[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/45 (2006.01)

[12] 发明专利说明书

专利号 ZL 200510077097.7

[45] 授权公告日 2010 年 1 月 20 日

[11] 授权公告号 CN 100583040C

[22] 申请日 2005.6.15
[21] 申请号 200510077097.7
[30] 优先权
[32] 2004.8.30 [33] US [31] 10/930,037
[73] 专利权人 国际商业机器公司
地址 美国纽约阿芒克
[72] 发明人 崔世民 罗奇·乔治斯·阿香博
劳尔·埃斯特班·西尔弗拉 高耀清
[56] 参考文献
US6173421B1 2001.1.9
US6675378B1 2004.1.6
US5701470A 1997.12.23
US6286135B1 2001.9.4
US6779186B2 2004.8.17
审查员 徐 春

[74] 专利代理机构 北京市金杜律师事务所
代理人 朱波海

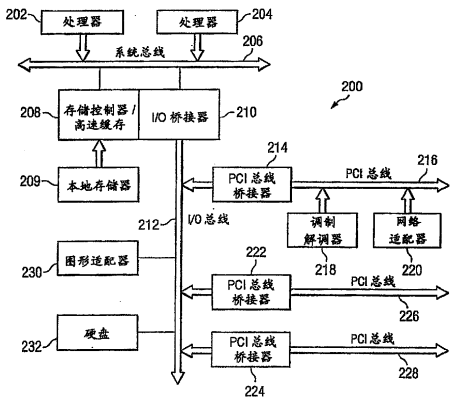
权利要求书 5 页 说明书 18 页 附图 9 页

[54] 发明名称

改善数据高速缓存性能的方法和设备

[57] 摘要

本发明的机制提供了过程间的强度缩量以改善数据高速缓存的性能。 在正推法期间，本发明收集全局变量的信息并且分析全局对象的使用方式以便选择用于优化的候选计算。 在逆推法期间本发明将全局对象重新映射到较小大小的新的全局对象并且通过用较小的全局对象的间接的或索引的引用来代替候选计算，并且插入引用候选全局对象的每个计算的新全局对象的存储操作，产生高速缓存效率更高的代码。



1. 一种在数据处理系统中用于改善数据高速缓存性能的方法，该方法包括:

在正推法期间在软件程序中收集至少一个全局对象的信息，其中所述全局对象是全局动态对象和全局阵列之一;

在所述正推法之后分析所收集的信息以形成分析信息; 并且

使用所述分析信息在逆推法期间修改所述软件程序的中间格式文件，

其中，所述收集步骤包括:

识别至少一个用于优化的候选全局对象; 以及

收集至少一个计算，所述至少一个计算在所述至少一个候选全局对象上运行，

并且其中，所述分析步骤包括:

根据所述至少一个计算中的哪个计算具有比当前计算更低的预计算成本，来从所述至少一个计算中选择至少一个候选计算。

2. 如权利要求 1 的方法，其中所述收集步骤还包括:

使用转义分析来分析所述至少一个全局对象的别名; 以及

收集至少一个存储操作，所述至少一个存储操作修改所述至少一个候选全局对象。

3. 如权利要求 1 的方法，其中所述分析步骤还包括:

根据别名信息来完成至少一个候选全局对象; 以及

创建所述选择的候选计算的至少一个全局变量，其中所述至少一个候选计算和所述至少一个全局变量是所述分析信息的一部分。

4. 如权利要求 3 的方法，其中为所述所选的候选计算而产生的全局变量的数目依赖于与所述至少一个候选全局对象相关联的间接的和索引的引用之一的级别的数量。

5. 如权利要求 3 的方法，其中该修改步骤包括:

由所述至少一个全局变量的间接的或索引的引用之一来代替所述所选的候选计算；以及

对于每个所述至少一个候选全局对象，在所述软件程序中的所述至少一个候选全局对象的所有定义点处插入存储操作。

6. 如权利要求 2 的方法，其中所述识别步骤包括：

排除可以在未定义的过程中修改的全局对象；

确定能否识别所述至少一个全局对象的准确别名；

如果能识别所述至少一个全局对象的准确别名，则分析所述至少一个全局对象的大小；以及

如果所述至少一个全局对象的大小为大，则选择所述至少一个全局对象。

7. 如权利要求 6 的方法，其中用于对至少一个全局对象的大小进行分析的分析步骤是通过使用从所述软件程序的静态数据流和运行时间配置文件之一所得到的信息来进行的。

8. 如权利要求 2 的方法，其中所述收集至少一个计算以及至少一个存储操作包括：

确定计算是否只包括全局对象；

如果该计算只包括全局对象，则确定所述计算是否能够映射到较小大小的数据类型的简单装载；并且

如果所述计算能够映射到较小大小的数据类型的简单装载，则将排除索引信息的计算添加到使用列表。

9. 如权利要求 8 的方法，还包括：

确定所述至少一个候选全局对象是否是全局对象；以及

如果所述至少一个候选全局对象是全局对象，则将间接和索引的引用之一的所有级别添加到定义列表。

10. 如权利要求 9 的方法，还包括：

如果所述至少一个候选全局对象不是全局对象，则确定所述至少一个候选全局对象是否是全局阵列；并且

如果所述至少一个候选全局对象是全局阵列，则将索引引用符号

添加到定义列表。

11. 如权利要求 8 的方法，其中用以确定所述计算能否映射到较小大小的数据类型的简单装载的所述确定步骤包括:

确定由所述计算引用的所有全局对象的值是否在范围内；以及

如果由所述计算引用的所有全局对象的值在范围内，则选择所述计算。

12. 如权利要求 11 的方法，还包括:

如果由所述计算引用的所有全局对象的值不在范围内，则确定该计算是否是转换表达式，其中所述转换表达式是将包含所述至少一个全局对象的表达式的类型和精度之一转换为较小的数据类型；并且

如果所述计算是转换成较小数据类型的转换表达式，则选择所述计算。

13. 如权利要求 12 的方法，还包括:

如果所述计算不是转换表达式，则确定所述计算是否是比较表达式；并且

如果所述计算是比较表达式，则选择该计算。

14. 如权利要求 13 的方法，还包括:

如果所述计算不是比较表达式，则确定所述计算是否没有以其他方式用于所述计算的同一循环和控制流区域之一中；以及

如果所述计算没有以其他方式用于所述计算的同一循环和控制流区域之一中，则选择该计算。

15. 如权利要求 3 的方法，其中所述选择步骤包括:

确定计算的预计算的总成本；以及

如果由所述计算引用的所述至少一个候选全局对象大小为大并且所述计算的执行成本比由所述计算引用的所述至少一个候选全局对象的执行成本实质上要高，则选择该计算。

16. 如权利要求 5 的方法，其中所述插入步骤包括:

如果所述选择的候选计算是对所述至少一个候选全局对象的间

接的和索引的引用之一的最高级别的存储操作，则插入至少一个候选全局变量计算值的间接的和索引的引用之一的最高级别的存储操作。

17. 如权利要求 16 的方法，还包括:

如果所述选择的候选计算对所述至少一个候选全局动态对象的间接引用的第二高级别分配存储，则插入存储操作以向所述至少一个全局变量的间接引用的第二高级别分配较小大小的存储。

18. 如权利要求 17 的方法，还包括:

如果所述选择的候选计算对所述至少一个候选全局动态对象的间接引用指针的基础指针或较低级别分配存储，则插入存储操作以向所述至少一个全局变量的间接引用指针的基础指针或较低级别分配同样大小的存储。

19. 如权利要求 18 的方法，还包括:

如果所述选择的候选计算为所述至少一个候选全局对象的指针赋值，则通过用所述至少一个全局变量及其相应的别名代替所述选择的候选计算来插入所述至少一个全局变量的存储操作。

20. 一种用于改善数据高速缓存性能的数据处理系统，该数据处理系统包括:

收集装置，用于在正推法期间在软件程序中收集至少一个全局对象的信息，其中所述全局对象是全局动态对象和全局阵列之一;

分析装置，用于在所述正推法之后分析所收集的信息以形成分析的信息; 以及

修改装置，用于使用所述分析的信息在逆推法期间修改所述软件程序的中间格式文件，

其中，所述收集装置包括:

识别设备，用来识别至少一个用于优化的候选全局对象; 以及

收集设备，用来收集至少一个计算，所述至少一个计算在所述至少一个候选全局对象上运行，

并且其中，所述分析装置包括：

选择设备，用来根据所述至少一个计算中的哪个计算具有比当前计算更低的预计算成本，来从所述至少一个计算中选择至少一个候选计算。

21. 如权利要求 20 的数据处理系统，其中所述收集装置还包括：

分析设备，用来使用转义分析来分析所述至少一个全局对象的别名；

并且其中，所述收集设备还用来收集至少一个存储操作，所述至少一个存储操作修改所述至少一个候选全局对象。

22. 如权利要求 20 的数据处理系统，其中所述分析装置还包括：

完成设备，用于根据别名信息来完成至少一个候选全局对象；以及

创建设备，用于对于所选的候选计算创建至少一个全局变量，其中所述至少一个候选计算和所述至少一个全局变量是所述分析信息的一部分。

23. 如权利要求 21 的数据处理系统，其中所述收集设备包括：

第一确定装置，用于确定计算是否只包括全局对象；

第二确定装置，如果该计算只包括全局对象，则该第二确定装置确定所述计算是否能够映射到较小大小的数据类型的简单装载；并且

添加装置，如果所述计算能够映射到较小大小的数据类型的简单装载，则该添加装置将排除索引信息的所述计算添加到使用列表。

24. 如权利要求 22 的数据处理系统，其中所述选择设备包括：

确定装置，用于确定计算的预计算的总成本；以及

选择装置，如果由所述计算引用的所述至少一个候选全局对象的大小为大并且所述计算的执行成本实质上高于由所述计算引用的所述至少一个候选全局对象的执行成本，则该选择装置选择该计算。

改善数据高速缓存性能的方法和设备

相关申请的交叉引用

本发明涉及名为“使用过程间强度缩量优化软件程序的方法和设备”的申请，代理案号为CA920040084US1，其与本申请同时提交，转让与同一受让人，且通过参考引入本申请。

技术领域

本发明涉及一种改善的数据处理系统。特别地，本发明涉及改善数据处理系统的数据高速缓存的性能。更特别地，本发明涉及使用全局对象的过程间强度缩量来改善数据高速缓存的性能。

背景技术

在数据处理系统中，数据高速缓存是连接处理器和主存储器的高速缓存的一部分。每当请求例如变量值的一段数据时，处理器在主存储器中进行查找之前会将该数据定位在数据高速缓存上。这样，数据高速缓存存储处理器最常检索的数据。

然而，随着处理器速度的增加，由于处理器处理请求的速度远快于数据随机存取存储器（DRAM）的处理速度，因而DRAM或数据高速缓存的延迟也增加了。这使得改善数据高速缓存的性能成为现代编译器设计的优先考虑。

