(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 103140834 A
(43)申请公布日 2013.06.05

(21)申请号 201180047474.6
(22)申请日 2011.08.03
(30)优先权数据
12/849,724 2010.08.03 US
(85)PCT申请进入国家阶段日
2013.03.29
(86)PCT申请的申请数据
PCT/US2011/046412 2011.08.03
(87)PCT申请的公布数据
WO2012/018906 EN 2012.02.09
(71)申请人 超威半导体公司
地址 美国加利福尼亚州
(72)发明人 埃里克·R·卡斯波尔
劳伦特·莫里凯蒂
(74)专利代理机构 上海胜康律师事务所 31263
代理人 李献忠
(51)Int.Cl.
G06F 9/50(2006.01)
G06F 12/02(2006.01)

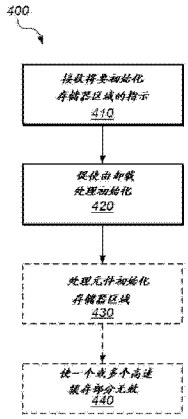
权利要求书2页 说明书12页 附图6页

(54)发明名称

用于填充存储器区域的处理器支持

(57)摘要

公开了涉及在处理器和/或处理元件之间分配工作负载的技术。具有至少第一和第二处理元件的计算机系统可促使初始化一个或多个存储器区域的请求被所述第二处理元件处理。可由所述第二处理元件直接访问包括将要初始化的指定存储器区域的存储器来完成初始化。因此，虽然所述第二处理元件促使所述存储器区域被初始化，但是所述第一处理元件能自由执行其它计算任务。由于所述第二处理元件执行所述初始化，所以可以不打扰与所述第一处理元件相关的高速缓存，这可避免数据从所述高速缓存移位。



CN 103140834 A

1. 一种非瞬时性计算机可读介质，其具有存储在其上的可由计算设备的至少第一处理元件执行以执行操作的程序指令，所述操作包括：

响应于将要初始化的所述计算设备的存储器区域的指示，促使所述计算设备的第二处理元件处理所述存储器区域的初始化。

2. 根据权利要求 1 所述的非瞬时性计算机可读介质，其中由所述第一处理元件执行的控制程序从第一程序接收所述存储器区域的指示。

3. 根据权利要求 2 所述的非瞬时性计算机可读介质，其中所述控制程序正在执行所述第一程序。

4. 根据权利要求 2 所述的非瞬时性计算机可读介质，

其中所述指示指定对应于可通过所述控制程序操作的一个或多个数据对象的存储器的一个或多个存储器区域；并且

其中所述操作还包括填充所述一个或多个存储器区域的所有内容。

5. 根据权利要求 4 所述的非瞬时性计算机可读介质，其中所述操作还包括：

作为垃圾回收进程的部分，所述控制程序生成将要初始化的存储器区域的多个指示；和

促使所述第二处理元件处理所述多个存储器区域的初始化，其中所述初始化包括用由编程语言规范指定的默认内容填充所述多个存储器区域的所有内容。

6. 根据权利要求 2 所述的非瞬时性计算机可读介质，其中所述控制程序包括存储在所述非瞬时性计算机可读介质上的一个或多个库文件，且其中所述控制程序接收所述指示包括所述控制程序通过应用编程接口（API）接收所述指示。

7. 根据权利要求 1 所述的非瞬时性计算机可读介质，其中所述促使初始化包括动态生成可由所述第二处理元件执行以改变所述存储器区域的内容的一个或多个指令集的至少部分。

8. 根据权利要求 1 所述的非瞬时性计算机可读介质，其中促使所述第二处理元件处理所述存储器区域的初始化不会促使所述计算机系统的高速缓存存储所述初始化的存储器区域的初始化后的内容；

其中所述高速缓存被配置来响应于所述第一处理元件访问包括所述存储器区域的所述计算机系统的存储器而存储所述存储器区域的内容。

9. 根据权利要求 1 所述的非瞬时性计算机可读介质，其还包括可执行以促使生成所述第一和第二处理元件中的至少一个的程序指令。

10. 一种方法，其包括：

响应于将要初始化的存储器区域的指示，在第一处理元件上执行的第一程序促使第二处理元件处理所述存储器区域的初始化，其中计算设备包括所述第一和第二处理元件和包括所述存储器区域的存储器。

11. 根据权利要求 10 所述的方法，其还包括所述第二处理元件使用直接存储器存取（DMA）来初始化所述存储器区域，而所述第一处理元件不直接访问所述存储器区域。

12. 根据权利要求 10 所述的方法，其还包括所述第一程序中的垃圾回收进程产生所述指示。

13. 根据权利要求 10 所述的方法，其中所述第一程序是控制程序，所述方法还包括所

述第二处理元件根据所述控制程序的一个或多个启发式规则初始化所述存储器区域。

14. 根据权利要求 10 所述的方法，其还包括：

响应于所述存储器区域被初始化，所述计算设备使所述计算设备的数据高速缓存的一个或多个部分无效；

其中所述一个或多个无效部分对应于在所述存储器区域初始化之前所述存储器区域的内容。

15. 一种计算机系统，其包括：

存储器子系统，其包括主存储器；

二级存储装置；和

至少第一和第二处理元件；

其中所述二级存储装置具有存储在其上的可由所述第一处理元件执行以促使所述计算机系统执行如下操作的程序指令：

响应于将要初始化的所述主存储器的存储器区域的指示，促使所述第二处理元件处理所述存储器区域的初始化。

16. 根据权利要求 15 所述的计算机系统，其中所述第一和第二处理元件是异构的。

17. 根据权利要求 15 所述的计算机系统，其还包括：

与所述第一处理元件相关联的高速缓存，其中所述高速缓存被配置来响应于所述第一处理元件访问所述主存储器而存储所述主存储器的内容；且

其中所述促使所述存储器区域的初始化不会导致所述高速缓存存储所述存储器区域的初始化后的内容。

18. 根据权利要求 15 所述的计算机系统，其还包括：

存储器存取控制器，其被配置来对所述第二处理元件提供对所述主存储器的直接访问；

其中促使所述存储器区域的初始化包括所述第二处理元件使用所述存储器存取控制器访问所述存储器区域，且其中促使初始化不包括所述第一处理元件访问所述存储器区域。

19. 一种处理元件，其中所述处理元件包括存储器初始化电路，其被配置来促使第二处理元件处理存储器设备的存储器区域的初始化，其中所述促使初始化响应于所述存储器区域将要被初始化的指示而被执行。

