

(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 102156630 A

(43) 申请公布日 2011. 08. 17

(21) 申请号 201110027146. 1

(22) 申请日 2011. 01. 20

(30) 优先权数据

12/691, 378 2010. 01. 21 US

(71) 申请人 微软公司

地址 美国华盛顿州

(72) 发明人 S · H · 托布 E · 奥玛拉 J · 达菲

(74) 专利代理机构 上海专利商标事务所有限公司 31100

代理人 蔡悦 钱静芳

(51) Int. Cl.

G06F 9/312 (2006. 01)

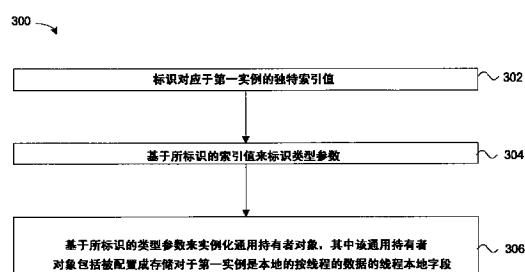
权利要求书 2 页 说明书 10 页 附图 6 页

(54) 发明名称

按线程按实例的数据存储

(57) 摘要

一种存储按线程、按实例的数据的方法，包括标识对应于第一实例的唯一索引值，基于所标识的索引值来标识类型参数，以及基于所标识的类型参数来实例化通用持有者对象。该通用持有者对象包括被配置成存储对于第一实例是本地的按线程数据的线程本地字段。



1. 一种存储按线程、按实例的数据的方法,包括 :

标识对应于第一实例的独特索引值;

基于所标识的索引值来标识类型参数;

基于所标识的类型参数来实例化通用持有者对象,所述通用持有者对象包括被配置成存储对于所述第一实例是本地的按线程的数据的线程本地字段;以及

其中所述标识独特索引值、标识类型参数、以及实例化通用持有者对象由至少一个处理器来执行。

2. 如权利要求 1 所述的方法,其特征在于,所述线程本地字段通过将线程静态属性应用于静态变量来创建。

3. 如权利要求 1 所述的方法,其特征在于,所标识的类型参数对于所述第一实例是独特的,并且不被其他实例同时共享。

4. 如权利要求 1 所述的方法,其特征在于,还包括:

实例化包括引用所述通用持有者对象的第一字段以及存储所述索引值的第二字段的线程本地对象。

5. 如权利要求 1 所述的方法,其特征在于,还包括:

创建哑类型数组;以及

其中,所标识的类型参数取自所述哑类型数组。

6. 如权利要求 1 所述的方法,其特征在于,还包括:

提供用于存储索引值的静态变量;以及

每次生成新实例时更新所述静态变量。

7. 如权利要求 6 所述的方法,其特征在于,还包括:

标识先前使用但不再使用的索引值;以及

将所标识的先前使用的索引值存储在数据结构中。

8. 如权利要求 7 所述的方法,其特征在于,所述标识独特索引值包括:

如果所述数据结构当前正存储至少一个索引值,则从所述数据结构中选择所述独特索引值;以及

如果所述数据结构当前没有正存储至少一个索引值,则基于所述静态变量选择所述独特索引值。

9. 如权利要求 1 所述的方法,其特征在于,还包括:

为多个实例中的每一个重复所述标识独特索引值、标识类型参数、以及实例化通用持有者对象。

10. 如权利要求 9 所述的方法,其特征在于,还包括:

确定所述多个实例中的实例总数是否超出预定阈值;以及

当所述实例总数超出预定阈值时,将数据槽分配给线程来用于存储按线程、按实例的数据。

11. 如权利要求 1 所述的方法,其特征在于,所述独特索引值是以 10 为基数的数字,并且所述方法还包括:

将所述以 10 为基数的数字转换成具有不同基数的数字;以及

其中,所述类型参数是基于所述具有不同基数的数字来标识的。

12. 一种存储计算机可执行指令的计算机可读存储介质,所述指令在由至少一个处理器执行时使得所述至少一个处理器执行一种存储按线程、按实例的数据的方法,所述方法包括:

标识对应于第一实例的独特索引值;

基于所标识的索引值来标识类型参数;以及

基于所标识的类型参数来实例化通用持有者对象,所述通用持有者对象包括被配置成存储对于所述第一实例是本地的按线程的数据的线程本地字段。

13. 如权利要求 12 所述的计算机可读介质,其特征在于,所述线程本地字段通过将线程静态属性应用于静态变量来创建。

14. 如权利要求 12 所述的计算机可读介质,其特征在于,所标识的类型参数对于所述第一实例是独特的,并且不被其他实例同时共享。

15. 如权利要求 12 所述的计算机可读介质,其特征在于,所述方法还包括:

实例化包括引用所述通用持有者对象的第一字段以及存储所述索引值的第二字段的线程本地对象。

16. 如权利要求 12 所述的计算机可读介质,其特征在于,所述方法还包括:

创建哑类型数组;以及

其中,所标识的类型参数取自所述哑类型数组。

17. 如权利要求 12 所述的计算机可读介质,其特征在于,所述方法还包括:

提供用于存储索引值的静态变量;以及

每次生成新实例时更新所述静态变量。

18. 如权利要求 17 所述的计算机可读介质,其特征在于,所述方法还包括:

标识先前使用但不再使用的索引值;以及

将所标识的先前使用的索引值存储在数据结构中。

19. 如权利要求 18 所述的计算机可读介质,其特征在于,所述标识独特索引值包括:

如果所述数据结构当前正存储至少一个索引值,则从所述数据结构中选择所述独特索引值;以及

如果所述数据结构当前没有正存储至少一个索引值,则基于所述静态变量选择所述独特索引值。

20. 一种存储按线程、按实例的数据的方法,包括:

实例化线程本地对象,由此生成第一实例;

标识对应于所述第一实例的独特索引值;

基于所标识的索引值来标识类型参数;