已经介绍了用以优化全局数据定位的两种主要方法，其依次改善了数据高速缓存的性能。一种方法使用了采用空间定位的编码转换。在编译器的编码优化阶段期间执行该方法。通常的编码转换包括循环分片，条状开采和循环交换。另一个方法是使用采用了数据再利用的数据布局转换。通常的布局转换包括阵列填充和数据重建。

如通过参考引入本申请的，名为“使用过程间强度缩量来优化软

件程序的方法和设备”的相关专利申请中所提到的，成本高的计算可以被识别并代之以成本较低的计算。这样，软件程序中的计算总量就在过程间得到了减少并且软件程序的性能得到了提高。

然而，在程序的给定过程间角度需要一种改善数据高速缓存效用的解决方案。因此，具有一个能够解析软件程序中的全局对象的使用方式并且识别对象压缩时机的改善的方法和设备是很有利的，这样就能够产生高速缓存效率更高的代码。

发明内容

提供一种方法，设备和计算机指令以使用全局对象的过程间强度缩量来改善数据高速缓存的性能。本发明的机制使用在正推法期间收集的信息分析源程序中全局对象的使用方式并且确定全局对象能否被映射到大小较小的全局对象。一旦识别出候选计算，本发明的机制就在逆推法期间修改源程序的中间代码从而用大小较小的新的全局对象的间接或索引的引用来代替候选计算并且对于每个引用候选全局变量的计算向新的全局对象插入存储操作。

附图说明

在所附的权利要求书中陈述了认为是本发明特点的新颖特征。然而最好通过联系附图参考下述对于说明性实施例的详细描述，理解发明本身、优选实施方式以及进一步的目的和优点。其中：

图 1 描述了能够实施本发明的数据处理系统的网络的示意图；

图 2 是数据处理系统的方框图，根据本发明的优选实施例其能够用作服务器；

图 3 是说明可以实施本发明的数据处理系统的方框图；

图 4 是说明在本发明的优选实施例中源程序、编译器和机器语言指令之间的关系的方框图；

图 5 是根据本发明的优选实施例使用过程间强度缩量优化软件程序的处理的流程图；

图 6A 是根据本发明的优选实施例的在正推法期间收集信息的处理的流程图；

图 6B 是根据本发明的优选实施例的使用过程间的别名分析来识别候选全局对象的处理的流程图；

图 6C 是根据本发明的优选实施例收集全局对象的存储操作以及全局对象的计算的处理的流程图；

图 6D 是根据本发明的优选实施例的确定一个计算能否映射到较小数据类型的简单加载的处理的流程图；

图 7A 是根据本发明的优选实施例分析收集的信息的处理的流程图；

图 7B 是根据本发明的优选实施例的根据成本分析为强度缩量选择候选计算以及为候选计算创建全局变量的处理的流程图；

图 8A 是根据本发明的优选实施例在逆推法期间修改代码的中间表达的处理的流程图；

图 8B 是根据本发明的优选实施例对每个引用全局变量的计算插入存储操作的处理的流程图；

图 9 是说明根据本发明的优选实施例在使用过程间强度缩量的优化之前的示例性软件程序的示意图；

图 10 是说明根据本发明的优选实施例在使用过程间强度缩量的优化之后的示例性软件程序的示意图；

图 11 是说明根据本发明的优选实施例在使用过程间强度缩量的进一步优化之后的示例性软件程序的示意图；

具体实施方式

现在参考附图，图 1 描述了能够实施本发明的数据处理系统的网络的图示。网络数据处理系统 100 是其中可以实施本发明的计算机网络。网络数据处理系统 100 包括网络 102，用以提供在网络数据处理系统 100 中连接在一起的各种设备和计算机之间的通信链路。网络 102 可以包括例如有线、无线通信链路或光缆的连接。

在所描述的例子中，服务器 104 与存储单元 106 一起连接到网络 102。另外，客户 108，110 和 112 连接到网络 102 上。这些客户 108、110 和 112 可以是，例如，个人计算机或网络计算机。在所描述的例子中，服务器 104 提供例如引导文件、操作系统图像和应用程序的数据到客户 108-112。客户 108、110 和 112 是服务器 104 的客户。网络数据处理系统 100 可以包括未示出的附加的服务器、客户及其他设备。在所描述的例子中，网络数据处理系统 100 是具有代表着全球网络和网关，并使用传输控制协议/互联网协议（TCP/IP）协议组进行互相通信的网络 102 的互联网。互联网的中心是主节点或主计算机之间的高速数据通信线的中枢，这些主节点或主计算机由成千上万个发送数据和信息的商业的、政府的、教育的以及其他的计算机系统组成。当然，网络数据处理系统 100 也可以由多个不同类型的例如内联网、局域网（LAN）或广域网（WAN）的网络来实现。图 1 意图作为例子而不是作为本发明的结构限制。

参考图 2，根据本发明的优选实施例描述了其能够实现为例如图 1 中的服务器 104 的服务器的数据处理系统的方框图。数据处理系统 200 可以是对称多处理器（SMP）系统，该系统包括连接到系统总线 206 的多个处理器 202 和 204。可选择地，可以使用单个处理器系统。存储控制器/高速缓存 208 也连接到系统总线 206，208 提供到本地存储器 209 的接口。I/O 总线桥接器 210 连接到系统总线 206 并且提供到 I/O 总线 212 的接口。存储控制器/高速缓存 208 和 I/O 总线桥接器 210 可以如描述的那样集成在一起。

连接到 I/O 总线 212 的外围部件互连（PCI）总线桥接器 214 提供到 PCI 本地总线 216 的接口。多个调制解调器可以连接到 PCI 本地总线 216。典型的 PCI 总线的实现将支持四个 PCI 扩展槽或附加连接器。图 1 中到客户 108-112 的通信链路可通过调制解调器 218 和通过附加连接器连接到 PCI 本地总线 216 的网络适配器 220 来提供。

附加的 PCI 总线桥接器 222 和 224 向附加的 PCI 本地总线 226 和 228 提供接口，通过其可以支持附加的调制解调器或网络适配器。

以这种方式，数据处理系统 200 允许到多网络计算机的连接。如图所示存储器映射图形适配器 230 和硬盘 232 也可以直接或间接地连接到 I/O 总线 212。

本领域普通技术人员将理解图 2 中描述的硬件可以变化。例如，例如光盘驱动器等的外围设备也可以附加于或代替所述的硬件来使用。所述例子不意味着对于本发明的结构限制。

图 2 中描述的数据处理系统可以是例如 IBM eServer™ pSeries® 系统，该系统是位于纽约阿蒙德的美国国际商用机器公司的产品，运行着高级交互执行程序(AIX®) 操作系统或 LINUX™ 操作系统。

IBM, eServer, Pseries 和 AIX 是美国国际商用机器公司的商标或注册商标。LINUX 是 Linus Torvalds 的商标。

现在参考图 3，其为描述了可以实施本发明的数据处理系统的方框图。数据处理系统 300 是客户计算机的例子。数据处理系统 300 采用外围部件互连（PCI）本地总线结构。虽然所述的例子采用了 PCI 总线，但是也可以使用例如图形加速端口（AGP）和工业标准结构（ISA）的其他总线结构。处理器 302 和主存储器 304 通过 PCI 桥接器 308 连接到 PCI 本地总线 306。PCI 桥接器 308 也可以包括对于处理器 302 的集成存储控制器和高速缓存。通过直接元件互连或通过附加板可以形成到 PCI 本地总线 306 的附加连接。在所述的例子中，局域网（LAN）适配器 310、SCSI 主机总线适配器 312 和扩展总线接口 314 通过直接元件连接而连接到 PCI 本地总线 306。相反，音频适配器 316、图形适配器 318 以及音频/视频适配器 319 通过插入到扩展槽的附加板而连接到 PCI 本地总线 306。扩展总线接口 314 为键盘和鼠标适配器 320、调制解调器 322 和附加的存储器 324 提供连接。小型计算机系统接口（SCSI）主机总线适配器 312 为硬盘驱动器 326、磁带驱动器 328 和 CD-ROM 驱动器 330 提供连接。典型的 PCI 本地总线的实现将支持三或四个 PCI 扩展槽或附加连接器。

操作系统运行在处理器 302 上并且用于对图 3 中的数据处理系统 300 内的各种元件进行协调并提供控制。该操作系统可以是商用的操

作系统，例如从微软公司可得到的 Windows™ XP。Windows 是微软公司的商标。例如 Java 的面向对象的程序设计系统可以与操作系统一起运行并且从在数据处理系统 300 上执行的 Java 程序或应用向操作系统提供调用。“Java”是 Sun 微系统公司的商标。操作系统、面向对象的程序设计系统以及应用程序或程序的指令位于例如硬盘驱动器 326 的存储设备上并且可以装载进主存储器 304 中由处理器 302 执行。

本领域技术人员将理解图 3 中的硬件可以依据实现方式而变化。其他的内部硬件或外围设备，例如闪速只读存储器 ROM，等效非易失性存储器或光盘驱动器等，可以附加于或代替图 3 中描述的硬件来使用。而且，本发明的处理可以应用于多处理器数据处理系统。

作为另外一个例子，数据处理系统 300 可以为独立系统，该独立系统配置为可引导的，而不依赖于某种类型的网络通信接口。作为又一个例子，数据处理系统 300 可以是个人数字助理（PDA）设备，其配置有 ROM 和/或闪速 ROM 以提供存储操作系统文件和/或用户产生的数据的非易失性存储器。

图 3 中描述的例子和上述例子不意味着暗示结构的限制。例如，数据处理系统 300 除了采取 PDA 形式也可以是笔记本电脑或手提电脑。数据处理系统 300 也可以是公共咨讯站或网络设备（Web appliance）。

本发明提供了一种使用全局对象的过程间强度缩量来改善数据高速缓存的性能的方法、设备以及计算机指令。本发明的机制通过压缩或预计算全局对象来减少在软件程序循环中重复引用的全局对象的大小。全局对象包括全局动态对象和全局阵列。

对象压缩减小了高速缓存在热点的消耗并且提高了有效高速缓存的块大小以及有效高速缓存容量。预计算通过利用处理器专用硬件减少了在强度上的计算成本。

本发明的机制执行二重分析。该二重分析包括在正推法程序中遍历软件程序的中间格式文件以收集信息。中间格式文件包括调用图、

控制流图和数据流图。也可以使用其它类型的中间格式文件。

在正推法期间，本发明的机制以逆深度优先顺序或从上到下的顺序遍历调用图以收集软件程序中全局变量的定义和使用该信息。定义信息保留了在程序中存储所有全局变量的痕迹。使用信息保留了作为成本预算的候选的高成本计算的痕迹。

另外，本发明的机制使用转义分析来分析全局对象的别名，并且识别对于对象压缩或预算的候选全局对象。转义分析是一种可以用来确定全局对象的别名，即确定两个全局对象可否涉及存储器中的同一位置的技术。任何作为自变量或返回值通过的对象由定义进行转义。当一个对象转义了，这意味着它已经暴露在现有程序的外边了。如果能确定全局对象的准确别名，则全局对象可以是对象压缩的候选。使用位矢量列出候选全局对象。在遍历调用图期间改进该候选列表并且在遍历结束时完成该列表。