20. 一种非瞬时性计算机可读存储介质，其包括数据结构，所述数据结构可由可在计算机系统上执行以执行制造包括由所述数据结构描述的电路的集成电路的过程的一部分的程序使用，以所述数据结构描述的所述电路包括：

存储器初始化单元，其被配置来促使计算设备的第二处理元件而不是所述计算设备的第一处理元件来处理存储器设备的存储器区域的初始化，其中所述促使初始化响应于所述存储器区域将要被初始化的指示而被执行。

21. 根据权利要求 20 所述的非瞬时性计算机可读存储介质，其中所述存储介质存储 HDL、Verilog 或 GDSII 数据中的至少一项。

用于填充存储器区域的处理器支持

技术领域

[0001] 本公开涉及计算机处理器，且更明确地涉及接收请求以填充存储器区域的处理器。

背景技术

[0002] 在计算机操作期间，存储器区域可能需要用某些值初始化(填充)。初始化存储器区域会占用某些计算资源－例如，执行初始化的处理器可能必须将值写入一系列存储器位置中，这可能耗时。在这种初始化期间，处理器可能无法执行其它计算任务。

[0003] 此外，存储器初始化操作对于与处理器相关联的高速缓存可能是破坏性的。随着高速缓存内容在存储器初始化期间被移位，高速缓存性能可受到处理器的负面影响。例如，高速缓存的一些或所有预先存在的内容(在开始初始化存储器区域之前)可能将被正在初始化的存储器区域的内容置换。这种置换可减慢程序执行，因为可随后访问其它存储器以检索先前存在于高速缓存中的数据。

发明内容

[0004] 本文中公开了允许计算机系统或计算设备从第一处理元件到第二处理元件分配某些存储器操作的方法和结构的各个实施方案。

[0005] 在所述的一个实施方案中，公开了一种计算机可读介质，其具有存储在其上的可由计算设备的至少第一处理元件执行以执行操作的程序指令，所述操作包括接收将要初始化的计算设备的存储器区域的指示，且响应于所述接收，促使计算设备的第二处理元件处理存储器区域的初始化。在其它实施方案中，从由第一处理元件执行的控制程序接收指示。

[0006] 另一实施方案包括一种方法，其包括接收将要初始化的计算设备的存储器区域的指示的第一程序，其中正在计算设备的第一处理元件上执行第一程序，且响应于所述接收，第一程序促使由计算设备的第二处理元件处理存储器区域的初始化。在其它实施方案中，第二处理元件使用直接存储器存取(DMA)来初始化存储器区域，而第一处理元件不直接访问存储器区域。

[0007] 又一实施方案是一种计算机系统，其包括存储器子系统，所述存储器子系统包括主存储器、二级存储装置和至少第一和第二处理元件，其中二级存储装置具有存储在其上的可由第一处理元件执行以促使计算机系统接收将要初始化的存储器区域的指示的程序指令，其中存储器区域位于主存储器中，且响应于所述接收，促使计算设备的第二处理元件处理存储器区域的初始化。在其它实施方案中，计算机系统包括与第一处理元件相关联的高速缓存，其中高速缓存被配置来响应于第一处理元件访问主存储器而存储主存储器的内容，且其中促使存储器区域的初始化不会导致高速缓存存储存储器区域初始化后的内容。

附图说明

[0008] 图 1 是图示被配置来从第一处理元件到第二处理元件分配存储器初始化的计算

机系统的一个实施方案的方框图。

[0009] 图 2A 至图 2B 是描绘初始化之前和之后的示例性存储器区域的方框图。

[0010] 图 3A 是图示包括被配置来执行存储器初始化的控制程序的存储器子系统的实施方案的方框图。

[0011] 图 3B 是图示包括被配置来执行存储器初始化的操作系统的存储器子系统的实施方案的方框图。

[0012] 图 3C 是包括被配置来执行存储器初始化的 JAVA 虚拟机程序的实施方案的方框图。

[0013] 图 4 是图示其中从第一处理元件到第二处理元件分配存储器初始化的方法的一个实施方案的流程图。

[0014] 图 5 是图示其中从第一处理元件到第二处理元件分配存储器初始化的计算机系统的另一实施方案的方框图。

具体实施方式

[0015] 本说明书包括对“一个实施方案”或“实施方案”的引用。短语“在一个实施方案中”或“在实施方案中”的出现不一定指相同实施方案。可以与本公开一致的任何适当方式组合特定特征、结构或特性。

[0016] 术语。以下段落提供本了公开内容(包括随附权利要求)中找到的术语的定义和 / 或背景。

[0017] “包括。”这些术语是开放式的。如在随附权利要求中所使用的,这些术语不排除附加结构或步骤。考虑如下叙述的权利要求 :“一种设备,其包括一个或多个处理元件…”这种权利要求不排除所述装置包括附加组件(例如,网络接口单元、图形电路,等等)。

[0018] “被配置来。”各种单元、电路或其它组件可被描述或宣称为“被配置来”执行一项任务或多项任务。在这种上下文中,“被配置来”用于通过指示单元 / 电路 / 组件包括在操作期间执行这些一项任务或多项任务的结构(例如,电路)来暗指结构。因而,所述单元 / 电路 / 组件可以说是被配置来即使指定单元 / 电路 / 组件当前不可操作时(例如,没有开启)也执行任务。与“被配置来”的语言一起使用的单元 / 电路 / 组件包括硬件 - 例如电路、存储可执行以实施操作的程序指令的存储器,等等。单元 / 电路 / 组件“被配置来”执行一项或多项任务的叙述明确地不是旨在对于所述单元 / 电路 / 组件引用 35U.S.C. § 112 第六段。此外,“被配置来”可包括由软件和 / 或固件(例如,FPGA 或通用处理器执行软件)操纵的通用结构(例如,通用电路)来以能够执行发出的任务的方式操作。此外,“被配置来”可包括调适制程(例如,半导体制造设施)以制造被调适来实施或执行一项或多项任务的装置(例如,集成电路)。