基于所标识的类型参数来实例化通用持有者对象,所述通用持有者对象包括被配置成存储对于所述第一实例是本地的按线程的数据的线程本地字段;以及

其中所述实例化线程本地对象、标识独特索引值、标识类型参数、以及实例化通用持有者对象由至少一个处理器来执行。

按线程按实例的数据存储

技术领域

[0001] 本发明涉及计算机应用开发,尤其涉及一种计算机应用开发中的数据存储。

[0002] 背景

[0003] 软件程序从软件开发开始的日子起就被编写为顺序地运行。计算机随着时间不断变得更加强大,具有更多的处理能力和存储器来处理高级操作。这一趋势最近从日益增长的单处理器时钟速率转向增加单个计算机中可用处理器的数量,这导致相应的从顺序执行转向并行执行。软件开发者想要利用计算机处理能力上的改进来使他们的软件程序能够在采用新硬件时运行得更快。采用并行硬件,软件开发者安排特定软件程序的一个或多个任务并行地(也称为并发地)执行,以使例如同一逻辑运算可以同时利用多个处理器,由此在向该软件在其上运行的计算机添加更多处理器时提供更好的性能。

[0004] 概述

[0005] 提供本概述是为了以简化的形式介绍将在以下详细描述中进一步描述的一些概念。本概述并不旨在标识所要求保护的主题的关键特征或必要特征,也不旨在用于限制所要求保护的主题的范围。

[0006] 一个实施例利用包含线程静态字段的通用持有者实例,并且在运行时以实现快速的按线程、按实例的存储的方式来动态地参数化该通用持有者实例。在一个实施例中,通用类型的实例化被用作实现按实例、按线程的数据存储的机制。

[0007] 一个实施例提供了一种存储按线程、按实例的数据的方法,包括标识对应于第一实例的独特索引值,基于所标识的索引值来标识类型参数,以及基于所标识的类型参数来实例化通用持有者对象。该通用持有者对象包括被配置成存储第一实例本地的按线程的数据的线程本地字段。

[0008] 附图简述

[0009] 包括附图来提供对各实施例的进一步理解,且这些附图被合并在本发明书内并构成其一部分。附图示出各实施例,并且与说明书一起用于解释本发明的原理。其他实施例和各实施例的许多预期优点将随着参考下面的详细描述进行更好的理解而得到认识。附图的元素不一定相对于彼此而缩放。相同的附图标记指代对应的类似部分。

[0010] 图1是示出根据一个实施例的适于执行按线程、按实例的数据存储应用程序的计算设备的图示。

[0011] 图2是根据一个实施例的用于在图1所示的计算设备上操作的按线程、按实例的数据存储应用程序的图示。

[0012] 图3是根据一个实施例的存储按线程、按实例的数据的方法的流程图。

[0013] 图4是根据另一实施例的存储按线程、按实例的数据的方法的流程图。

[0014] 图5是根据一个实施例的标识独特索引值的方法的流程图。

[0015] 图6是根据一个实施例的生成用于存储按线程、按实例的数据的持有者的方法的流程图。

[0016] 详细描述

[0017] 在以下详细描述中,对附图进行了参考,附图构成了实施例的一部分且在其中作为示例示出了可在其中实践本发明的各特定实施例。可以理解,可以使用其它实施例并且可以做出结构上或逻辑上的改变而不背离本发明的范围。因此,以下详细描述并不旨在限制,并且本发明的范围由所附权利要求来限定。

[0018] 一个实施例提供了用于按线程、按实例的数据的存储的数据存储应用程序,但此处所描述的技术和方法还满足除这些以外的其他目的。在一个实现中,此处所描述的一个或多个技术可被实现为诸如微软® .NET框架等框架程序内的、或在任何其它类型的程序或服务内的特征。

[0019] 现有解决方案提供了全局数据(即,静态数据)以及与特定线程相关联的数据(即,线程本地数据、或线程静态数据、或按线程的数据)的数据存储和检索。例如,类可包括静态成员,并且该类的所有实例将共享该静态成员的相同的值(即,静态成员表示全局数据)。对于线程本地变量,每一线程基本上具有其自己的该变量的副本。存在其中数据不仅应按线程地存储而且应按实例地存储的情形。对于按实例的数据,类可包括非静态成员,并且该类的每一单独实例可存储该非静态成员的不同的值。在托管框架中用于支持按线程、按实例的数据存储的现有机制常常是极其昂贵的。一个实施例提供了利用按线程的存储机制并且具有与其同等的性能的按线程、按实例的数据存储的解决方案。一个实施例提供了使用线程静态特性的本地线程且基于实例的存储的系统和方法。

[0020] 在微软® .NET框架中,System.ThreadStaticAttribute(系统线程静态属性)提供了快速且支持即时(Just in Time, JIT)编译器的机制,用于存储静态变量中的按线程的数据。当线程静态属性被置于静态成员上时,该成员被作为线程本地成员来对待。另外,在.NET框架中的通用类型或类为用于实例化通用实例的每一通用参数类型集合维护一单独的静态字段。一个实施例利用了这些事实,利用包含ThreadStatic(线程静态)静态字段的通用持有者实例,并且在运行时以实现快速的按线程、按实例的存储的方式来动态地参数化这些通用持有者实例。在一个实施例中,通用类型实例化被用作实现按实例、按线程的数据存储的机制。在一个实施例中,通用类型静态数据持有者被回收以供将来的实例使用,并且当达到通用限制时系统回退到用于按实例、按线程的数据存储的现有的较慢的机制。

[0021] 一个实施例是基于.NET执行环境中的两个事实:(1)当ThreadStaticAttribute(线程静态属性)被应用于静态字段时,该字段对访问它的每一线程维护一不同的值;以及(2)在通用类型中,静态字段应用于用于参数化该通用类型的特定的类型集合。例如,考虑在以下伪码示例I中给出的类型或类:

[0022] 伪码示例 I

```
[0023] public class MyHolder<T>{public static string Data ;}  
[0024] public class FirstType {}  
[0025] public class SecondType {}
```

[0026] 对于伪码示例I中给出的类型,字段MyHolder<FirstType>.Data将不同于字段MyHolder<SecondType>.Data。在一个实施例中,ThreadStaticAttribute被应用于通用类型的静态成员,并且随后该通用类型用多个T参数来参数化。对于所提供的每一新类型参数,获得不同的ThreadStatic字段。在一个实施例中,创建该通用的参数化类型的实例,所创建的每一ThreadLocal(线程本地)实例具有一不同的参数集。尽管ThreadStatic

成员是静态的且由该类型的所有实例所共享,但一个实施例给予每个 ThreadLocal 实例该通用持有者的独特的且没有其他 ThreadLocal 实例共享的参数化版本,由此,实际上使得 ThreadStatic 静态特性改为是实例成员。