当识别出候选全局对象时，本发明的机制向使用列表添加代表引用全局对象的计算的树状输入。排除全局对象的索引，因为它们可以在本地变量上操作并且是不需要的。如果在候选计算中引用的全局变量是全局动态对象，那么除了树形计算，所有级别的间接引用符号也添加到该使用列表。如果引用的全局变量是全局阵列，本发明的机制就添加索引的引用符号到定义列表。

一旦识别出候选全局对象并且收集了定义和使用信息，本发明的机制就在遍历完调用图所有的节点之后在正推法的结尾分析该信息。该分析包括根据完成的别名信息来完成对于对象压缩或预计算的全局对象候选，以及根据整个程序的成本分析来识别对于强度缩量的候选计算。

如果一个计算只包含全局对象则该计算可以确定为对象压缩的候选，其中引用的全局对象的索引可以包含本地变量。当且仅当一个候选计算能够映射到大小较小的数据类型的对象的简单装载时，该计算也可以确定为对象压缩的候选。

关于优化的候选计算，例如，如果预计算的总成本比当前计算成

本低很多，计算的预计算可能为有利。预计算的总成本由两个因素确定：计算和实现定义的执行量，以及如果全局变量是动态对象或阵列时，引用的全局变量的总大小。

识别候选计算之后，本发明的机制产生新的全局变量。全局变量的大小依赖于计算中对于全局变量的间接或索引的引用的级别数。对于最高级的间接引用的新的全局变量而具有能够根据结构保持计算的数值范围的大小最小或较小的类型。对于全局变量的别名对象也产生新的符号。别名对象之间的关系也被记录以用于别名分析。

在产生新的全局变量之后，本发明的机制修改在逆推法期间所编译代码的中间格式文件。在逆推法期间，本发明的机制以深度优先顺序或从下到上的顺序遍历中间格式文件的调用图。

本发明的机制通过用新的全局动态对象或阵列的间接的或索引的引用代替候选计算的所有具体值来修改中间格式文件。关于候选全局对象，本发明的机制在软件程序中对于每个引用了候选全局对象的计算向新的全局变量在其所有的定义点插入一个存储操作，在其定义点处定义了基础符号、间接或索引的引用符号或其别名符号。

这样，本发明提供了过程间的强度缩量方法，该方法改善了数据高速缓存的性能。本发明预行计算源程序中的预计算的总成本并且确定计算是否是优选的候选。然后本发明在过程间中以新的全局对象的间接引用或新的全局阵列的索引的引用来代替候选计算，这样就减小在软件程序循环中重复引用的全局对象的大小。

现在转向图 4，即在本发明的优选实施例中描述了说明源程序、编译器和机器语言指令之间的关系的方框图。如图 4 中所说明的，在这个说明性例子中，例如程序员的用户可以定义源程序 400。源程序 400 包括变量和过程。

变量可以是全局的或本地的。全局变量可以由源程序 400 内的任何过程存取。在本例中，变量 a 406 定义为浮点型而变量 b 407 定义为整型。变量 a 406 和变量 b 407 都可由过程 foo 408 和过程 bar 412 存取。在过程中，可能出现本地变量。在本例中，变量 i 410 在过程

foo408 中定义为整型并且只在过程 foo 408 中可存取。

一旦确定了源程序 400，程序员就可以用编译器 402 编译源程序 400。编译器 402 可以在例如图 2 中的数据处理系统 200 或图 3 中的数据处理系统 300 的数据处理系统中实现。在编译处理中，编译器 402 分几个阶段处理源程序 400：词法分析阶段 414、语法分析阶段 416、中间格式文件阶段 418、代码优化阶段 420 和代码产生阶段 422。

词法分析阶段 414 分析源程序 400。在该阶段，编译器 402 读出源程序 400 中的字符并且把它们分组进代表着逻辑内聚的字符串，例如标识符、操作符和关键词的令牌流中。

语法分析 416 在令牌串上施加一个层次结构。在语法分析 416 期间，编译器 402 从词法分析 414 中获得令牌串并且通过进行从上到下或从下到上的剖析来确定该串是否是源程序的有效结构。

一旦完成了词法分析和语法分析，编译器 402 产生源程序 400 的明确的中间代码 418，该代码可以采取各种形式。例如，一个中间代码可以是调用图、数据流图或控制流图。在优选实施例中，本发明的机制对由编译器 402 产生的调用图进行二重遍历。然而，也可以使用其它中间格式文件。

在编码优化阶段 420 中，编译器 402 进行各种转换以改善中间编码。这些转换包括例如循环分片或条状开采的循环转换。这些转换改善了目标机器代码的性能。

最后，编译器 402 通过对程序使用的每个变量选择存储位置产生目标机器代码。每个中间指令被翻译成机器语言指令串，例如执行同一任务的机器语言指令 404。机器语言指令 404 可以用于例如 UNIX®平台的具体平台。UNIX 是 Open Group 的注册商标。然后程序员可以以改善的性能在具体平台上执行这些指令。

现在转向图 5，即根据本发明的优选实施例描述了使用内部程序的强度缩量来优化软件程序的处理的流程图。当如图 4 的编译器 402 的编译器产生一个源程序的中间代码时处理开始（步骤 502）。产生的中间格式文件可以是调用图、控制流图或数据流图。

接着，本发明的机制在正推法期间收集包括全局变量的使用和定义信息的信息（步骤 504）。正推法是以逆深度优先或从上到下的顺序遍历调用图。一旦收集了信息，本发明的机制就在遍历完调用图的所有节点之后在正推法结束时分析收集的信息（步骤 506）。

最后，本发明的机制在逆推法期间修改编译代码的中间格式文件（步骤 508）。逆推法是以深度优先或从下到上的顺序遍历调用图。其后处理结束。通过使用二重分析，通过增加更新标记有每个程序最小计算数目的引用而避免了冗余计算。

现在转向图 6A，即描述了根据本发明的优选实施例的在正推法期间收集信息的处理的流程图。该处理更加详细地描述了图 5 中的步骤 504。如图 6A 所描述的，当本发明的机制以深度优先顺序或从上到下的顺序遍历由编译器产生的中间格式文件时，处理开始了（步骤 602）。

接着，本发明的机制使用全局对象的过程间别名分析来识别候选全局对象（步骤 604）。在识别了候选全局对象之后本发明的机制收集对全局对象的存储操作和全局对象的计算（步骤 606）。全局对象的计算收集在使用列表中并且对全局对象的存储操作收集在定义列表中。一旦产生使用列表和定义列表，之后该处理结束。

现在转向图 6B，即根据本发明的优选实施例描述了使用过程间的别名分析来识别候选全局对象的处理的流程图。该处理更加详细地描述了图 6A 中步骤 604 的处理。如图 6B 所描述的，当本发明的机制排除了可以在未定义的程序中修改的全局变量时，处理开始（步骤 608）。

定义的程序是具有相应的中间格式文件的程序。如果一个程序的中间格式文件是不可得的，则该程序被识别为未定义程序。一个未定义程序的例子包括将汇编代码链接到编译代码的程序。未定义程序中的全局变量被排除，这是因为在这样的程序中不可能进行预计算。

接着，本发明的机制使用转义分析来分析全局动态对象和阵列的

别名（步骤 610）。可以确定别名的全局对象可以作为候选。然后，本发明的机制分析全局动态对象和阵列的大小（步骤 612）。全局对象或阵列的大小可以通过静态程序数据流分析或运行时间配置文件而得到。大小大的全局对象可以为候选。最后，由本发明的机制建立候选全局对象的列表（步骤 614）并且其后处理结束。

下面转向图 6C，即根据本发明的优选实施例描述了用于收集对于全局对象的存储操作和全局对象的计算的处理的流程图。该处理更加详细地描述了图 6A 中的步骤 606。

如图 6C 所描述的，当本发明的机制定位于下一个计算时，处理开始（步骤 620）。接着，本发明的机制确定该计算是否只包含全局变量（步骤 622）。在这种情况下，所引用的全局变量的索引可以包含本地变量。如果该计算不只包含全局变量，则该计算可能不是潜在的候选并且处理继续到步骤 636。

如果该计算只包含全局变量，则本发明的机制确定该计算能否映射到一个大小较小的数据类型的简单装载（步骤 624）。该步骤在图 6D 中更详细的进行了说明。如果该计算不能映射到一个大小较小的数据类型的简单装载，则处理继续到步骤 636。如果该计算能映射到一个大小较小的数据类型的简单装载，则本发明的机制向使用列表添加代表排除索引的计算的树（步骤 626），因为该索引是不需要的。

在添加了代表计算的树之后，本发明的机制确定在本计算中引用的全局变量是否是全局动态对象（步骤 628）。如果该全局变量是全局动态对象，则本发明的机制不只将基础对象还将间接引用符号的所有级别添加到定义列表（步骤 632）并且处理继续到步骤 636。

如果全局变量不是全局动态对象，则由本发明的机制作出关于全局变量是否是全局阵列的判定（步骤 630）。如果该全局变量是全局阵列，则本发明机制向定义列表添加索引的引用符号（步骤 634）并且该处理继续到步骤 636。否则，该处理也进行到步骤 636。

在步骤 636 中，由本发明的机制作出关于在源程序中是否存在附加计算的判定。如果存在附加计算，则处理返回到步骤 620 以检索

下一个计算。否则，之后处理结束。

接下来转向图 6D，即根据本发明的优选实施例描述了用于确定一个计算能否映射到较小的数据类型的简单装载的处理的流程图。该处理更加详细的描述图 6C 中步骤 624。如图 6D 所描述的，当本发明的机制确定一个表达式中的所有引用变量的值能否确定在范围中时，处理开始（步骤 640）。本发明的机制通过使用静态程序数据流分析或运行时间配置文件执行过程间的范围分析来收集全局对象的范围信息。

如果表达式的所有引用变量的值能够确定是在范围中，则较小的数据类型能够用来存储计算值的范围并且该处理继续到步骤 650。否则，本发明的机制确定该计算是否将包含全局动态对象或者全局阵列的表达式的类型和精度变为较小的数据类型，其为一个转换表达式（步骤 642）。转换表达式将变量的数据类型变为一个较小的数据类型，例如将双精度型变为整数型。转换表达式可以被缩减到转换的类型的大小。如果该表达式是将表达式的类型变为较小的数据类型的转换表达式，则该表达式可以是对象压缩的候选并且该处理继续到步骤 650。

然而，如果该表达式不是将表达式的类型变为较小的数据类型的转换表达式，则本发明的机制确定该表达式是否是比较表达式（步骤 644）。比较表达式是估计为值 0 或 1 的表达式。比较表达式可以依据结构将全局对象的大小缩减到单个位或字节。

如果该表达式是比较表达式，则处理继续到步骤 650。否则，本发明的机制确定该表达式是否没有以其他方式用于同一循环或其控制流区域（步骤 646）。在这种情况下，该表达式是自包含的。如果该表达式没有以其他方式用于同一循环或其控制流区域，则该处理继续到步骤 650。否则，之后处理结束。