[0019] “处理元件。”这个术语在所属领域中具有其普通和接受的意义,且包括能够执行计算机指令的装置(例如,电路)或装置组合。在各个实施方案中,处理元件可以指单核处理器、多核处理器的核,或多核处理器的双核或多核组。

[0020] “处理器。”这个术语在所属领域中具有其普通和接受的意义,且包括装置,其包括一个或多个处理元件。在没有限制的情况下,处理器可以指中央处理单元(CPU)、协处理器、算术处理单元、图形处理单元、数字信号处理器(DSP),等等。

[0021] “第一”、“第二”等等。如本文中所使用，这些术语用作其之后的名词的标签，且不暗示任何类型的顺序(例如，空间、时间、逻辑，等等)。例如，在具有八个处理元件或核的处理器中，术语“第一”和“第二”处理元件可用于指八个处理元件的任何两个。换句话说，“第一”和“第二”处理元件不限于逻辑处理元件 0 和 1。

[0022] “计算机”或“计算机系统。”这个术语在所属领域中具有其普通和接受的意义，且包括一起操作的一个或多个计算设备和存储于其上的任何软件。计算设备包括一个或多个处理元件和存储器子系统。存储器子系统可存储可由一个或多个处理元件执行以执行各种任务的程序指令。

[0023] “计算机可读介质。”如本文中所使用，这个术语指(非瞬时性、有形)介质，其可被计算机或计算机系统读取，且包括磁性、光学和固态存储介质，诸如硬盘驱动器、光盘、DVD、易失性或非易失性 RAM 装置、全息存储器、可编程内存，等等。如本文中应用于计算机可读介质的术语“非瞬时性”只是旨在从权利要求的范畴排除被认为不符合 35U.S.C. §101 的任何标的，诸如瞬时性(无形的)介质(例如，载波)，且并非旨在排除另外被认为是法定的任何标的。

[0024] “操作系统。”这个术语在所属领域中具有其普通和接受的意义，且包括(例如，响应于来自应用的请求)控制访问计算机系统的资源的程序或程序集。在一些实施方案中，操作系统控制访问 I/O 设备，诸如通信设备、存储设备等等。如本文中所述，在某些实施方案中，操作系统可包括可执行以促使第二处理元件执行存储器初始化的指令。

[0025] “高速缓存。”这个术语在所属领域中具有其普通和接受的意义，且包括存储器或存储数据的其它存储器，且可通过提供相对于一些其它存储器或存储的更快访问而改进对于这些数据的未来请求。

[0026] “促使计算机系统执行操作。”程序指令的执行可被描述或宣称为“促使计算机系统执行操作。”所述短语被广义地解译，涵盖当执行时执行在询问中的操作的指令，以及安装或实例化当执行时执行操作的代码的指令。例如，计算机可读介质可包括指令，其可执行以促使计算机系统从计算机系统的第一处理元件到计算机系统的第二处理元件分配存储器区域的存储器初始化。

[0027] “可执行。”这个术语在所属领域中具有其普通和接受的意义，且包括以与一个或多个特定处理元件相关联的格式的指令(即，某一指令集架构(ISA))，但也包括以可被控制程序(例如，JAVA 虚拟机)解译的中间格式(例如，JAVA 字节码)的指令，以对处理元件的 ISA 产生指令。根据这个定义，第一处理元件上“正在执行”的程序使其至少一些指令由所述第一元件执行(虽然所述程序的其它指令可由另一元件执行)。程序的执行也包括程序的解译。

[0028] “应用编程接口(API)”。这个术语在所属领域中具有其普通和接受的意义，且包括使得软件与其它软件相互作用的接口。程序可进行 API 调用以使用应用、库例程、操作系统的功能，等等。

[0029] * * *

[0030] 如本文中所述，计算机系统可能需要用某些数据初始化(填充)计算机存储器，从而擦除事先被所述存储器存储的数据。在一些实施方案中，可根据接收(新)存储器的分配的请求而出现对初始化存储器的需要。在一个实施方案中，JAVA 虚拟机(JVM)程序(用于运

行其它 JAVA 程序)可将存储器区域“置零”使得 JAVA 程序可以空(默认)数据开始使用这些存储器区域。在另一实施方案中,操作系统可例如在允许用户程序访问所述存储器之前用全零覆写存储器。(在一些实施方案中,被擦除的数据可持有密码、信用卡号或操作系统不希望用户程序能够访问的其它数据。)也预期由其它类型的程序的许多其它种类的存储器初始化,且本公开不限于 JVM 或操作系统软件。在初始化期间填充到存储器区域中的数据可以(但不需要)是全零,如下文中进一步描述。

[0031] 在一个实施方案中,计算机系统具有第一处理器,诸如中央处理单元(CPU),其被配置来执行例如通用指令。计算机系统也具有第二处理器,诸如图形处理单元(GPU),其被配置来执行特殊用途的指令,诸如图形指令。在其它实施方案中,第一处理器(或处理元件)可包括单个装置、封装或集成电路中的 CPU 和 GPU 两者的功能。计算机系统也具有存储器子系统。在一个实施方案中,计算机系统被结构化(即,被编程)使得由第二处理器执行某些指令序列。这些指令序列可由第一处理器执行的指令产生,且可包括存储器初始化例程。因而,第一处理器可被释放以执行其它任务,同时第二处理器执行初始化。(例如,可能不是立刻需要将要初始化的存储器区域,所以第一处理器可能能够继续执行程序,同时第二处理器执行存储器初始化。)除了改进第一处理器的性能之外,本文中公开的技术也可以例如通过避免移位来自高速缓存的数据而改进与第一处理器相关联的数据高速缓存的性能。