[0027] 图 1 是示出根据一个实施例的适于执行按线程、按实例的数据存储应用程序的计算设备 100 的图示。在所示的实施例中,计算系统或计算设备 100 包括多个处理单元 102 和系统存储器 104。取决于计算设备的精确配置和类型,存储器 104 可以是易失性(如 RAM)、非易失性(诸如 ROM、闪存等)或两者的结合。

[0028] 计算设备 100 还可具有附加的特征 / 功能。例如,计算设备 100 还可包含附加存储(可移动和 / 或不可移动),包括但不限于磁盘、光盘或磁带。这样的附加存储在图 1 中由可移动存储 108 和不可移动存储 110 示出。计算机存储介质包括以用于存储诸如计算机可读指令、数据结构、程序模块或其他数据等的任何合适的方法或技术实现的易失性和非易失性、可移动和不可移动介质。存储器 104、可移动存储 108 和不可移动存储 110 都是计算机存储介质的示例(例如,存储用于执行方法的计算机可执行指令的计算机可读存储介质)。计算机存储介质包括,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、磁带盒、磁带、磁盘存储或其它磁性存储设备、或能用于存储所需信息且可以由计算设备 100 访问的任何其它介质。任何这样的计算机存储介质都可以是计算设备 100 的一部分。

[0029] 计算设备 100 包括允许计算设备 100 与其它计算机 / 应用程序 115 通信的一个或多个通信连接 114。计算设备 100 还可包括诸如键盘、定点设备(例如,鼠标)、笔、语音输入设备、触摸输入设备等的输入设备 112。计算设备 100 还可包括诸如显示器、扬声器、打印机等的输出设备 111。

[0030] 在一个实施例中,计算设备 100 包括按线程、按实例的数据存储应用程序 200。数据存储应用程序 200 在以下参考图 2 进一步详细描述。

[0031] 图 2 是根据一个实施例的用于在图 1 所示的计算设备 100 上操作的按线程、按实例的数据存储应用程序 200 的图示。应用程序 200 是驻留在计算设备 100 上的应用程序之一。然而,应用程序 200 可另选地或另外地被具体化为一个或多个计算机上的计算机可执行指令和 / 或与图 1 所示的不同变型。另选地或另外地,应用程序 200 的一个或多个部分可以是系统存储器 104 的一部分、可以在其它计算机和 / 或应用程序 115 上、或可以是计算机软件领域的技术人员想到的其它合适的此类变型。

[0032] 数据存储应用程序 200 包括程序逻辑 202,其负责执行在此描述的一些或全部技术。程序逻辑 202 包括用于实例化线程本地对象的逻辑 204;用于标识索引值的逻辑 206;用于基于所标识的索引值来标识类型参数的逻辑 208;用于基于所标识的类型参数来实例化通用持有者对象的逻辑 210;用于创建哑类型的数组的逻辑 212;用于确定实例的总数是否超出预定阈值的逻辑 214;用于通过向线程分配数据槽来提供按线程、按实例的数据存储的逻辑 216;以及用于操作该应用程序的其他逻辑 218。

[0033] 现在转向图 3-6,将更详细地描述用于实现数据存储应用程序 200 的一个或多个实施例的技术。在某些实现中,图 3-6 示出的技术至少部分地在计算设备 100 的操作逻辑中实现。

[0034] 图 3 是根据一个实施例的存储按线程、按实例的数据的方法 300 的流程图。在方

法 300 中的 302, 标识对应于第一实例的独特索引值。在 304, 基于所标识的索引值来标识类型参数。在一个实施例中, 所标识的类型参数对第一实例是独特的, 并且不被其他实例同时共享 (即, 根据一个实施例, 其他实例可在稍后使用相同的参数, 只要没有其他实例在相同时间正同时使用这些参数)。在 306, 基于所标识的类型参数来实例化通用持有者对象, 其中通用持有者对象包括线程本地字段, 该字段被配置成存储对于第一实例是本地的按线程的数据。根据一个实施例, 线程本地字段通过将线程静态属性应用于静态变量来创建。在一个实施例中, 方法 300 中的标识独特索引值 (302)、标识类型参数 (304)、以及实例化通用持有者对象 (306) 由至少一个处理器来执行。

[0035] 图 4 是根据另一实施例的存储按线程、按实例的数据的方法 400 的流程图。在方法 400 中的 402, 创建哑类型的数组。在 404, 实例化线程本地对象, 由此生成第一实例。在 406, 标识对应于第一实例的独特索引值。在 408, 基于所标识的索引值从哑类型的数组中标识类型参数。在一个实施例中, 所标识的类型参数对于第一实例是独特的, 并且不被其他实例同时共享。在 410, 基于所标识的类型参数来实例化通用持有者对象, 其中通用持有者对象包括线程本地字段, 该字段被配置成存储对于第一实例是本地的按线程的数据。在 412, 对多个实例中的每一实例重复方法 400 中的实例化线程本地对象 (404)、标识独特索引值 (406)、标识类型参数 (408)、以及实例化通用持有者对象 (410)。在 414, 确定多个实例中的实例总数是否超出预定阈值。在 416, 当实例的总数超出预定阈值时, 将数据槽分配给线程用于存储按线程、按实例的数据。

[0036] 在一个实施例中, 方法 400 中的第一实例包括引用通用持有者对象的第一字段以及存储索引值的第二字段。根据一个实施例, 方法 400 中的线程本地字段通过将线程静态属性应用于静态变量来创建。

[0037] 图 5 是根据一个实施例的示出标识独特索引值 (例如, 分别在图 3 和 4 中的框 302 和 406) 的方法 500 的流程图。在方法 500 中的 502, 提供用于存储索引值的静态变量。在 504, 每次生成新实例时, 如果数据结构 (见 508) 不包含任何索引值, 则更新静态变量。在一个实施例中, 如果数据结构包含至少一个索引值, 则在 504 不更新静态变量。在 506, 标识先前使用但不再使用的索引值。在 508, 将所标识的先前使用的索引值存储在数据结构中。在一个实施例中, 在调用实例最终化器时将索引值存储在数据结构中, 这在无用信息收集器检测到该对象不再被使用并且收集该对象时发生。还可在用户通过调用其 Dispose (处置) 方法来明确地指示 ThreadLocal<T> 将不再被使用时, 将索引值存储在数据结构中。在 510, 如果数据结构当前正存储至少一个索引值, 则从该数据结构中选择独特索引值。在 512, 如果数据结构当前没有正存储至少一个索引值, 则基于静态变量选择独特索引值。在一个实施例中, 该独特索引值是以 10 为基数的数字。在 514, 将表示独特索引值的以 10 为基数的数字转换成具有不同基数的数字 (例如, 以 16 为基数的数字)。根据一个实施例, 所选择的基数是基于用于参数化通用持有者的类型的数量来预测的 (例如, 如果使用 16 个哑类型来参数化, 则将选择基数 16)。在 516, 基于具有不同基数的数字来标识类型参数。

