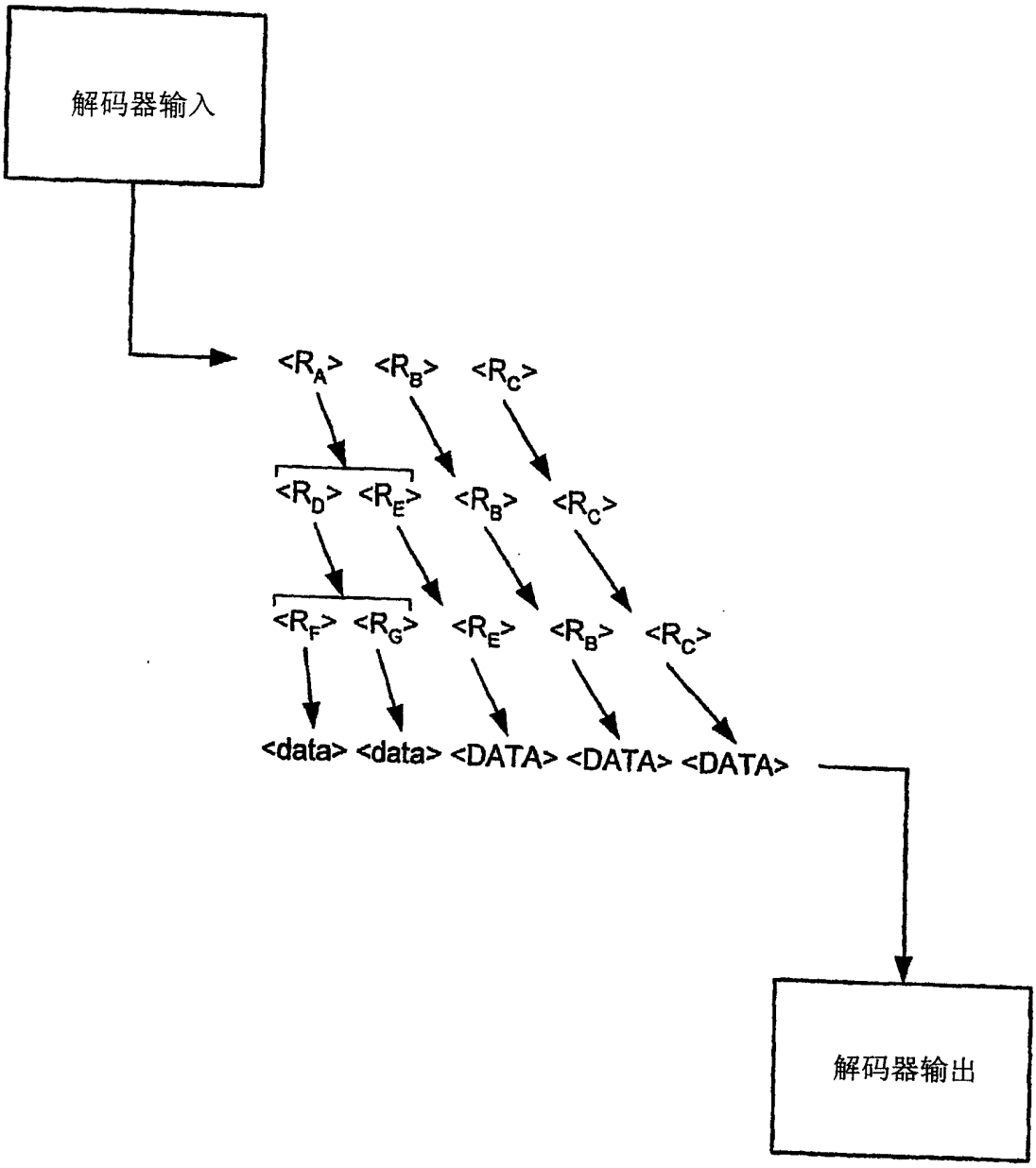

解码器输入
<R_A> <R_B> <R_C>
<R_D> <R_E> <R_B> <R_C>
<R_F> <R_G> <R_E> <R_B> <R_C>
<data> <data> <DATA> <DATA> <DATA>
解码器输出