[0032] 现在转向图 1,描绘了其被配置来从第一处理元件到第二处理元件分配存储器初始化的计算机系统 10 的一个实施方案。计算机系统 10 包括由总线 20 链接的第一处理元件 100A 和第二处理元件 100B。在一个实施方案中,总线 20 允许处理元件 100A 和 100B 访问存储器子系统 60 内的一个或多个存储器区域 64。存储器子系统 60 可包含各种程序 62,其中一些可执行以请求(或促使)使用处理元件 100B 初始化存储器。此外,虽然图 1 中示出为视觉上不同的组件,但是存储器子系统 60 的一部分或全部可形成处理元件 100A、处理元件 100B 的电路的部分,或可以是包括处理元件 100A 和 100B 两者的单个装置的一部分。在一个实施方案中,高速缓存 30 可访问处理元件 100A,且被配置来存储对应于在存储器子系统 60 中存储的数据的数据。在一个实施方案中,存储器存取控制器 75 可耦接到处理元件 100A、100B、存储器子系统 60 的任何组合(或在内部实施),且可耦接到总线 20。计算机系统 10 可在各个实施方案中被不同地配置。

[0033] 处理元件 100A 和 100B 可对应于任何类型的处理器(或定位于其内)(例如,中央处理单元、算术处理单元、图形处理单元、数字信号处理单元,等等)。在一个实施方案中,处理元件 100A 是中央处理单元(一个或多个核的组),且处理元件 100B 是不同类型的处理单元,例如,图形处理单元(可具有一个或多个核)。在一些实施方案中,处理元件 100A 和 100B 中的一者或两者可包括多核。在其它实施方案中,处理元件 100A 和 100B 可以是定位在相同芯片上的一个或多个处理器核的不同组。在一些实施方案中,处理元件 100A 和 100B 可包括各种处理元件的集群或组(例如,元件 100A 可以是两个四核处理器的组)。

[0034] 在一个实施方案中,将处理元件耦接到存储器子系统 60 的总线 20 可以是北桥总线,或所属领域中的技术人员已知的任何其它处理器总线或处理器互连。在一个实施方案中,总线 20 是可定位在相同芯片上的处理器核(一个或多个的组)之间的互连。然而,总线 20 无需限于单个总线或互连,然而并且可以是一个或多个总线、(点对点)互连,或适于将数据传送到本文中所述的结构的其它通信路径和设备的任何组合。

[0035] 存储器子系统60包括一个或多个处理器设备。在各个实施方案中，这些存储器设备可包括RAM模块、嵌入式存储器(例如，eDRAM)、固态存储装置、二级存储装置，诸如硬盘驱动器或任何其它计算机可读介质，该术语如本文中所定义。在一个实施方案中，存储器子系统60包括存储器子系统60的一个或多个存储器设备内的一个或多个存储器区域64。存储器区域64不一定有固定大小或位置，但可取而代之指具有任意开始和结束位置(或地址)的一个或多个存储部分。因此在一个具体实施方案中，第一存储器区域可以是大小为4000KB的一系列存储器位置，而第二存储器区域是大小为32KB的一系列存储器位置。在一个实施方案中，存储器区域64可跨越多个存储器设备(或甚至跨越多种类型的存储器设备；例如，单个存储器区域可包括RAM模块和硬盘驱动器上的存储空间)。存储器区域可以或可以不是物理上或逻辑上相连的。

[0036] 存储器子系统60及其存储器区域可被处理元件100访问。例如，处理元件100A可经由总线20从存储器子系统60中检索数据(并将数据存储在其内)。在各个实施方案中，如本文中和下文所述，存储器子系统60也可被处理元件100B访问。在各个实施方案中，存储器子系统60存储一个或多个程序62。程序62可以是可在计算机系统10上执行的任何程序。因此，在各个实施方案中，程序62可以是JVM、操作系统、API库、在JVM或操作系统上运行的用户程序，等等。在各个实施方案中，程序62可具有从处理元件100A到100B分配存储器初始化的能力，如本文中进一步所述。

[0037] 存储器存取控制器75在一个实施方案中耦接到存储器子系统60，且在各个实施方案中被配置来控制、管理、协调和/或允许由处理元件100到存储器子系统60的存储器访问。存储器存取控制器75在一个实施方案中是直接存储器存取(DMA)控制器，且可与处理元件100A和/或100B定位在相同芯片上。在各个实施方案中，除非被处理元件100A提醒、通知或授予许可，否则存储器存取控制器75可限制处理元件100B访问存储器区域64- 在这种情况中，存取控制器75可允许访问存储器子系统60的一些(或全部)区域。在一个实施方案中，存储器存取控制器75可被配置来使用(和/或耦接到)总线20。

[0038] 高速缓存30可被处理元件100A访问，且包括被配置来保存对应于存储器子系统60的数据的高速缓存。因此高速缓存30可被配置来保存存储在存储器子系统60中的数据子集，以对处理元件100A提供对所述数据的更快访问。在各个实施方案中，高速缓冲30可包括分层高速缓存系统，包括L1、L2、L3或其它高速缓存。在各个实施方案中，高速缓存30可部分或全部定位在处理元件100A内，或者可部分或全部定位在处理元件100A外部(例如，在一个实施方案中，高速缓存30包括处理元件100A内的L1高速缓存，和元件100A外部的L2高速缓存)。与给定处理元件“相关联”的高速缓存被配置来被所述处理元件访问。

[0039] 在一些情况中，缓存操作将促使事先存储在高速缓存30中的数据用其它数据置换(或移位)。在一些实施方案中，当处理元件100A直接访问存储器子系统60的存储器区域时，高速缓存30的一部分将用于存储访问的数据。例如，如果处理元件100A直接访问存储器子系统60以初始化存储器区域64，那么高速缓存30中预先存在的数据可被所述存储器区域新初始化的数据移位。从高速缓存移位的数据可能花费较长时间来访问，这将导致执行时间更长。例如，考虑以下C代码：