[0038] 现在将参考伪码示例更详细地描述图 3-5 所示的且以上所讨论的方法。在以下示例中, 上述线程本地对象与 “ThreadLocal<T>” 类或类型的实例相对应。在一个实施例中, ThreadLocal<T> 的实例存储若干字段, 包括在以下伪码示例 II 中所示的两个 :

[0039] 伪码示例 II

[0040] private HolderBase m_holder;

[0041] private int m_currentInstanceIndex;

[0042] 如伪码示例 II 中所示, ThreadLocal<T> 的每一实例包括 HolderBase(持有者基础) 字段和索引字段。HolderBase 是实际 T 数据生存的地方, 它被包装在装箱 (boxed) 实例中, 如以下伪码示例 III 中所示 :

[0043] 伪码示例 III

[0044]

```
abstract class HolderBase
{
    internal abstract Boxed Boxed { get; set; }
}
```

[0045] HolderBase 是用于允许 ThreadLocal<T> 为数据使用多个存储机制的抽象类。例如, 从 HolderBase 派生的类可以使用 Thread.GetData/SetData(线程取得数据 / 设置数据) 方法来存储 T, 如以下伪码示例 IV 所示 :

[0046] 伪码示例 IV

[0047]

```
sealed class TLSHolder : HolderBase
{
    private LocalDataStoreSlot m_slot = Thread.AllocateDataSlot();
```

```
    internal override Boxed Boxed
```

```

}
```

[0048]

```
        get { return (Boxed)Thread.GetData(m_slot); }
```

```
        set { Thread.SetData(m_slot, value); }
```

```

}
```

```
}
```

[0049] 一个实施例使用从 HolderBase 派生的通用类型来提供更快的实现, 如以下伪码示例 V 所示 :

[0050] 伪码示例 V

[0051]

```
sealed class GenericHolder<U, V, W> : HolderBase
{
    [ThreadStatic]
    private static Boxed s_value;

    internal override Boxed Boxed
    {
        get { return s_value; }
        set { s_value = value; }
    }
}
```

[0052] 在伪码示例 V 中, 装箱 T 被直接存储到 ThreadStatic 静态成员中, 这比使用伪码示例 IV 中所示的 Thread.GetData/SetData 更快, 因为 CLR(公共语言运行库) 提供了与将数据存储到非 ThreadStatic 静态特性类似的这一功能的优化实现。然而, 每一个 GenericHolder<U, V, W> 仅有一个静态 s_value。基于实例的存储可通过为要使用的每一个独特的 s_value 创建 GenericHolder 的不同通用实例化(具有不同的 U、V 和 W 类型)来提供。在一个实施例中, 这通过检索可用类型的数组来完成(例如, 通过在 ThreadLocal<T> 在其中生存的汇编件(assembly) 上反射, 以及取得该汇编件中的所有引用类型的列表)。在另一实施例中, 这通过创建出于此目的的特定类型集合来实现, 如以下伪码示例 VI 所示:

[0053] 伪码示例 VI

[0054]

```

class C0 { } class C1 { } class C2 { } class C3 { }
class C4 { } class C5 { } class C6 { } class C7 { }
class C8 { } class C9 { } class C10 { } class C11 { }
class C12 { } class C13 { } class C14 { } class C15 { }

```

```

private static Type[] s_dummyTypes =
{
    typeof(C0), typeof(C1), typeof(C2), typeof(C3),
    typeof(C4), typeof(C5), typeof(C6), typeof(C7),
    typeof(C8), typeof(C9), typeof(C10), typeof(C11),
    typeof(C12), typeof(C13), typeof(C14), typeof(C15)
};

```

[0055] 来自伪码示例 VI 中给出的列表中的类型可用于参数化 GenericHolder<U, V, W>。在一个实施例中，基于指定索引来检索来自该列表的类型，如以下伪码示例 VII 所示：

[0056] 伪码示例 VII

[0057]

```

private Type[] GetTypesFromIndex(int index)
{
    Type[] types = new Type[TYPE_DIMENSIONS];

    for (int i = 0; i < TYPE_DIMENSIONS; i++)
    {
        types[i] = s_dummyTypes[index % s_dummyTypes.Length];
        index /= s_dummyTypes.Length;
    }

    return types;
}

```

[0059] 伪码示例 VII 中的索引用于计算对应于每一 U、V、W 的唯一类型组合，随后它可用于实例化 GenericHolder<U, V, W>。在一个实施例中，每个 ThreadLocal<T> 得到它自己的独特索引，这通过为所创建的每个新的 ThreadLocal<T> 维护原子地更新的静态整数变量

来完成。在一个实施例中,如果先前使用但当前不使用的索引值是可用的,或者如果已经超出限制,则不发生静态整数变量的更新。根据一个实施例,伪码示例 VII 中的代码将以 10 为基数的索引值映射到用于标识类型 U、V、W 的集合的以 16 为基数的数字。

[0060] 对于具有三个类型参数(即, U、V、W)的 GenericHolder 的情况, 伪码示例 VII 中的 TYPE_DIMENSIONS(类型维数)的值将是 3。如果改为将 GenericHolder 创建成 GenericHolder<U, V, W, X>, 则 TYPE_DIMENSIONS 的值将是 4。独特索引的数量是 s_dummyTypes.Length(静态哑类型长度)的 TYPE_DIMENSIONS 次幂。由此,对于以上示例,可以存在 GenericHolder<U, V, W> 的 16^3 (即, 4096) 个独特实例。

[0061] GenericHolder 的独特实例的数量可通过添加更多哑类型和 / 或增加类型维度的数量来增加。然而,存在由此引起的某些潜在问题。在某一点,如果创建了足够的 ThreadLocal<T> 实例,则系统将用完可用索引(如以上所暗示的,假定在一个实施例中由于哑类型的数量和类型维度的数量而存在硬编码的限制)。在一个实施例中,这通过从不再使用的 ThreadLocal<T> 实例中集中这些索引来解决。当 ThreadLocal<T> 不再被使用时(这可通过最终化或通过实例的明确处置来检测),其索引被添加到维护所有先前使用但当前不使用的索引的共享的线程安全数据结构(例如,并发栈)中。当新的 ThreadLocal<T> 被实例化时,首先检查该数据结构以查看是否存在任何这样的可用索引,以及如果存在,则取得它并且使用它。如果不存在任何这样的可用索引,则检查先前引用且为所创建的每个新的 ThreadLocal<T> 实例原子地更新的共享整数变量,并且只要该变量的值低于最大值,则该值被递增并且被用作该新实例的索引。