在步骤 650 中，该表达式被识别为潜在候选并且能够映射到一个较小大小的数据类型的简单加载且之后处理结束。

现在转向图 7A，即根据本发明的优选实施例描述了用于分析收

集的信息的处理的流程图。该处理更加详细地描述了图 5 中的步骤 506。如图 7A 所描述的，当本发明的机制分析使用由图 6B，6C 和 6D 中的处理所产生的使用列表和定义列表时，处理开始（步骤 702）。接着，本发明的机制使用完成的别名信息来改进候选全局对象（步骤 704）。

一旦改进了候选全局对象，本发明的机制就使用成本分析选择候选计算（步骤 706）。一旦选择了候选计算，本发明的机制就对于候选计算产生新的全局变量（步骤 708）。产生的新的全局变量包括基础符号和间接或索引的引用符号。之后该处理结束。

现在转向图 7B，即根据本发明的优选实施例描述了用于根据成本分析为强度缩量选择候选计算并且对该候选计算产生全局变量的处理的流程图。该处理更加详细地描述了图 7A 中的步骤 704 和 706。如图 7B 所描述的，当本发明的机制在使用列表中定位下一个项时，处理开始（步骤 708）。接着，本发明的机制选择被树状计算引用的全局变量（步骤 710）并且从与引用的全局变量相对应的定义列表得到执行成本（步骤 712）。对于流不敏感分析，该执行成本可以是执行计数。对于流敏感分析，该执行成本可以是为各个全局变量存储指定的加权。根据从控制流和数据流分析中获得的达到定义和达到的使用信息来确定该加权。

在获得执行成本之后，本发明的机制比较树状计算引用的全局变量的执行计数（步骤 714）并且确定该树状计算是否是预计算优化的候选（步骤 716）。该步骤包括确定该计算的预计算的总成本是否比当前计算成本低很多。

如果该树状计算是预计算优化的候选，则本发明的机制确定在该计算中引用的全局变量是否有很大的大小（步骤 718）。如果该全局变量的大小为大，则本发明的机制对于产生该计算的新全局符号（步骤 720）和该计算中的间接或索引的引用的每个级别的新的全局符号（步骤 722）。产生该符号之后，本发明的机制将该符号添加到使用列表（步骤 724）。然后该处理继续到步骤 730。

转回到步骤 718，如果该全局对象大小不大，则本发明的机制确定该树状计算是否包括任何子计算（步骤 726）。如果存在子计算，则本发明的机制将该子计算添加到使用列表（步骤 728）并且返回到步骤 710 以进一步分析该子计算。如果不存在子计算，则该处理继续到步骤 730。

在步骤 730 中，本发明的机制确定在使用列表中是否存在附加项。如果存在附加项，则该处理返回到步骤 708 以获得下一个项。否则，之后处理结束。

现在转向图 8A，即根据本发明的优选实施例描述了在逆推法期间修改代码的中间格式文件的处理的流程图。该处理更加详细地描述了图 5 中的步骤 508。如图 8A 所描述的，当本发明的机制以深度优先顺序或从下到上的顺序遍历由编译器产生的中间格式文件时，处理开始（步骤 802）。

接着，本发明的机制为引用全局变量的每一个计算插入存储操作（步骤 804）。该存储操作被插入到在整个程序中所有的全局变量的定义点，在这些点上定义了基础符号、其间接或索引的引用符号或其别名符号。最后，本发明的机制用新的全局动态对象的间接引用或新的全局阵列的索引引用来代替候选计算（步骤 806）。之后处理结束。

现在转向图 8B，即根据本发明的优选实施例描述了对于引用全局变量的每个计算插入存储操作的处理的流程图。该处理更加详细地描述了图 8A 中的步骤 804。如图 8B 所描述的，当本发明的机制对候选计算所引用的下一个全局变量进行定位时，处理开始（步骤 808）。

接着，本发明的机制确定候选计算是否是对动态对象或阵列的间接或索引的引用的最高级别的存储操作（步骤 814）。如果该候选计算是对间接或索引的引用的最高级别的存储操作，则本发明的机制插入该计算的间接或索引的引用值的最高级别的存储操作（步骤 816）。之后处理继续到步骤 830。

如果该候选计算不是间接或索引的引用的最高级别的存储操作，则本发明的机制继续确定该候选计算是否是对于动态对象的间接引用的第二高级别的存储分配（步骤 818）。如果该候选计算是动态对象的间接引用的第二高级别的存储分配，则本发明的机制插入一个存储操作以对具有较小大小的新的动态对象的间接引用的第二高级别进行分配存储（步骤 820）。如果该间接引用的最高级别是 1，则该间接引用的第二高级别是基础指针。之后处理继续到步骤 830。

返回到步骤 818，如果该候选计算不是对于动态对象的间接引用的第二高级别的存储分配，则本发明的机制确定该候选计算是否是对于动态对象的间接引用指针的基础指针或较低级别的存储分配（步骤 822）。

如果该候选计算是对于动态对象的间接引用指针的基础指针或较低级别的存储分配，本发明的机制就插入一个存储操作以对具有同样大小的新的动态对象的间接引用指针的基础指针或较低级别分配存储（步骤 824）。之后处理继续到步骤 830。

否则，本发明的机制确定该候选计算是否是动态对象的指针赋值（步骤 826）。如果该候选计算是动态对象的指针赋值，则本发明的机制通过用新的全局变量和其相应的别名符号代替整个赋值语句来向该新的全局变量插入存储操作（步骤 828）。之后该处理继续到步骤 830。如果该候选计算不是指针赋值，则该处理也继续到步骤 830。

在步骤 830 中，本发明的机制确定附加的全局变量是否由候选计算引用。如果附加全局变量由候选计算引用，则处理返回步骤 808 以定位由候选计算引用的下一个全局变量。否则这之后该处理结束。

现在转到图 9，即说明了根据本发明的优选实施例描述了使用过程间的强度缩量在优化之前的示例性软件程序的示意图。如图 9 所描述的，在该示例性实施例中，源程序 900 包括变量声明 902，bar 过程 904 和 foo 过程 906。

变量声明 902 包括两个声明，变量“a”和“b”的声明。变量“a”是类型指针到浮点指针的全局动态对象。变量“b”是 M×N 维的二

维整数阵列。

Bar 过程 904 包括变量“a”的存储操作 908，其向变量“a”分配大小为 M 个浮点指针的存储。Bar 过程 904 还包括动态阵列“a[i]”的存储操作 910，其向变量 a[i]分配大小为 N 个浮点的存储。

过程 foo906 包括两个表达式，表达式 912 和 914。表达式 912 将 Fa（a[i][j]）916 与 1 比较。Fa916 是包括引用 a 的表达式。表达式 914 是按模计算表达式，其将 Fb（b[i][j]）918 对于 256 求模以将 Fb918 的值转换为字节的大小。

在正推法期间，本发明的机制不仅收集基础全局对象的信息还收集其间接或索引的引用的级别。在该例中，定义列表包括对基础符号 a、间接引用的第二高级 a[]、间接引用的第一高级 a[][]以及索引引用 b[][]的项。也记录了对象的基础符号与其间接或索引的引用之间的关系。

另一方面，使用列表包括树状表达式 912 和 914 的项。表达式 912 被添加到使用列表，这是因为 Fa916 是比较表达式并且被本发明的机制识别为对象压缩的候选。表达式 914 也被添加到使用列表，这是因为 Fb918 将一个值转换成比整型的大小小的字节的大小。因此，表达式 914 也被识别为对象压缩的候选。当该表达式被记录在使用列表中时，就从存储在使用列表中的项中排除所记录的表达式的索引信息。

在正推法期间收集定义和使用信息后，本发明的机制根据成本分析来分析该信息并且选择出对强度缩量的候选计算。如果一个计算的预计算的总成本远低于当前计算的成本，则选择该计算为优化的候选。在该例子中，如果确定变量“a”和“b”大小为大并且其执行计数远高于其定义的执行计数，则选择表达式 912 和 914 为优化的候选计算。

然后本发明的机制产生对两个选择的计算的新的全局变量。产生的全局变量的数目依赖于该计算中其间接的或索引的引用的级别的数目。在该例中，对全局变量“a”产生的新的全局变量是 isra，对

应于间接引用 a[]的第二高级的是 isra[]，而对应于间接引用 a[][]的第一高级的是 isra[][]。相似地，也对于全局变量 b 产生 isrb 和 isrb[][]。

在产生新的全局变量之后，本发明的机制对于每一个选择的引用这两个全局变量的计算插入存储操作。本发明的机制以从下到上的顺序或以深度优先的顺序遍历调用图。

现在转向图 10，即说明了根据本发明的优选实施例描述的使用程序间的强度缩量的优化之后的示例性软件程序的图。如图 10 所描述的，在本示例性实施中，优化的程序 1000 类似于图 9 中的源程序 900，其包括声明 1002，bar 过程 1004 和 foo 过程 1006。然而，本发明的机制对于在声明 1002 中声明的每一个变量在 bar 过程 1004 中插入两个存储操作。

在声明 1002 中声明的变量中的一个是声明 1008 或“bool **isra”，用类型指针到布尔指针声明了所创建的变量“a”的新的全局变量。如图 9 所述，由于 Fa 是能够缩减到 1 比特或 1 字节的大小的比较表达式，所以表达式 912 能够缩减到布尔表达式。

在声明 1002 中声明的另一个变量是声明 1010 或“char isrb[M][N]”，用字符型和 M×N 维声明所创建的变量“b”新的全局阵列。如图 9 所述，因为表达式 914 将 Fb 的值转换为一字节的大小以及同样具有一字节大小的字符型，所以能够通过使用较小的数据类型优化表达式 914。

Bar 过程 1004 包括两个插入的存储操作 1014 和 1016。操作 1014 向新的全局变量“isra”分配具有 M 个布尔指针大小的存储。插入第二个插入的存储操作，操作 1016，以向新的全局符号产生的“isra[]”分配具有 N 个布尔指针大小的存储。操作 1016 以较小的大小分配存储，这是因为 isra[]是变量 a 的间接引用的第二高级。

在插入存储操作之后，本发明的机制用新的全局对象的间接引用或新的全局阵列的索引引用来代替所选的候选计算。在这个例子中，图 9 中的表达式 912 以 isra[i][j]来代替，其为变量“a”的间接引用的最高级，并且表达式 914 以 isrb[i][j]来代替，其为变量“b”的索

引引用。

现在转向图 11，即说明了根据本发明的优选实施例所描述的使用程序间的强度缩量的进一步优化之后的示例性软件程序的图。如图 11 所描述的，程序 1100 类似于图 10 中优化的程序 1000，除了 foo 过程 1102 被进一步优化，去掉了 if 分支语句。

总之，本发明提供一种使用全局对象的程序间的强度缩量来改善数据高速缓存的性能的方法、设备和计算机指令。当一个程序中的计算改变且以新的全局变量的间接或索引的引用来替代该循环中的计算时，本发明产生并更新全局变量。以此方式，过程间地分析全局变量的使用方式并且将全局对象重新映射到较小大小的对象，这样就可以产生高速缓存效率更高的程序并且改善数据高速缓存的性能。