[0040] int C=A+B;

[0041] int*Freespace=malloc(8192);

[0042] E=C;

[0043] 这个代码(当编译和执行时)可首先导致变量“C”的数据值被缓存。对 malloc()的调用可接着促使 8192 字节的存储器被初始化,从高速缓存移位“C”的值。在执行下一指令时(其将“C”的值赋予变量 E),高速缓存可能遭遇“丢失”,且因此须从更低级高速缓存或更远的存储器检索变量 C 的值,导致延迟。如果 C 的值一开始就没有从高速缓存被移位,那么可避免这种延迟,可能使性能加速。在各个实施方案中,高速缓存 30 的数据移位 / 置换通过置换策略来管理,所述策略包括对于所属领域的技术人员而言将出现的任何数量的硬件或软件方案,包括最近最少使用(LRU)的置换。

[0044] 现在转向图 2A,示出初始化之前的存储器区域 64 的实例。如所描绘,在各个实施方案中,存储器区域64包括多个存储器位置(包括位置212至216),其中每个可以是个别可寻址的,且被配置来存储给定数据量。如所示出,存储器位置 212 存储数据 205。在一些实施方案中,存储器位置 212 上的数据 205 可被正在由计算机系统 10 执行的程序事先写入,或可以是任意(随机)的。

[0045] 在图 2B 中,示出初始化之后的存储器区域 64 的实例。在这个实施方案中,存储器位置 212 上的数据 205 已通过将其初始化到具有零值的比特序列而被“置零”。如本文中进一步所讨论,在某些实施方案中可由处理元件 100B 执行这种初始化。“置零”只是一种形式的初始化 ;其它初始化可包括以测试图案写入数据(例如,对应于所有负值的值,十六进制值 0×DEADBEEF,等等)。在一些实施方案中,可根据外部规范(诸如 JAVA 编程语言规范)来执行初始化。初始化不限于上文所述的数据类型和值,且在各个实施方案中可包括填充一个或多个存储器区域的任何数据。

[0046] 在一些实施方案中,存储器初始化可限于初始化某一最小大小的存储器区域(可能由服务用于初始化的请求的控制程序自行决定)。例如,存储器初始化可限于初始化不小于一页(诸如由计算机系统 10 的操作系统所定义 -- 例如,8KB 的一页)或高速缓存线的宽度,或给定固定大小(诸如 1024 字节)的存储器区,等等。在这些实施方案中,可通过使用第二处理元件以初始化小存储器区域颁布用于存储器初始化的最小大小的阈值,以避免所涉及的可能的性能损失,原因是在各个实施方案中使用第二处理元件而不是第一处理元件来执行初始化可能涉及某些不可避免的开销成本。

[0047] 现在转向图 3A,示出方框图,其图示了包括存储器子系统 60 内的用户程序 304 和控制程序 310 的实施方案。在一个实施方案中,如上文相对图 1 所述,程序 304 和 310 两者是各自的程序 62。在各个实施方案中,用户程序 304 可能缺乏权限(或可能不被编程和 / 或设计)来直接访问存储器并初始化存储器区域,同时可执行控制程序 310 以初始化存储器区域(例如,使用初始化例程 313)。例如,程序 304 可以是 JAVA 进程和 / 或用户应用 310,而程序 310 可以是 JVM 或操作系统 ;见下文图 3B 至图 3C 的讨论)。在各个实施方案中,用户程序 304 和控制程序 310 存储在子系统 60 中的一个或多个存储器设备内(例如,控制程序 310 可存储在硬盘驱动器上,且在执行期间也(全部或部分)加载到 RAM 模块中)。

[0048] 在各个实施方案中,控制程序 310 包括可被处理元件 100A 和 / 或处理元件 100B 执行的指令 -- 即,给定控制程序 310 可包括可被处理元件 100A、处理元件 100B 或 100A 和 100B 的一些组合执行的指令。例如,在一个实施方案中,控制程序 310 包括在单指令集架构(ISA)中可被 100A 和 100B 两者执行的指令,而在另一实施方案中,控制程序 310 包括在

第一 ISA 中可被处理元件 100A 执行的指令，并且也包括在第二个不同 ISA 中可被处理元件 100B 执行的指令。因此在一些实施方案中，存储器初始化例程 313 可包括在与控制程序 310 的其它部分不同的 ISA 中的指令。

[0049] 在一个实施方案中，控制程序 310 包括程序指令集，其包括初始化例程 313，所述初始化例程 313 可执行以从用户程序 304 中接收存储请求 305。(在另一实施方案中，控制程序 310 在内部产生存储器请求 305。)存储器初始化例程 313 可执行以促使处理元件 100B (而不是元件 100A) 初始化可由初始化请求 305 指定的一个或多个存储器区域 64。在各个实施方案中，存储器初始化例程 313 可包括对应于在编程语言(如 OPENCL、JAVA、C++，等等)中写入的代码的指令。对应于例程 313 的代码可被解译和 / 或编译以在各个实施方案中执行初始化例程 313。

[0050] 可如何使用 OPENCL 代码以生成可由处理元件 100B 执行的指令的实例可在 2010 年 5 月 21 日申请的标题为“DISTRIBUTINGWORKLOADS IN A COMPUTING PLATFORM”的美国专利第 12/785,052 号中找到，所述案件以引用的方式并入本文中。

[0051] 在各个实施方案中，可执行存储器初始化例程 313 以促使处理元件 100B 初始化存储器区域 64。在一个实施方案中，响应于初始化请求 305 (其可由用户程序 304 生成) 而开始执行初始化例程 313。在各个实施方案中，初始化请求 305 可采用各种形式，且包括可用于识别或确定将要初始化的一个或多个存储器区域 64 的信息。在一个实施方案中，请求 305 指定数据对象的名称。在一个实施方案中，初始化请求 305 包括内存基地址和将要初始化的存储器空间的偏移值(长度)。在其它实施方案中，初始化请求 305 包括内存基“起始”地址和将要初始化的存储器上“限”地址。然而，存储器请求 305 不因此受限制，且可包括可用于确定将要初始化的一个或多个存储器区域 64 的任何信息。