图 10

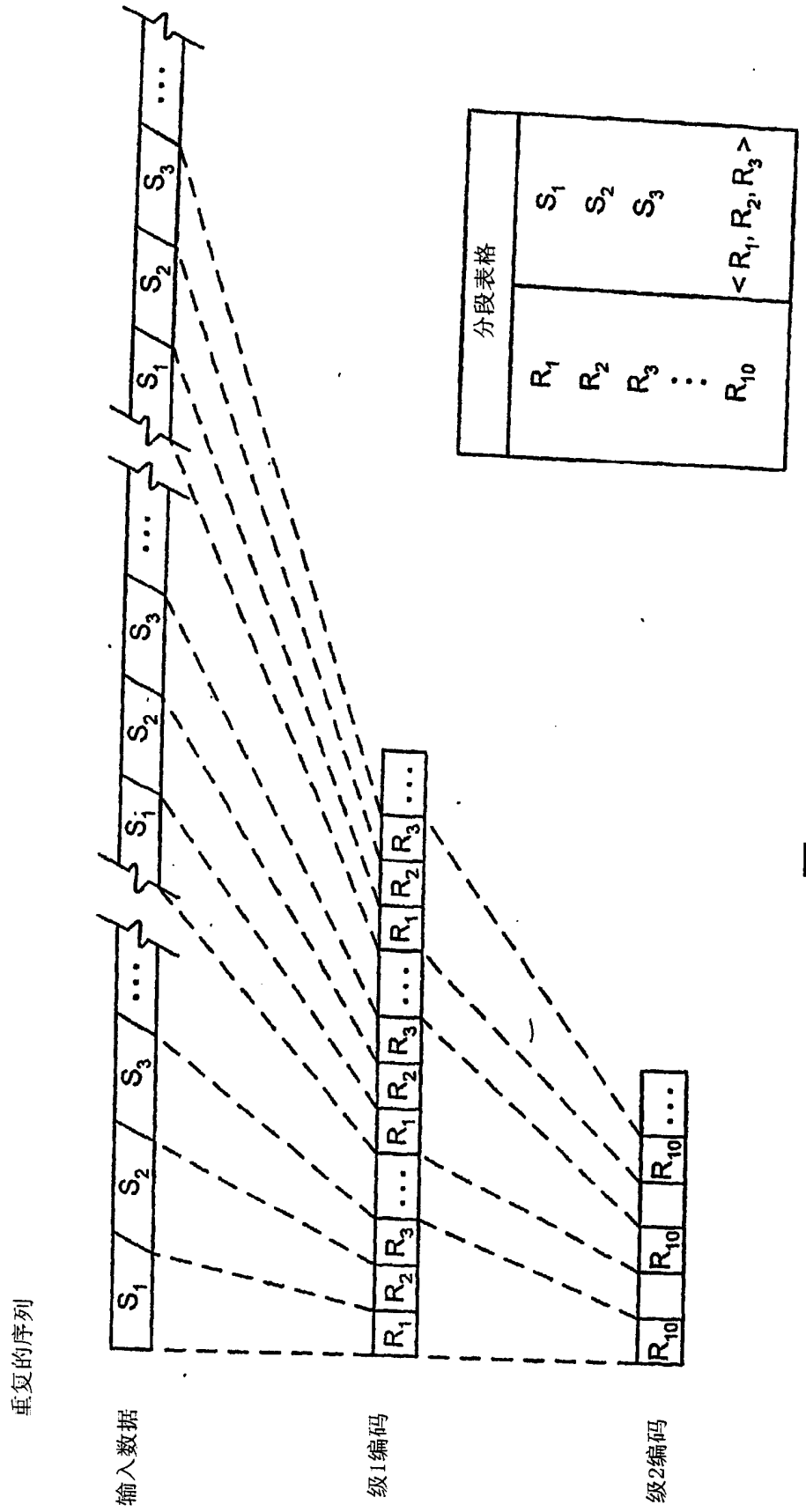


图 11

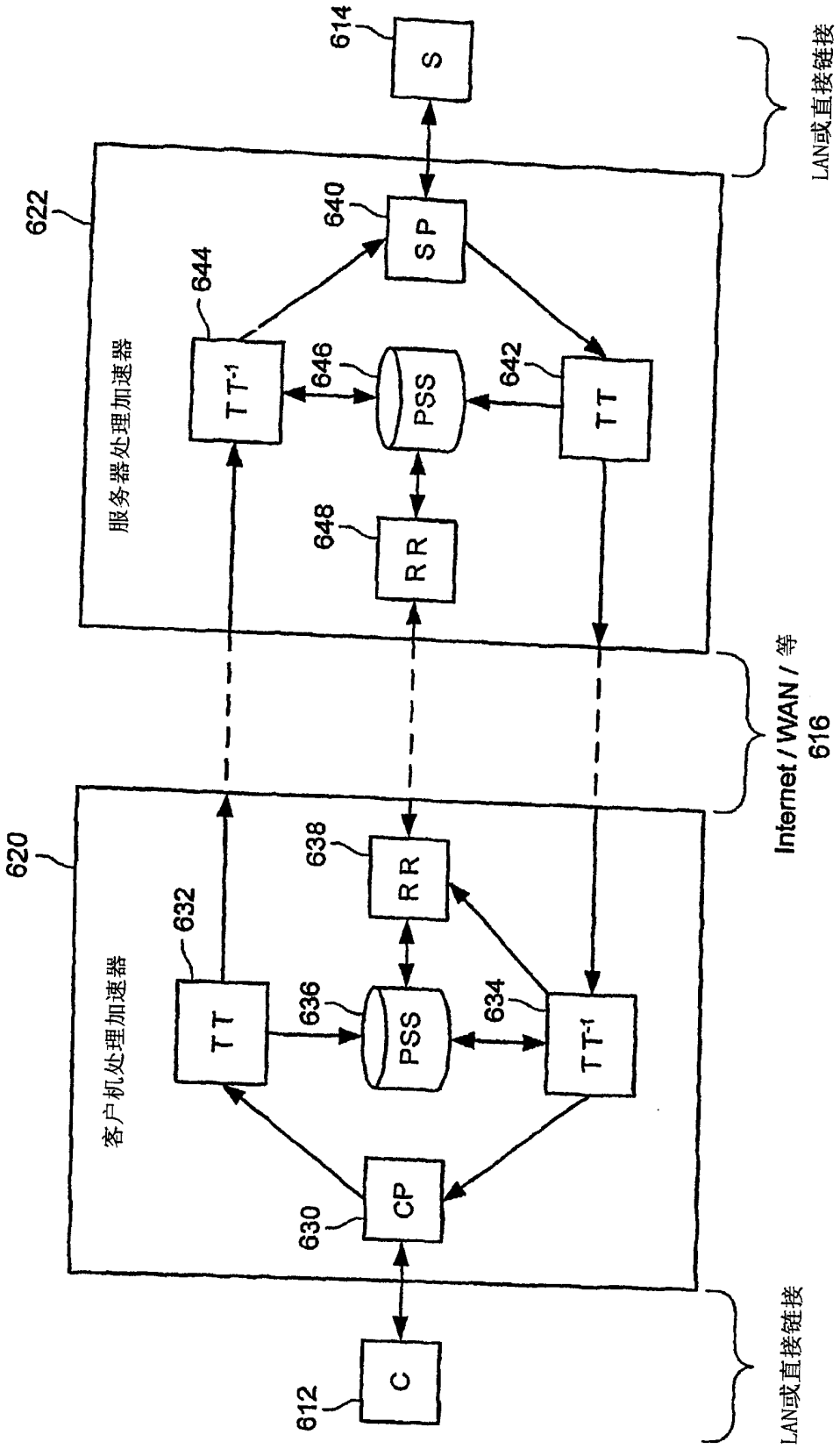

614
S
622
640
SP
644
TT-1
646
PSS
642
TT
648
RR
服务器处理加速器
LAN或直接链接
Internet / WAN / 等
616
620
638
RR
632
TT
636
PSS
634
TT-1
客户机处理加速器
630
CP
612
C
LAN或直接链接