[0062] 由于使运行时环境实例化以不同方式且为不同的 T 值所参数化的太多 GenericHolder<U, V, W> 实例所引起的可能的存储器消耗问题,根据一个实施例的系统还对跨所有类型所创建的通用持有者总数实施最大值。这在一个实施例中通过维护另一全局计数来实现,该全局计数用于指示已经执行了多少独特的通用实例化。在一个实施例中,如果全局计数的值达到最大值,则系统回退到不是基于通用实例化的较慢的实现(在一个实施例中,如果系统可得到未使用的池以外的索引,则该系统仍将使用较快的实现),

[0063] 根据一个实施例,在回退的情况下使用 TLSHolder,如伪码示例 IV 所示。这启用无限数量的 ThreadLocal<T> 实例(至多达由于诸如存储器等系统资源所允许的隐式最大值),其中,大多数实例基于通用实例化在快路径上执行,但在太多 ThreadLocal<T> 实例都被创建为同时运行的情形下,那些剩余的仍可起作用,但在一个实施例中将采用较慢方案。

[0064] 在一个实施例中, HolderBase(从其中派生 TLSHolder 和 GenericHolder<U, V, W>) 存储装箱实例,它包装 T 实例而不是直接存储 T。这样做存在几个原因。首先,访问实例上的字段比访问静态、ThreadStatic 或 TLS 数据(Thread.GetData/SetData) 快。在 ThreadLocal<T> 内部的单个操作可能需要采用对后台数据源的多个读 / 写的情况下,从持有者一次检索装箱值并且随后读 / 写其 T 字段、而不是迫使所有的读 / 写操作直接针对 HolderBase 进行是更高效的。其次,由于在一个实施例中所执行的 GenericHolder<U, V, W> 实例的集中 / 重用, ThreadLocal<T> 的两个不同实例将最终在应用程序的生存期使用相同的 U、V、W 参数集是可能的。如果发生这一情况,并且如果相同的线程最终访问两者,则线程将最终看到两者的相同的 ThreadStatic 状态,因为它是相同的底层 GenericHolder<U, V, W> 类型。由此,装箱实例还维护到当前与其相关联的 ThreadLocal<T> 的引用。在

一个实施例中,当代码访问装箱实例的 T 值时,它还检查以查看装箱是否包含最新的 ThreadLocal<T> 实例引用。如果不包含最新引用,这意味着装箱实例是陈旧的,并且在一个实施例中被作为未初始化的来对待,而不是重用找到的现有的值。根据一个实施例,装箱类的示例在以下伪码示例 VIII 中给出:

[0065] 伪码示例 VIII

[0066]

```
class Boxed
```

[0067]

```
{  
    internal T Value;  
    internal HolderBase m_ownerHolder;  
}
```

[0068] 一个实施例提供了对初始化存储在 ThreadLocal<T> 实例中的数据的内置初始化支持。考虑在以下伪码示例 IX 中给出的代码:

[0069] 伪码示例 IX

[0070] [ThreadStatic]

[0071] static string s_myValue = ComputeValue();

[0072] 示例 IX 中的代码创建了线程静态变量 s_myValue,这意味着每个线程将得到存储字符串的其自己的 s_myValue 副本。另外, s_myValue 被初始化为运行 ComputeValue(计算值)方法的结果。然而,这在 C# 中工作的方式是编译器生成代码使得 ComputeValue 仅被求值一次,这意味着其求值和存储到 s_myValue 仅为一个线程而发生,并不是为系统中的任何其他线程发生。这使得难以初始化 s_myValue。诸如以下伪码示例 X 中给出的代码可以在对 s_myValue 的每次访问之前被执行,以便确保它已经被正确地初始化:

[0073] 伪码示例 X

[0074] if(s_myValue == null) s_myValue = ComputeValue();

[0075] 一个实施例通过允许开发者向 ThreadLocal<T> 提供在每个线程上被调用以便初始化该值的委托(即,表示函数的对象)来解决此初始化问题。一示例在以下伪码示例 XI 中给出:

[0076] 伪码示例 XI

[0077] static ThreadLocal<string> s_myValue = new ThreadLocal<string>(ComputeValue);

[0078] 示例 XI 中的代码创建了 ThreadLocal<string> 实例,它确保访问 s_myValue.Value 的每个线程都将使它首先被初始化为运行 ComputeValue 的结果。

[0079] 图 6 是根据一个实施例的生成用于存储按线程、按实例的数据的持有者的方法 600 的流程图。在方法 600 中的 602,生成 ThreadLocal<T> 并将其分配给变量“t”。在 604,确定是否存在可用的先前使用的通用索引。如果在 604 确定了存在可用的先前使用的通用索引,则在 606 保留下一未使用的索引。在 608,将先前使用的通用索引转换成通用类型集

合 (U、V、W...)，并且生成 GenericHolder<U, V, W> 的实例。在 618，将 GenericHolder<U, V, W> 实例存储到 t.m_holder。

[0080] 如果在 604 确定不存在可用的先前使用的通用索引，则该方法 600 移至 610，在那里确定从按 T 的整数中是否可获得未使用的索引。如果在 610 确定未使用的索引可用，则接着在 612 确定是否达到最大全局（即，非按 T 的）通用实例化阈值。如果在 612 确定未达到最大全局阈值，则该方法 600 移至 614，在那里从按 T 的整数中分配新索引并且更新全局通用实例化计数，并且该方法 600 随后移至 608（如上所述）。

[0081] 如果在 610 确定未使用的索引不可用，或如果在 612 确定已经达到最大全局阈值，则该方法 600 移至 616。在 616，该方法 600 退回到慢路径以使用 TLSHolder 来提供按线程、按实例的存储，并且该方法 600 随后移至 618（如上所述）。

[0082] 尽管此处说明并描述了具体实施例，但本领域技术人员可以理解，可用各种替换和 / 或等价实现来替换此处示出并描述的具体实施例而不背离本发明的范围。本申请旨在覆盖此处讨论的具体实施例的任何改编或变型。因此，本发明旨在仅由权利要求书及其等效方案来限制。