[0052] 在执行期间，由处理元件 100A 和 / 或 100B 执行控制程序 310，但在至少一个实施方案中，初始化例程 313 的执行单独由处理元件 100B 借助于初始化请求 307 来执行。在各个实施方案中，可以不同方式进行由元件 100B 执行例程 313。在一个实施方案中，控制程序 310 的部分可由元件 100A 执行以“设置”由元件 100B 执行例程 313。处理元件 100A 可将控制消息、通知或指令发送到处理元件 100B，其包括对例程 313 的引用。在接收到这种控制消息时，处理元件 100B 可接着进入执行例程 313 (例如，通过直接存取存储器，和 / 或其中存储例程 313 的指令的高速缓存)。在另一实施方案中，用于初始化例程 313 的指令可简单地用到总线(诸如总线 20) 上，此时处理元件 100B 将识别并执行所述指令。在一个实施方案中，元件 100A 可执行指令(在元件 100A 的 ISA 中)以对于元件 100B 执行一个或多个配置操作，包括促使存储器存取控制器 75 给处理元件 100B 到存储器区域 64 的直接访问的配置操作。如将对所属领域的技术人员出现的各种其它技术也可用于促使处理元件 100B 执行初始化例程 313。

[0053] 在一个实施方案中，初始化例程 313 的指令包含对于将要初始化的一个或多个存储器区域 64 的一个或多个引用，以及可由处理元件 100B 执行以促使初始化一个或多个存储器区域的指令。填充初始化的存储器区域的数据可以是全零、全负值、图案化数据或任何其它数据，如上文所述。在一些实施方案中，可由控制程序 310 动态生成初始化例程 313 的部分(或全部)。在一个实施方案中，可响应于存储器请求 305 中的信息而出现动态生成。例如，如果存储器请求 305 指定将要初始化 RAM 的 8MB 部分，那么初始化例程 313 的至少一部

分可被动态修改以反映这个 8MB 值。

[0054] 初始化例程 313 可作为各种软件程序的部分而执行 - 例如，在一个实施方案中，例程 313 可作为库例程的部分而执行，其中根据应用编程接口（API）的规范而进行其请求 305。在另一实施方案中，例程 313 可作为 JAVA 垃圾回收进程的部分而执行（如在下文进一步参考图 3C 所描述的）。然而，初始化例程 313 不限于上文所述的程序类型。

[0055] 现在转向图 3B，示出描绘其中计算机系统 10 的操作系统 320 被配置来从第一处理元件到第二处理元件分配存储器初始化的实施方案的方框图。在一个实施方案中，操作系统 320 可完全或部分操作以执行上文相对于控制程序 310 所述的任何和所有操作。在各个实施方案中，操作系统 320 可接收、生成和 / 或处理一个或多个请求 305 以初始化一个或多个存储器区域 64。在一个实施方案中，可由操作系统 320 内的库（或模块）接收请求 305，其可能可被程序（诸如程序 62、程序 304）或甚至操作系统 320 本身调用。在各个实施方案中，这些库可作为一个或多个文件存储在存储器子系统 60 中，且可包括用于模块（诸如对应于 C 编程语言函数 malloc() 和 init() 的 322 和 324）的 API 接口。例如，在计算机系统 10 上运行的程序 62 可通过调用 malloc() 例程而请求具有分配到其的（更多）存储器。因而在一个实施方案中，操作系统 320 可通过加载和 / 或动态生成适当指令（诸如初始化例程 313），且接着促使这些加载或生成的指令由第二处理元件 100B 执行而服务所述请求。出于安全原因，这种初始化可以是可取的，以便避免例如新分配的存储器块从一个程序泄露数据到另一个。在一个实施方案中，Init 模块 324 可用于将另一进程加载到存储器中，且因此可能在内部生成对存储器的请求 305（其继而可促使初始化请求 307 被发送到处理元件 100B）。

[0056] 现在转向图 3C，示出描绘其中 JAVA 虚拟机（JVM）330 被配置来促使将从第一处理元件到第二处理元件分配存储器初始化的实施方案的方框图。JVM300 可完全或部分操作以执行上文相对于控制程序 310 所述的任何和所有操作，且可存储在存储器子系统 60（没有描绘）中。在一个实施方案中，JVM330 被配置来执行存储在存储器子系统 60 中的一个或多个 JAVA 程序的 JAVA 字节码（因而，控制程序 310 可因此执行其它程序，且此外在这方面不限于 JAVA 程序）。JAVA 字节码的执行可促使任何数量的 JAVA 对象 331 被实例化和 / 或销毁。在 JVM330 的各个实施方案中，JAVA 对象的默认初始值可设为全零。在各个实施方案中，这种初始化可通过垃圾回收进程 332 和 / 或构造函数例程 334 而执行（其在一些实施方案中可（且全部或部分）对应于初始化例程 313）。在一个实施方案中，作为垃圾回收进程 332 的最后步骤，通过将存储器区域置零使一个或多个存储器区域的全部可变得对于未来对象分配而可用（因此确保存储已初始化的存储器，直到下一垃圾回收导致附加初始化存储器为止）。或者在各个实施方案中，随着新的对象由 JAVA 用户程序得以分配，可以一次一个的基础完成置零。

[0057] 在一个实施方案中，垃圾回收进程 332 确定哪些 JAVA 对象不再使用且对于这些不使用的对象取消分配存储器。在取消分配此存储器的进程中，JVM330 可初始化一个或多个对应存储器区域以包含零值。JVM330 也可以促使运行一个或多个构造函数例程 334。构造函数例程 334 可以是默认例程，并且可要求对 JVM330 上运行的一个或多个 JAVA 程序进行分配可用存储器，并且可同样在执行这些 JAVA 程序（在各个实施方案中，其可对应于用户程序 304）的期间促使一个或多个存储器区域 64 的初始化。对所属领域的技术人员而言，将出现通过 JVM330 优化存储器区域初始化的各种技术和变换。例如，JVM330 可被配置来将