图 12

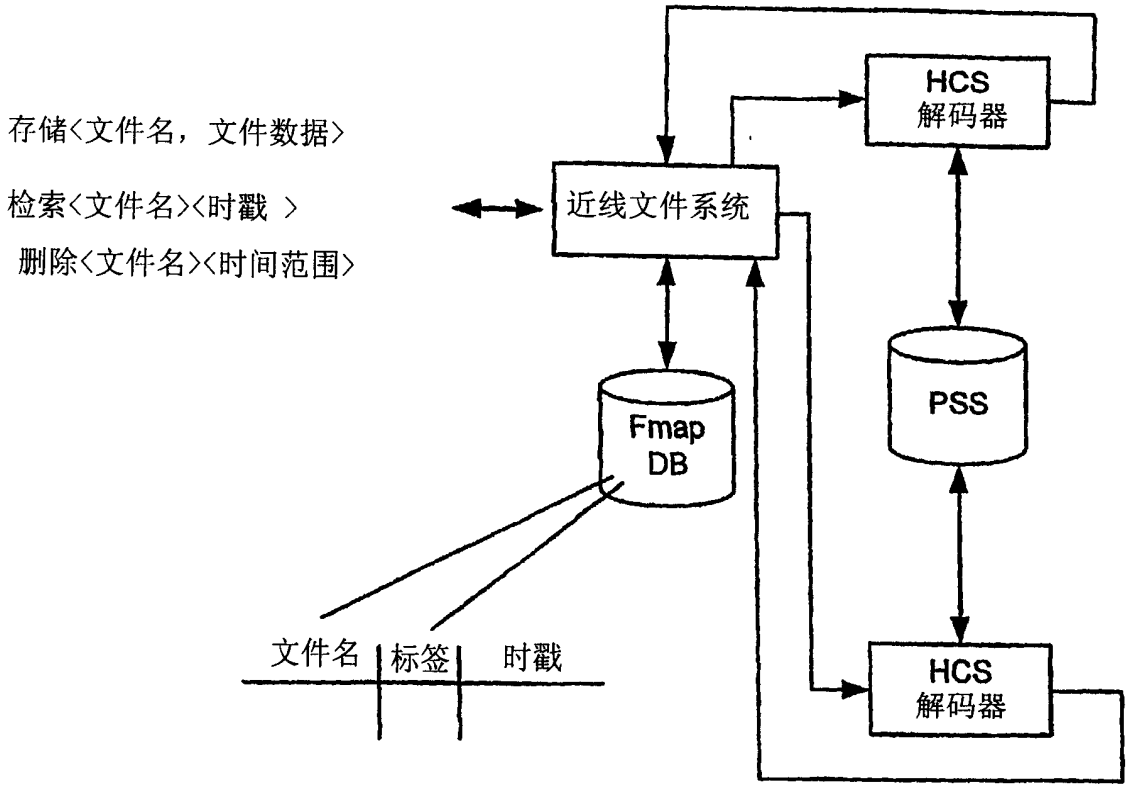

存储<文件名，文件数据>
检索<文件名><时戳 >
删除<文件名><时间范围>
HCS
解码器
近线文件系统
PSS
Fmap
DB
文件名
标签
时戳
HCS
解码器

图 13

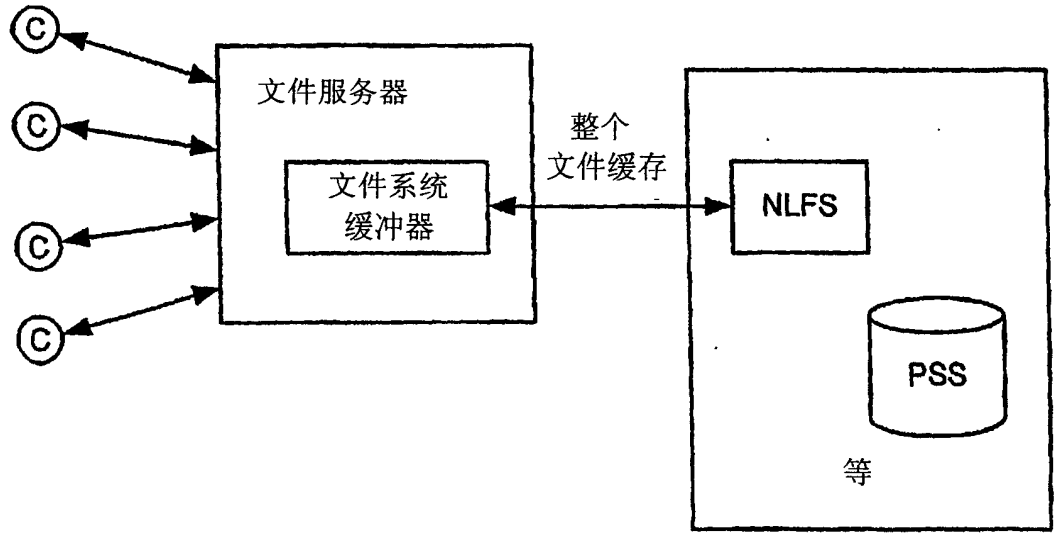

C
C
C
C
文件服务器
文件系统
缓冲器
整个
文件缓存
NLFS
PSS
等

图 14