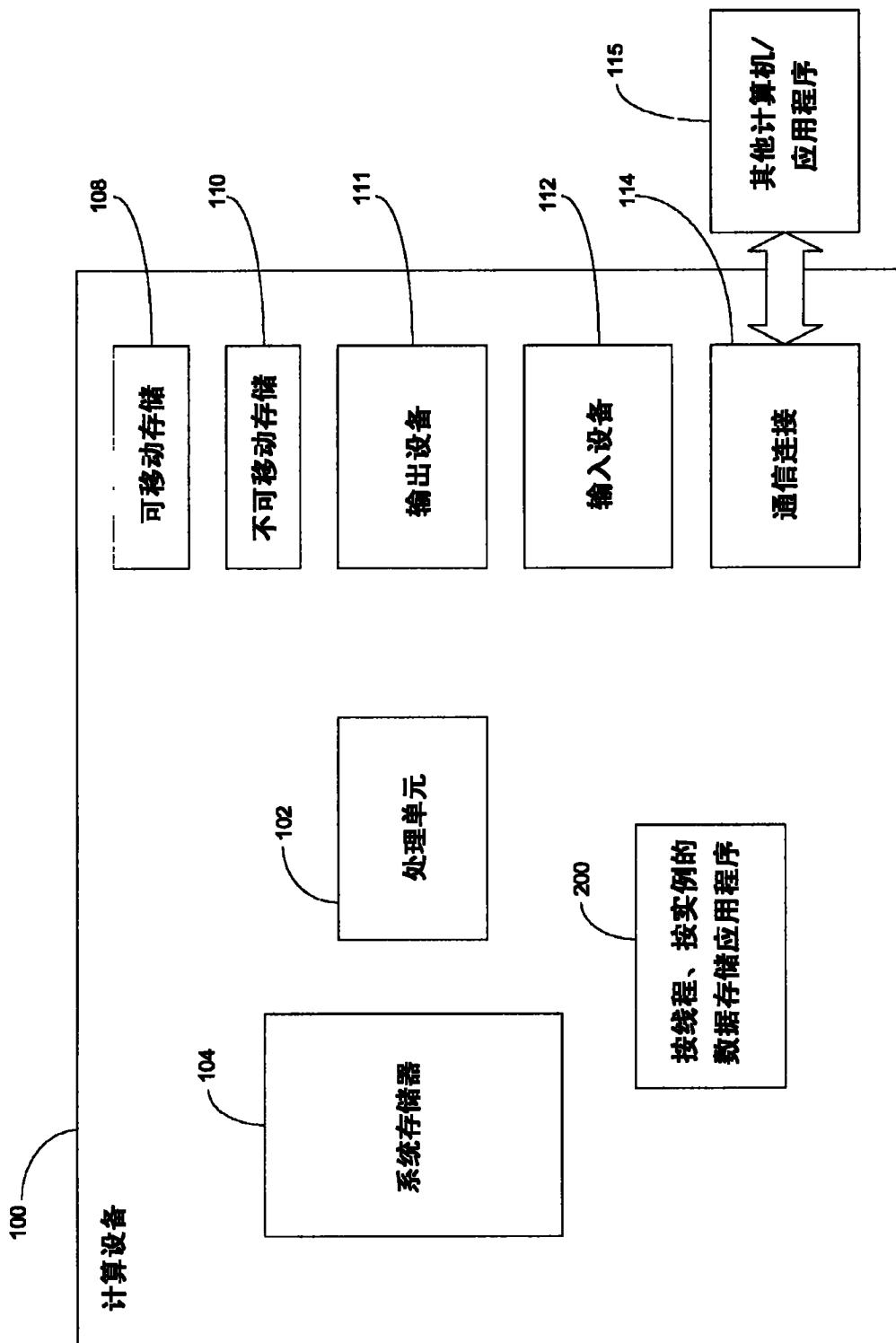


图 1

按线程、按实例的数据存储应用程序 200	程序逻辑 202
用于实例化线程本地对象的逻辑 204	
用于标识索引值的逻辑 206	
用于基于所标识的索引值来标识类型参数的逻辑 208	
用于基于所标识的类型参数来实例化通用持有者对象的逻辑 210	
用于创建哑类型数组的逻辑 212	
用于确定实例总数是否超出预定阈值的逻辑 214	
用于通过向线程分配数据槽来提供按线程、按实例的数据存储的逻辑 216	
用于操作应用程序的其他逻辑 218	