大量存储器(例如,1MB)“置零”并按需要分出所述存储器以满足新建 JAVA 对象的需求(而不是在每次实例化一个类时初始化存储器)。

[0058] 在各个实施方案中,除操作系统 320 和 JVM330 之外的许多程序可促使计算机系统 10 从处理元件 100A 到处理元件 100B 分配初始化存储器的任务。被设计来由计算机系统 10 的处理元件编译和执行(或解译)的不同编程语言可具有库,其包括被设计来利用存储器初始化分配(或卸载)能力的 API 例程。此外,可设计编译器以促使当从高级源代码中生成可执行代码时使用本文中所述的技术分配存储器初始化。在一个实施方案中,编译器可利用启发法以确定何时执行从第一处理元件到第二个分配一个或多个存储器填充操作的程序将是有利的(例如,可形成这种启发法的基础的因素可包括存储器区域的大小(也许当区域足够大时卸载 / 分配)、初始化之后存储器区域多频繁以及多快被访问、在执行初始化之后的给定时段内被访问的初始化区域的字节数、由于从没有卸载给定存储器初始化导致的高速缓存移位而预期的高速缓存缺失量,等等)。在一些实施方案中,本文中所述的存储器初始化技术在一些情况中对于源代码程序员是透明的 -- 例如,源代码程序员可能在 C 编程语言中根据所述编程语言的规范编程调用 malloc(),而不用知道处理所述调用的库例程将促使从第一元件到第二元件分配存储器初始化。

[0059] 现在转向图 4,示出由第一处理元件到第二处理元件卸载一个或多个存储器区域的初始化的方法 400 的一个实施方案的流程图。方法 400 可全部或部分由计算机系统 10 或任何其它适当计算机系统或计算设备(如下文所述的系统 500)执行。在步骤 410 中,接收将要初始化一个或多个存储器区域的指示。在一个实施方案中,这个步骤可通过执行控制程序 310 以例如从程序 304 接收存储器请求的处理元件 100A 执行。在一个实施方案中,步骤 410 包括接收由垃圾回收进程,如 JVM330 的进程 332 生成的请求。

[0060] 在步骤 420 中,响应于接收步骤 410 的指示,计算机系统 10 促使从处理元件 100A 到处理元件 100B 卸载所请求的存储器区域的初始化。在一个实施方案中,由处理元件 100A 执行步骤 420,并促使所请求的存储器区域的初始化被卸载到处理元件 100B。在各个实施方案中,步骤 420 也可以包括处理元件 100A 执行配置操作或另外以促使处理元件 100B 初始化存储器区域 64 的方式与处理元件 100B 相互作用(例如,设置元件 100B 来执行初始化例程 313)。

[0061] 在步骤 430 中,处理元件(步骤 410 的初始化请求已卸载(即,分配)到其中)初始化所指示的一个或多个存储器区域。在一个实施方案中,这个步骤由处理元件 100B 使用直接存储器存取(经由控制器 75)而执行以初始化所请求的存储器区域。因此在步骤 430 的各个实施方案中,在没有处理元件 100A 的情况下执行初始化直接改变将要初始化的存储器区域的值。在某些实施方案中,根据控制程序 310 的一个或多个预定规则、例程等等执行步骤 430。这些规则可包括启发法(例如,如上文所述的启发法)。

[0062] 在步骤 440 中,计算机系统 10 的高速缓存的一个或多个部分可能是无效的。在一些实施方案中,在具有多个处理元件的系统中(诸如计算机系统 10),存储器区域 64 中的数据拷贝可存储在存储器层次中(包括高速缓存 30)。如果根据方法 400 初始化存储器区域 64,那么在一些情况和一些实施方案中可能有必要执行高速缓存无效程序以便确保对应于初始化存储器区域 64 的陈旧数据拷贝没有保留在计算机系统 10 的高速缓存中(例如,高速缓存 30)。在各个实施方案中,可不同地由处理元件 100A、处理元件 100B 和 / 或存储器存

取控制器 75 开始步骤 440，并且可使用所属领域的技术人员已知的各种技术来执行。

[0063] 现在转向图 5，示出描绘能够实施上文所述的各个实施方案的示例性计算机系统 500 的方框图。计算机系统 500 的组件可与计算机系统 10 的组件完全或部分相同或类似。例如，所描绘的计算机系统 500 包括存储器子系统 60、处理元件 100A 和 100B、高速缓存 30 和存储器存取控制器 75。计算机系统 500 可以是任何各种类型的设备，包括但不限于服务器系统、个人计算机系统、台式电脑、膝上型电脑或笔记本电脑、主机计算机系统、手持式计算机、工作站、网络计算机、消费类设备，诸如移动电话、寻呼机或个人数据助理（PDA）。计算机系统 500 也可以是任何类型的网络外围设备，诸如存储设备、交换机、调制解调器、路由器，等等。虽然为了方便而在图 5 中示出单个计算机系统 500，但是系统 500 也可以作为一起操作的两个或多个计算机系统而实施。