很重要的是注意到当在全功能数据处理系统的情景下描述本发明时，本领域普通技术人员将理解本发明的处理能够以指令的计算机可读介质的形式和多种形式来分布，并且不管实际用于执行该分布的信号承载介质的具体类型本发明都同样地适用。计算机可读介质的例子包括例如软盘、硬盘驱动器、RAM、CD-ROM、DVD-ROM 的可再次记录型的介质以及例如数字和模拟通信链接、使用了例如射频和光波传输的传输形式的有线或无线通信链接的传输类型媒体。该计算机可读介质可以采用编码格式的形式，其在具体的数据处理系统中被解码以供实际使用。

本发明的描述是为了解释和说明的目的，而没有意图用本发明所公开的形式来穷尽或是限制发明。很多的修改和变形对于本领域普通技术人员是显而易见的。实施例的选择和描述是为了最好地解释本发明的原理，实际应用，并且可以使本技术领域的其他普通技术人员理解本发明的适用于预期的具体使用的具有各种修改的各种实施例。

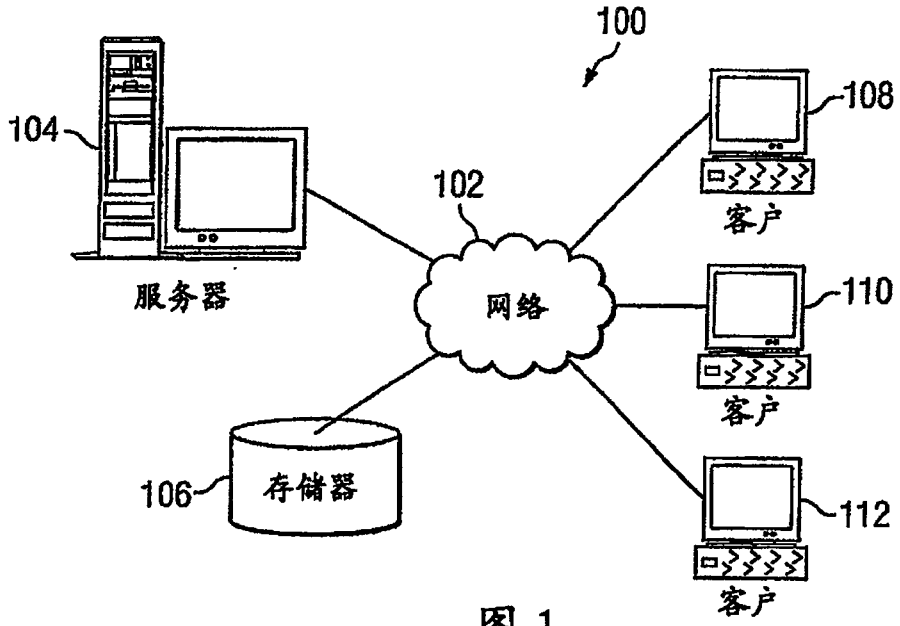

100
104
服务器
102
网络
108
客户
110
客户
112
客户
106
存储器

图 1

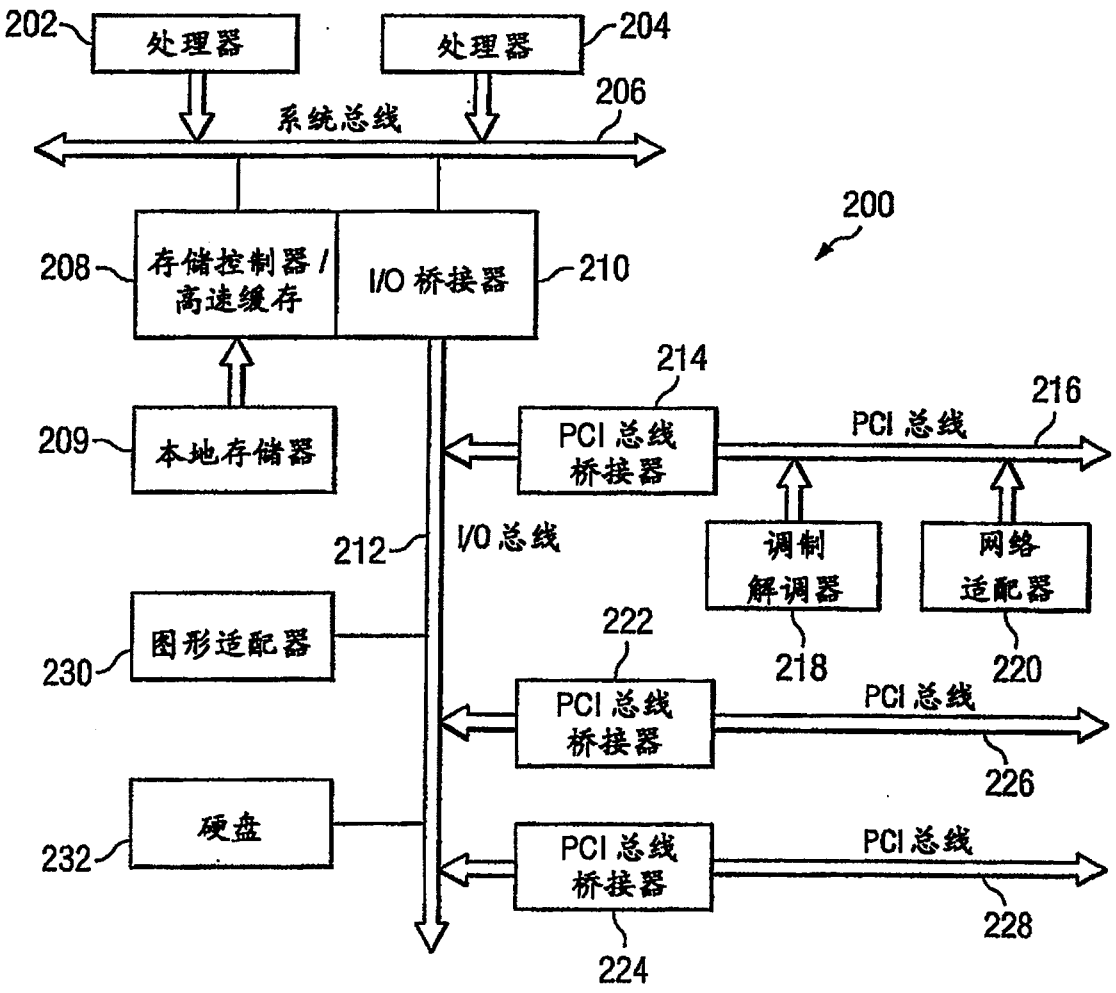

202
处理器
处理器
204
206
系统总线
200
208
存储控制器 / 高速缓存
I/O 桥接器
210
214
216
209
本地存储器
PCI 总线桥接器
PCI 总线
212
I/O 总线
调制解调器
网络适配器
222
230
图形适配器
218
220
PCI 总线桥接器
PCI 总线
226
232
硬盘
PCI 总线桥接器
PCI 总线
228
224

图 2

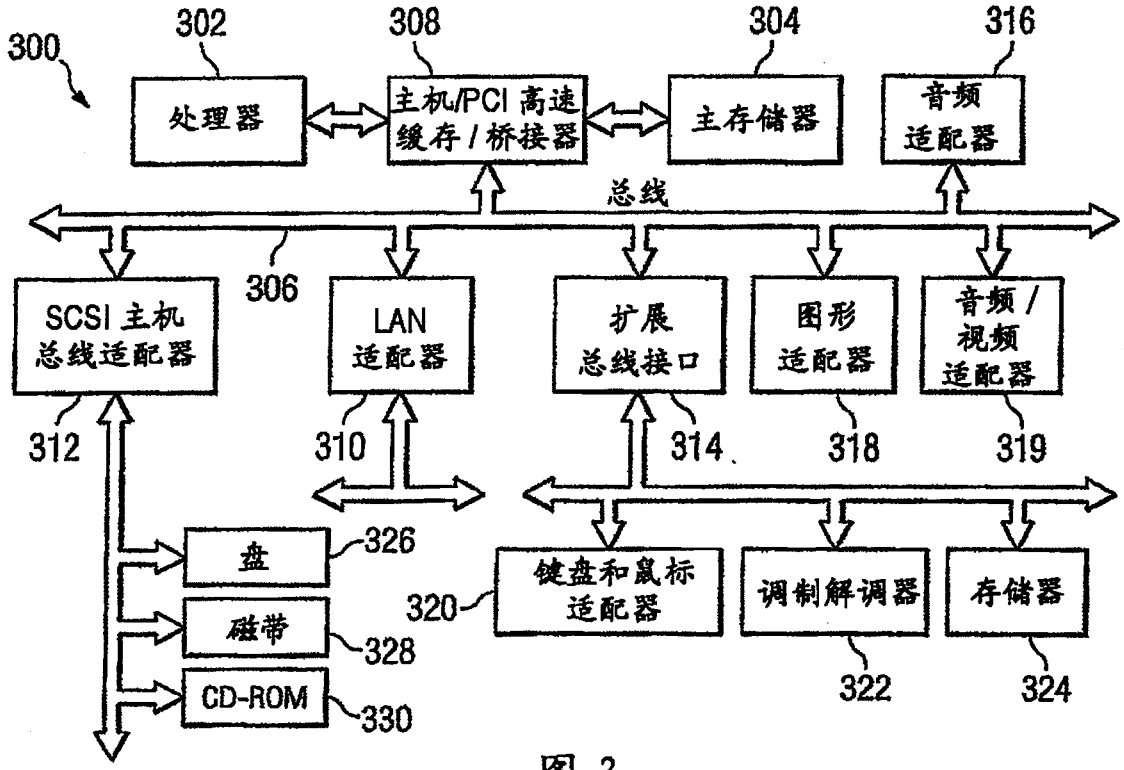

300
302
处理器
308
主机/PCI 高速缓存 / 桥接器
304
主存储器
316
音频适配器
总线
306
SCSI 主机总线适配器
LAN 适配器
扩展总线接口
图形适配器
音频 / 视频适配器
312
310
314
318
319
盘
326
磁带
328
CD-ROM
330
320
键盘和鼠标适配器
调制解调器
322
存储器
324