图 2

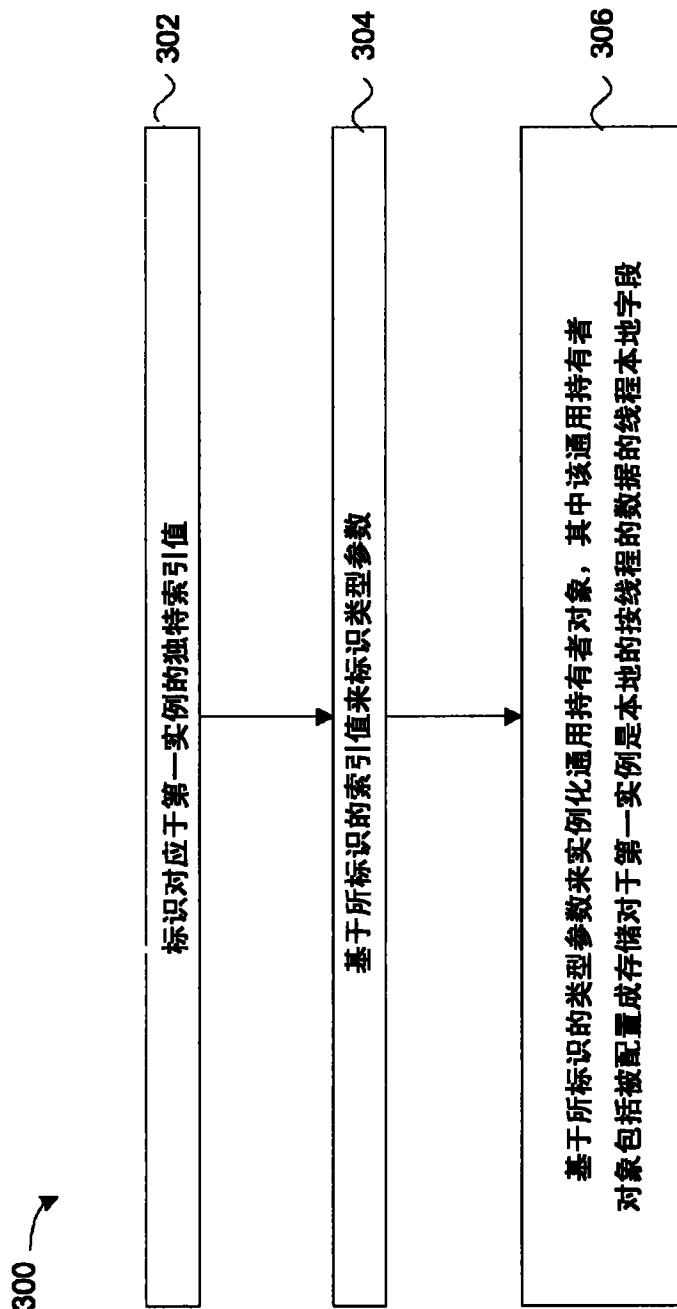


图 3

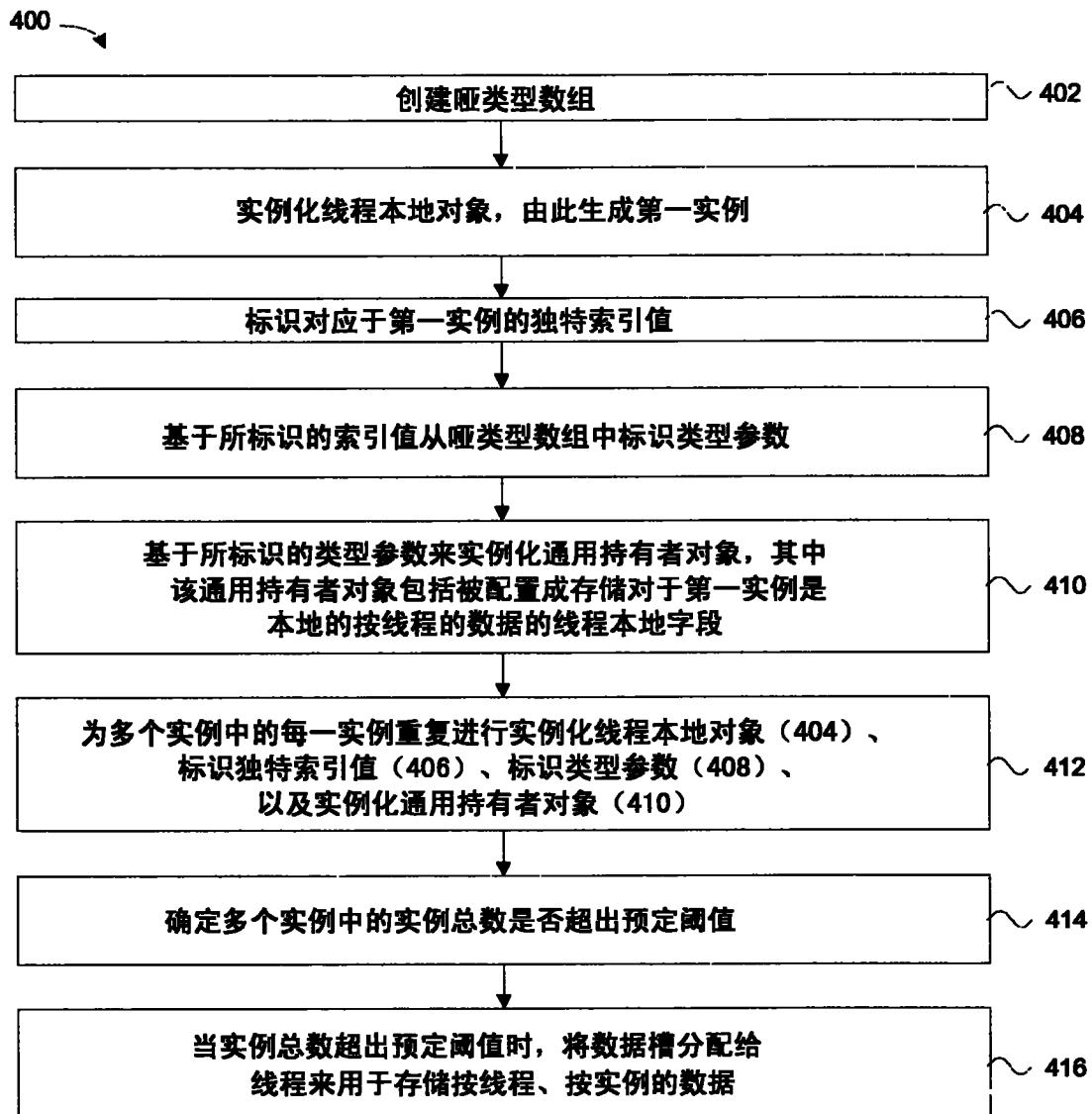


图 4

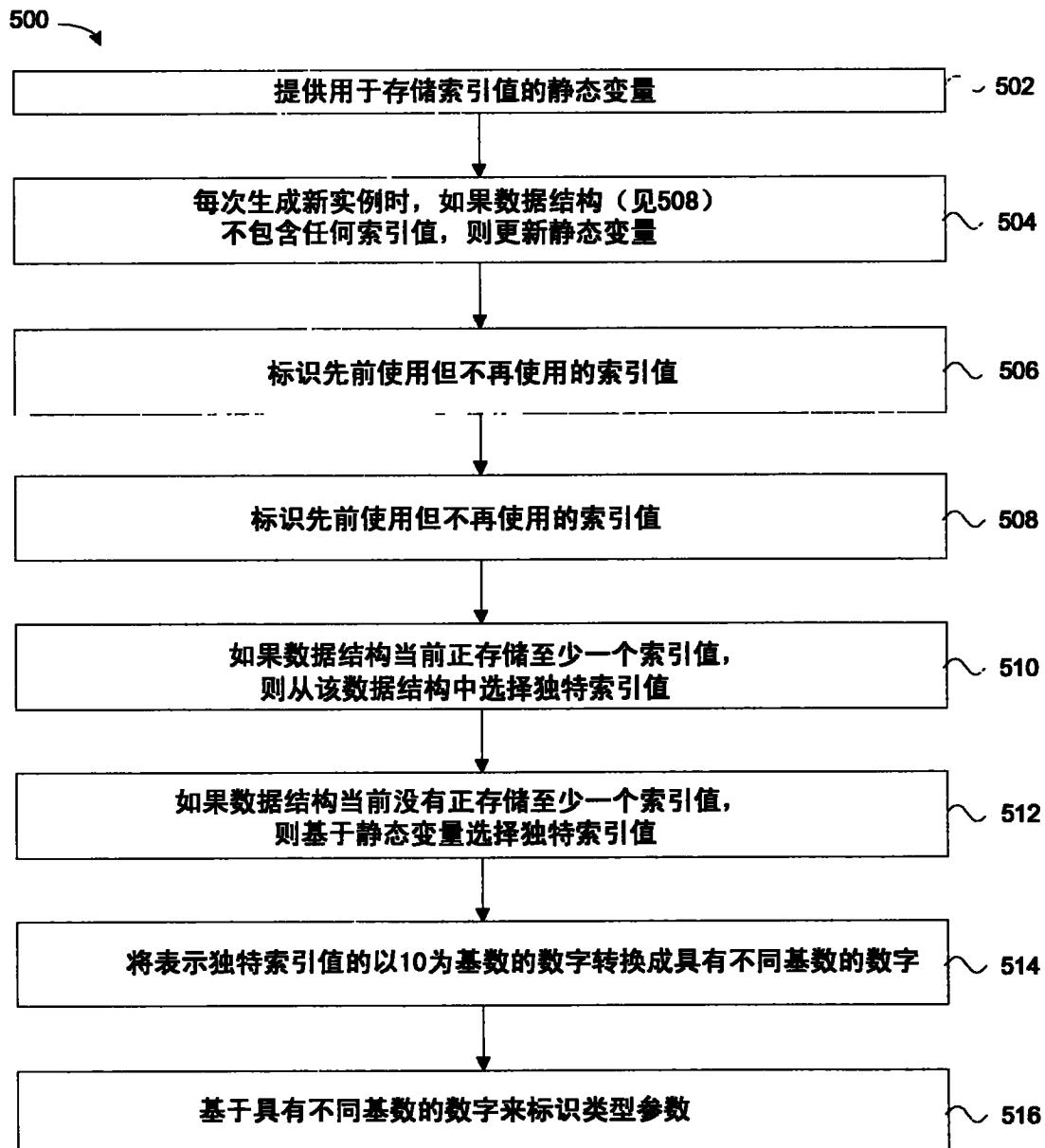


图 5

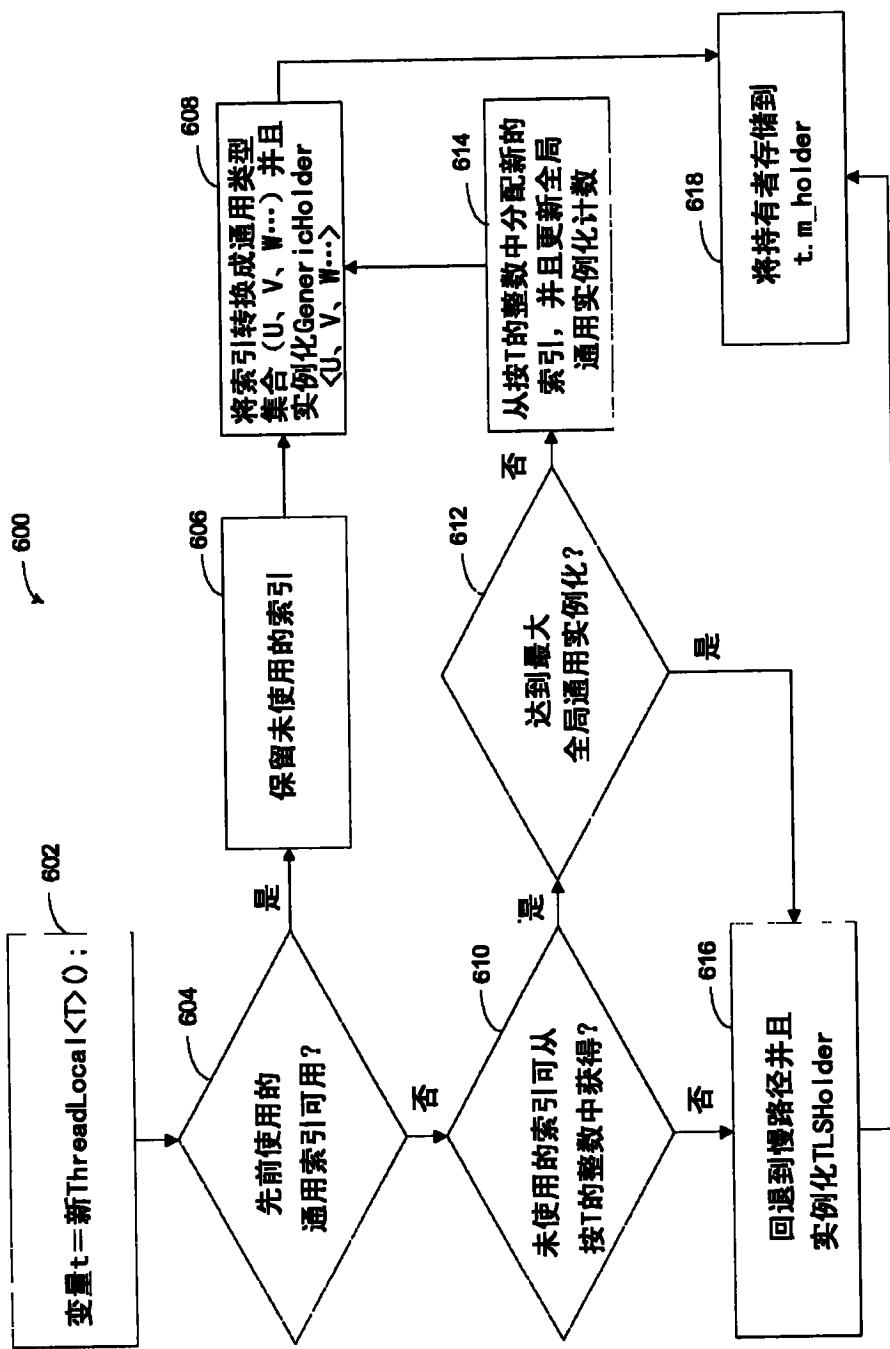


图 6