图 3

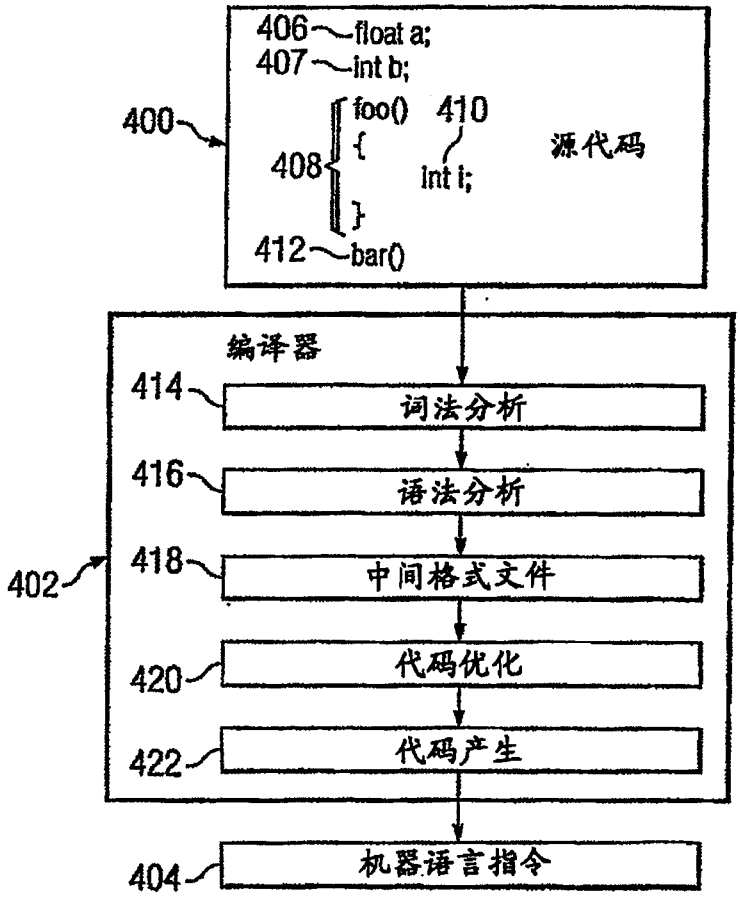

406 float a;
407 int b;
400
foo() 410
{
408
int i;
}
412 bar()
源代码
编译器
414 词法分析
416 语法分析
402
418 中间格式文件
420 代码优化
422 代码产生
404 机器语言指令

图 4

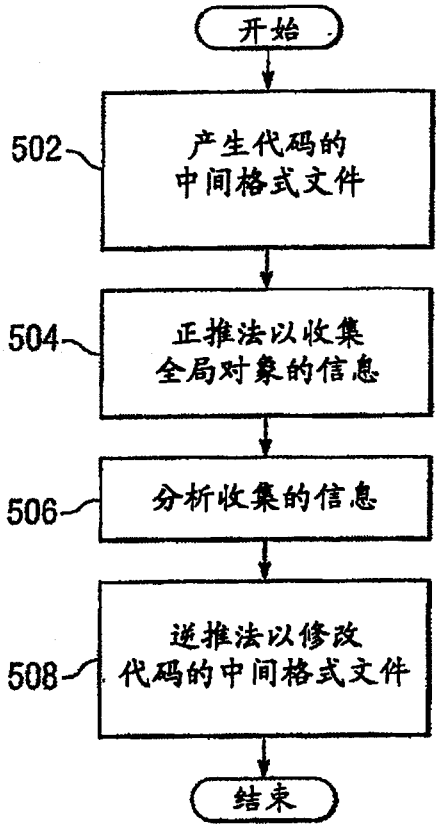

开始
502
产生代码的
中间格式文件
504
正推法以收集
全局对象的信息
506
分析收集的信息
508
逆推法以修改
代码的中间格式文件
结束

图 5

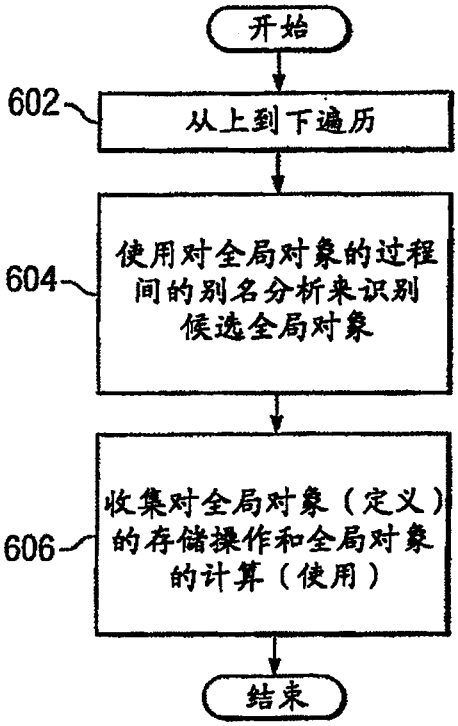

开始
602
从上到下遍历
604
使用对全局对象的过程
间的别名分析来识别
候选全局对象
606
收集对全局对象（定义）
的存储操作和全局对象
的计算（使用）
结束

图 6A

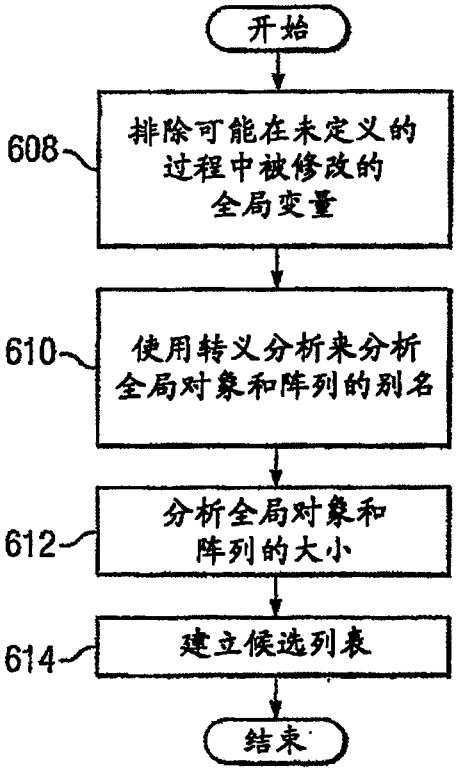

开始
608
排除可能在未定义的
过程中被修改的
全局变量
610
使用转义分析来分析
全局对象和阵列的别名
612
分析全局对象和
阵列的大小
614
建立候选列表
结束

图 6B

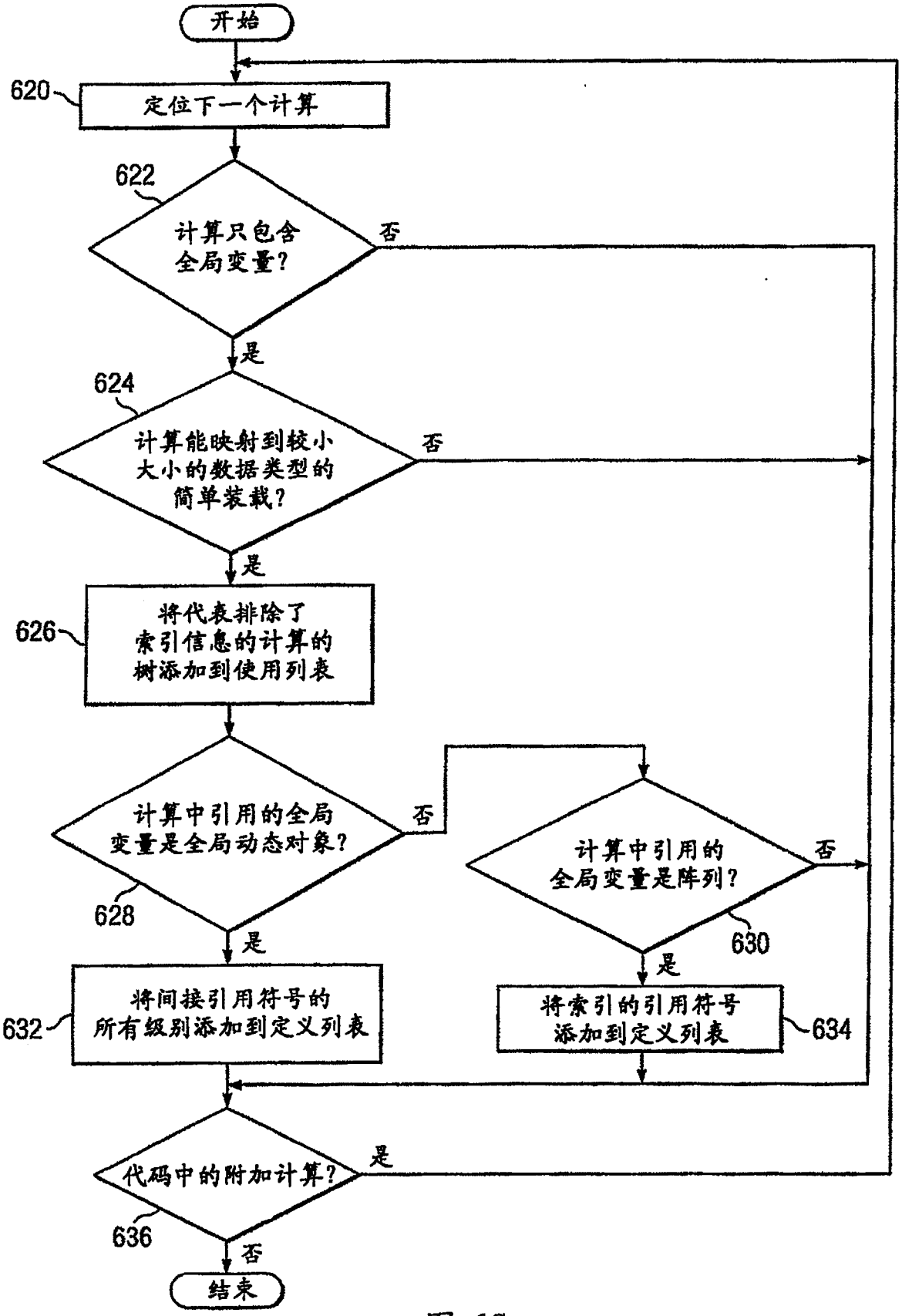

开始
620
定位下一个计算
622
计算只包含全局变量?
否
是
624
计算能映射到较小大小的数据类型的简单装载?
否
是
626
将代表排除了索引信息的计算的树添加到使用列表
计算中引用的全局变量是全局动态对象?
否
628
是
计算中引用的全局变量是阵列?
否
630
是
632
将间接引用符号的所有级别添加到定义列表
将索引的引用符号添加到定义列表
634
代码中的附加计算?
是
636
否
结束

图 6C

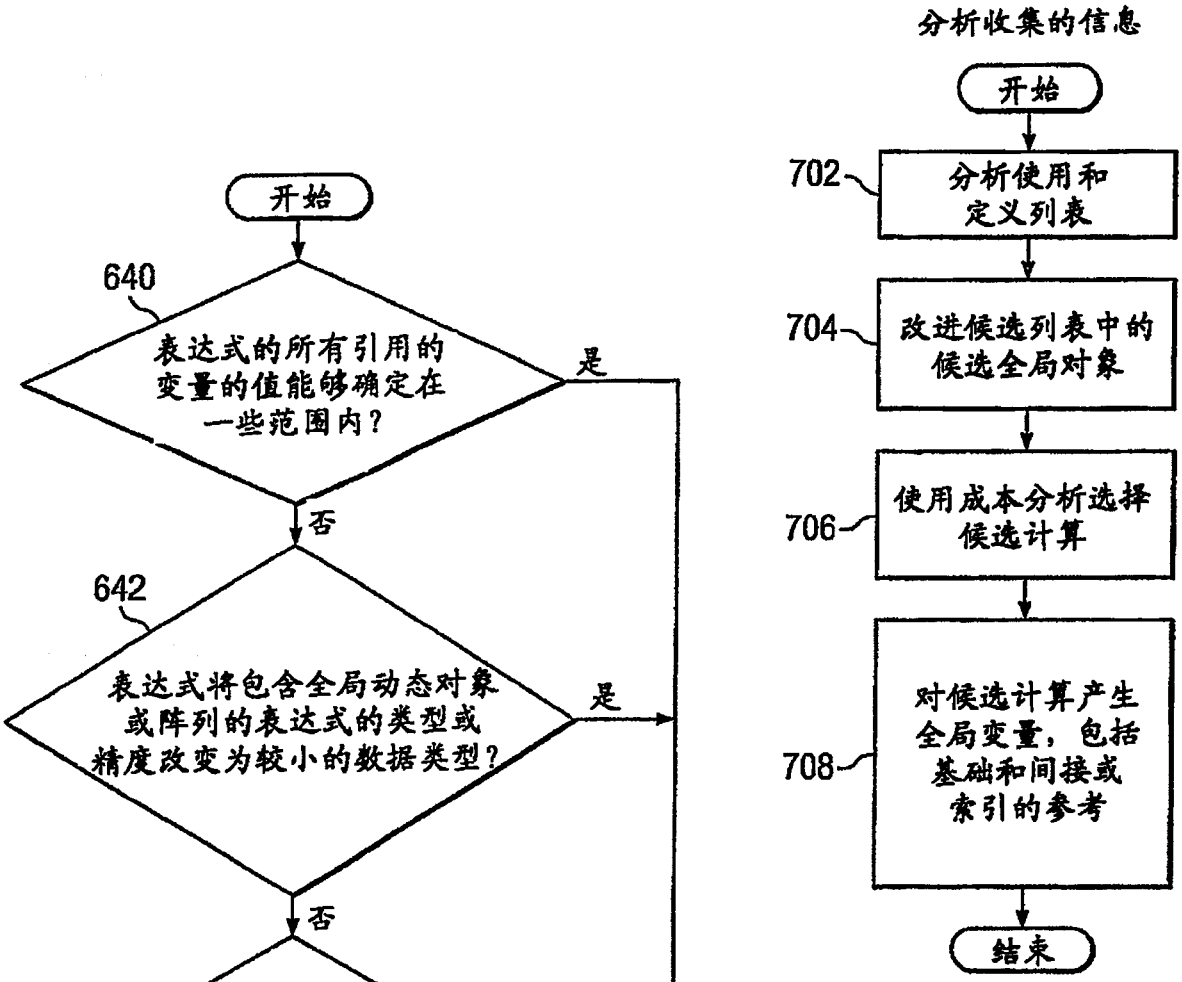

分析收集的信息
开始
702
分析使用和定义列表
704
改进候选列表中的候选全局对象
706
使用成本分析选择候选计算
708
对候选计算产生全局变量，包括基础和间接或索引的参考
结束

图 7A

开始
640
表达式的所有引用的变量的值能够确定在一些范围内？
是
否
642
表达式将包含全局动态对象或阵列的表达式的类型或精度改变为较小的数据类型？
是
否
表达式是比较表达式？
是
644
否
表达式没有以其他方式用于同一循环或其控制流区域？
是
646
否
表达式是潜在候选
650
结束

图 6D

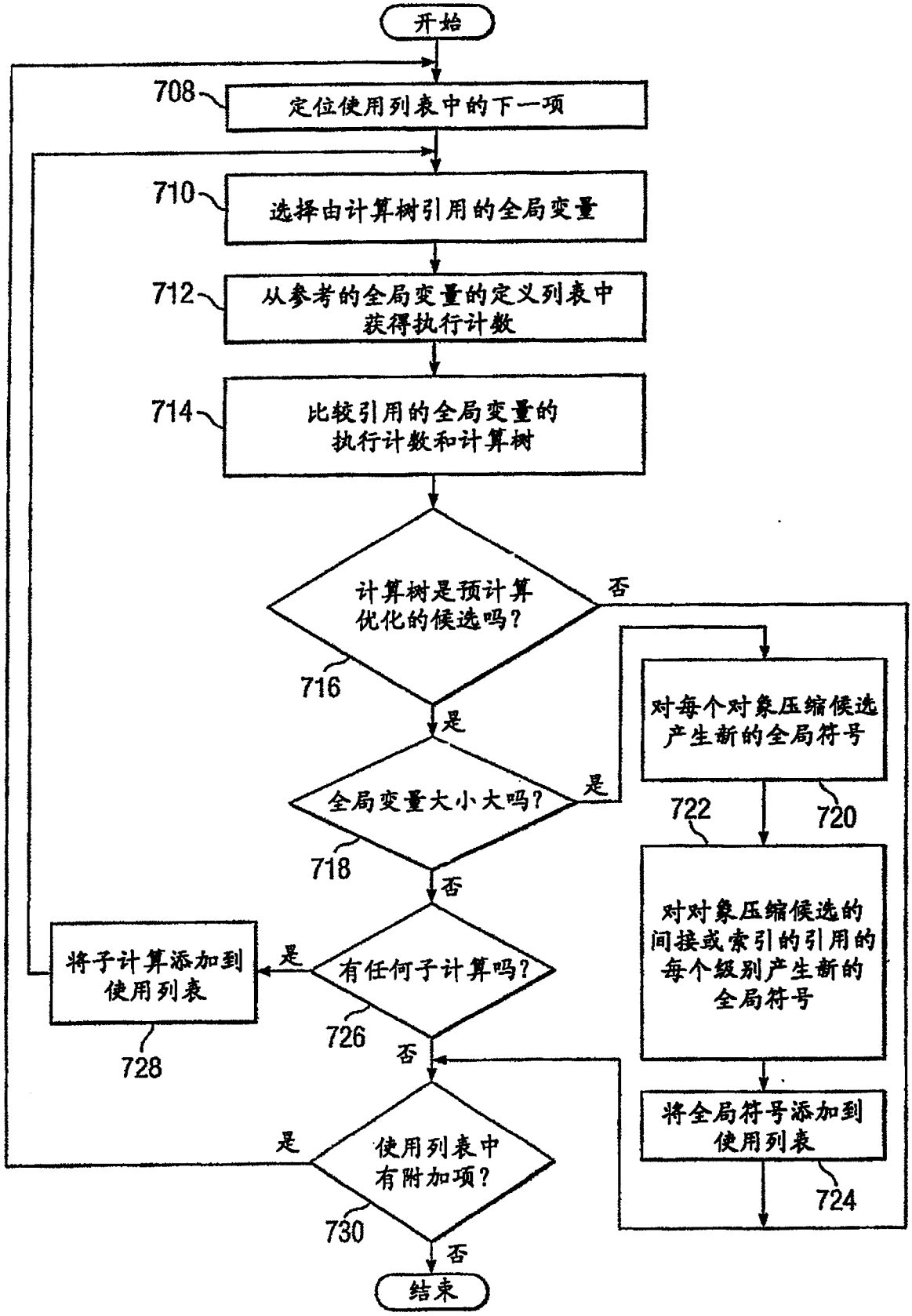

开始
708
定位使用列表中的下一项
710
选择由计算树引用的全局变量
712
从参考的全局变量的定义列表中获得执行计数
714
比较引用的全局变量的执行计数和计算树
计算树是预计算优化的候选吗?
否
716
是
全局变量大小大吗?
是
718
否
对每个对象压缩候选产生新的全局符号
722
720
对对象压缩候选的间接或索引的引用的每个级别产生新的全局符号
将全局符号添加到使用列表
724
有任何子计算吗?
是
将子计算添加到使用列表
728
726
否
使用列表中有附加项?
是
730
否
结束

图 7B

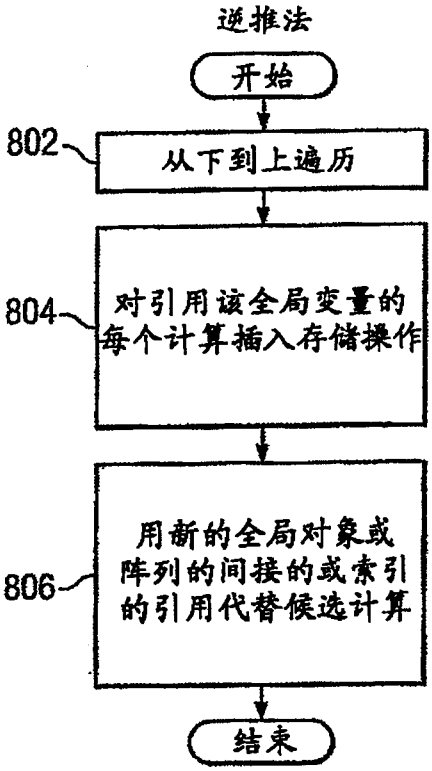

逆推法
开始
802
从下到上遍历
804
对引用该全局变量的每个计算插入存储操作
806
用新的全局对象或阵列的间接的或索引的引用代替候选计算
结束

图 8A

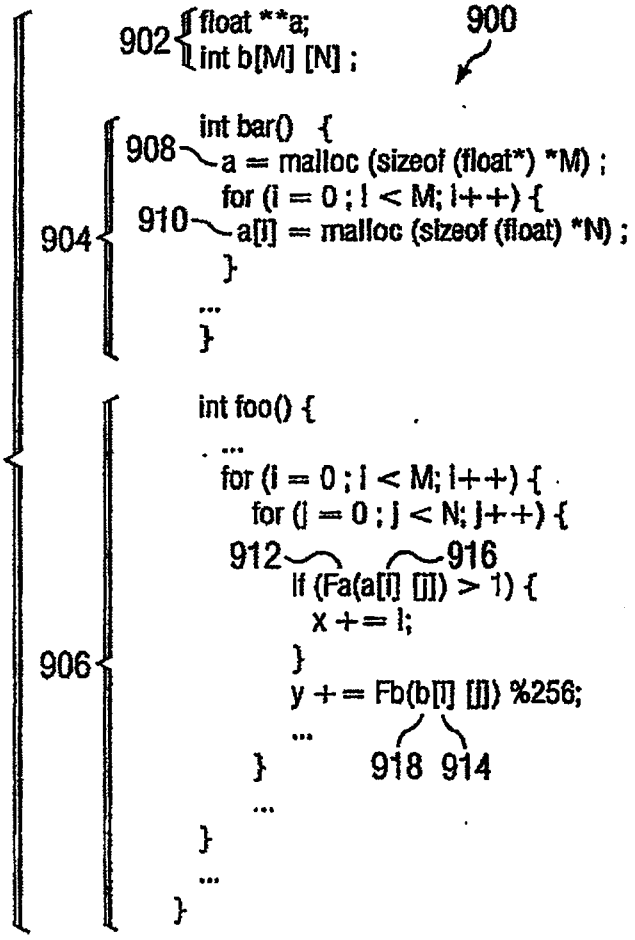

900
902
float **a;
int b[M] [N] ;
int bar() {
908
a = malloc (sizeof (float*) *M) ;
for (i = 0 ; i < M; i++) {
910
a[i] = malloc (sizeof (float) *N) ;
}
...
}
904
int foo() {
...
for (i = 0 ; i < M; i++) {
for (j = 0 ; j < N; j++) {
912
916
if (Fa(a[i] [j]) > 1) {
x += i;
}
y += Fb(b[i] [j]) %256;
...
}
...
}
...
}
906
918
914

图 9

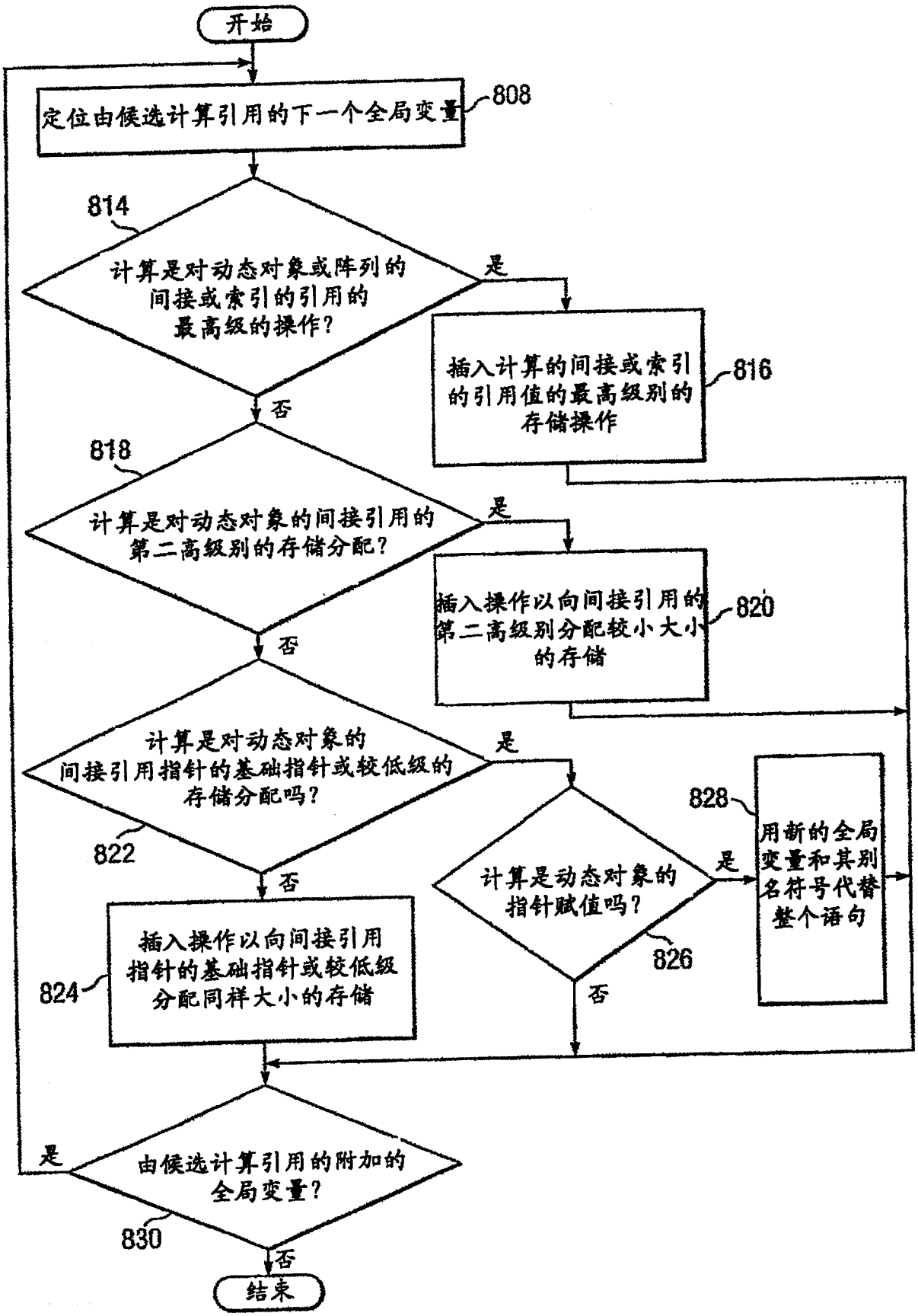

开始
定位由候选计算引用的下一个全局变量
808
814
计算是对动态对象或阵列的间接或索引的引用的最高级的操作?
是
否
插入计算的间接或索引的引用值的最高级别的存储操作
816
818
计算是对动态对象的间接引用的第二高级别的存储分配?
是
否
插入操作以向间接引用的第二高级别分配较小大小的存储
820
计算是对动态对象的间接引用指针的基础指针或较低级的存储分配吗?
是
否
822
计算是动态对象的指针赋值吗?
是
否
826
828
用新的全局变量和其别名符号代替整个语句
插入操作以向间接引用指针的基础指针或较低级分配同样大小的存储
824
由候选计算引用的附加的全局变量?
是
否
830
结束

图 8B

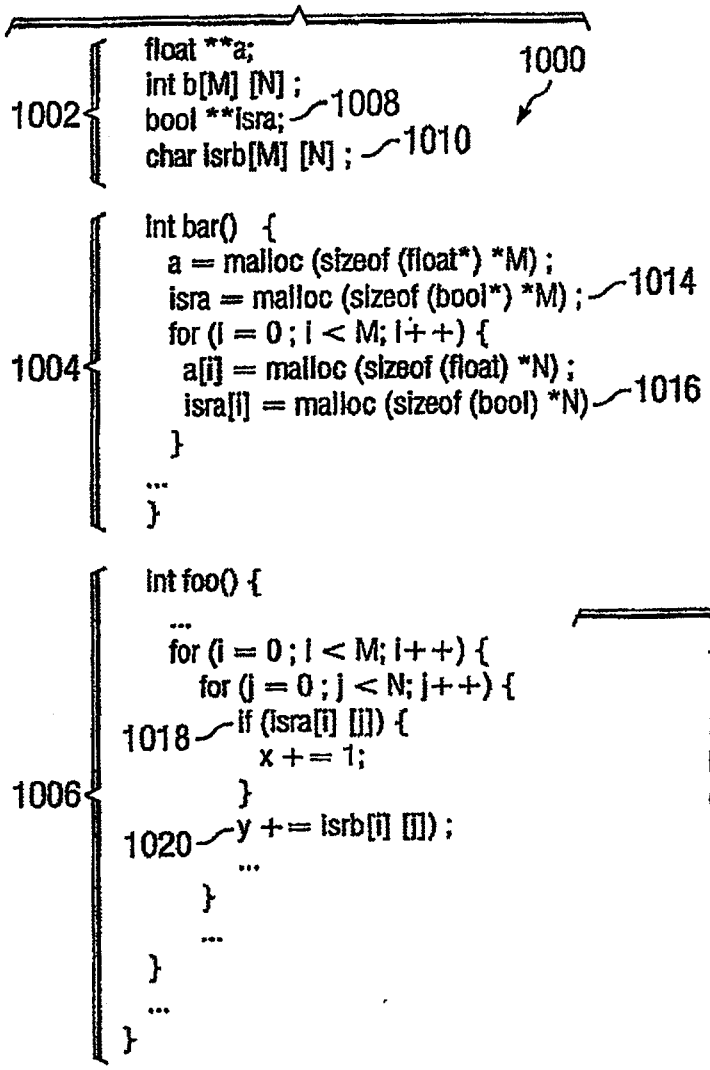

float **a;
1000
int b[M] [N] ;
1002
bool **isra;
1008
char isrb[M] [N] ;
1010
int bar() {
a = malloc (sizeof (float*) *M) ;
isra = malloc (sizeof (bool*) *M) ;
1014
for (i = 0 ; i < M; i++) {
1004
a[i] = malloc (sizeof (float) *N) ;
isra[i] = malloc (sizeof (bool) *N)
1016
}
...
}
int foo() {
...
for (i = 0 ; i < M; i++) {
for (j = 0 ; j < N; j++) {
1018
if (isra[i] [j]) {
x += 1;
1006
}
y += isrb[i] [j]) ;
1020
...
}
...
}
...
}

图 10

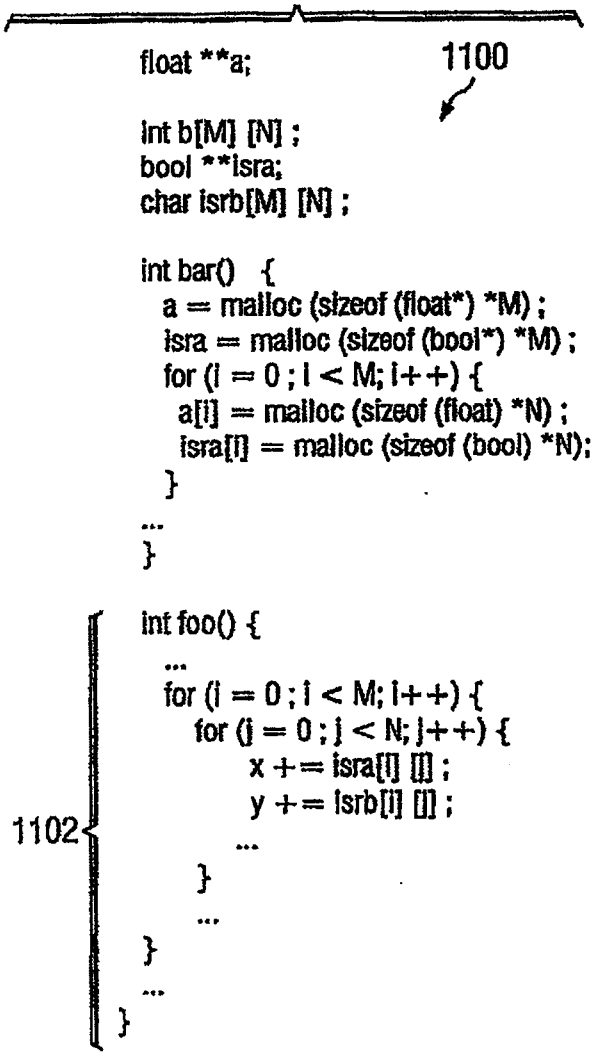

float **a;
1100
int b[M] [N] ;
bool **isra;
char isrb[M] [N] ;
int bar() {
a = malloc (sizeof (float*) *M) ;
isra = malloc (sizeof (bool*) *M) ;
for (i = 0 ; i < M; i++) {
a[i] = malloc (sizeof (float) *N) ;
isra[i] = malloc (sizeof (bool) *N);
}
...
}
int foo() {
...
for (i = 0 ; i < M; i++) {
for (j = 0 ; j < N; j++) {
x += isra[i] [j] ;
1102
y += isrb[i] [j] ;
...
}
...
}
...
}

图 11