[0064] 在计算机系统 500 的一个实施方案中，存储器子系统 60 包括二级存储装置 455 和 RAM 模块 444 和 446。在一个实施方案中，二级存储装置 455 具有存储在其上的程序指令，其可由第一处理元件 100A 执行以促使计算机系统接收将要初始化存储器区域的指示，其中存储器区域位于计算机系统的存储器中，且响应于所述接收指示，促使由计算设备的第二处理元件 100B 处理存储器区域的初始化。在某些实施方案中，处理元件 100A 和 100B 可以是异构的（即，不同类型）-- 例如其中元件 100A 是中央处理单元（CPU）且 100B 是图形处理单元（GPU）。此外，在一个实施方案中，高速缓存 30 可被配置来响应于处理元件 100A 访问存储器而将一个或多个存储器设备的内容存储在存储器子系统 60 中，其中促使存储器区域的初始化不包括促使高速缓存存储所述存储器区域初始化后的内容（即，高速缓存 30 可避免高速缓存 30 中的其它数据被对应于初始化的存储器区域的新初始化的数据移位）。在各个实施方案中，存储器存取控制器 75 可被配置来对处理元件 100B 提供对存储器子系统 60 中的一个或多个存储器设备的直接访问，其中促使存储器区域的初始化包括处理元件 100B 使用存储器存取控制 75 而访问存储器区域，且其中促使初始化不包括处理元件 100A 访问（即，改变）存储器区域。

[0065] 此外，在一个实施方案中，I/O 设备 444 经由总线 20 耦接到存储器子系统 60。在各个实施方案中，I/O 设备可包括其它存储设备（硬盘驱动器、光盘驱动器、可移动闪存驱动器、存储阵列、SAN 或其相关联的控制器）、网络接口设备（例如，到局域网或广域网的），或其它设备（例如，图形、用户接口设备，等等）。在一个实施方案中，计算机系统 500 经由网络接口设备耦接到网络。根据各个实施方案，I/O 设备可包括各种类型的接口，其可被配置来耦接到其它设备和其接口，并与其它设备和其接口通信。在一个实施方案中，I/O 接口是从前面到一个或多个背面总线的桥芯片（例如，南桥）。

[0066] 在各个实施方案中，存储器子系统 60 包括可由处理元件 100A 和 / 或 100B 使用的存储器。子系统 60 中的存储器可使用不同物理存储器介质（诸如硬盘存储器、软盘存储器、移动硬盘存储器、闪存存储器、随机存取存储器（RAM—SRAM、EDO RAM、SDRAM、DDRSDRAM、RAMBUS RAM，等等）、只读存储器（PROM、EEPROM，等等），等等）实施。计算机系统 500 中的存储器不限于存储，诸如 RAM444 和 446 和二级存储 455 ；相反，计算机系统 500 也可以包括其它形式的存储器，如没有描绘的高速缓存存储器，和 I/O 设备 444 上的二级存储器（例如，硬盘驱动器、存储阵列，等等）。在一些实施方案中，存储器的这些其它形式也可以存储可由处理元件 100A 和 / 或 100B 执行的程序指令。

[0067] 上文所述的技术和方法可作为存储在任何适当计算机可读介质上的计算机可读指令实施。这些指令可以是允许计算机系统和/或计算设备以上文所述的方式操作的软件，且可存储在存储器子系统60内的计算机可读介质中(或不在存储器子系统60内的另一计算机可读介质上)。因此，库例程、垃圾回收进程、其它软件例程和对象，以及任何或所有软件62、304、310、313、320、322、324、330、331、332、334可存储在这种计算机可读介质上。(如上文段落23中所述，这种介质可以是非瞬时性的。)

[0068] 此外，在一些实施方案中，上文所述的技术和方法可在硬件中实施。例如，一个实施方案是包括存储器初始化电路的处理元件，其被配置来促使由第二处理元件处理存储器设备的存储器区域的初始化，其中促使初始化是响应于将要初始化存储器区域的指示而执行。硬件实施方案可使用电路逻辑以实施上文所述的算法和技术(诸如方法400，例如)。

[0069] 硬件实施方案可使用硬件生成指令产生。例如，硬件生成指令可概述一个或多个数据结构，其描述高级设计语言(HDL)(诸如Verilog或VHDL)中的硬件功能的行为级或寄存器传输级(RTL)描述。所述描述可由可合成描述以产生网表的合成工具阅读。网表可包括门集(例如，在合成库中定义)，其表示被配置来实施存储器初始化分配/卸载的处理元件(诸如100A和/或100B)的功能。网表可接着被放置和路由以产生描述将要施加到掩膜的几何形状的数据集。可接着在各种半导体制造步骤中使用掩膜以制造半导体电路或对应于一个或多个处理元件(诸如100A和/或100B)的电路。或者，按照所需，数据库可以是网表(具有或不具有合成库)或数据集。因此，可执行硬件生成指令以促使根据所属制造领域的技术人员已知的技术而产生或制造实施上文所述的方法和技术的处理器和/或处理元件。此外，这种硬件生成指令可存储在任何适当的计算机可读介质上(其可位于存储器子系统内，诸如60，或其它计算机可读介质上)。

[0070] 在一些实施方案中，可使用如上文所述的计算机可读存储介质以存储由程序读取并直接或间接用于制造包括处理元件100A和/或100B的硬件的指令。例如，指令可概述一个或多个数据结构，其描述高级设计语言(HDL)(诸如Verilog或VHDL)中的硬件功能的行为级或寄存器传输级(RTL)描述。所述描述可由可合成描述以产生网表的合成工具阅读。网表可包括门集(例如，在合成库中定义)，其表示处理元件100、存储器初始化单元和/或存储器初始化电路的功能。网表可接着被放置和路由以产生描述将要施加到掩膜的几何形状的数据集。可接着在各种半导体制造步骤中使用掩膜以制造半导体电路或对应于硬件实施方案的电路。或者，按照所需，数据库可以是网表(具有或不具有合成库)或数据集。因此，一个实施方案是(非瞬时性)计算机可读存储介质，其包括数据结构，所述数据结构可由可在计算机系统上执行的程序使用以执行进程的一部分以制造包括由数据结构描述的电路的集成电路，其中所述数据结构中所述的电路包括存储器初始化单元，其被配置来促使由计算设备的第二处理元件(而不是计算设备的第一处理元件)处理存储器设备的存储器区域的初始化，其中所述促使初始化是响应于将要初始化存储器区域的指示而执行。

[0071] * * *

[0072] 虽然已在上文描述了具体实施方案，但是即使在相对于特定特征而描述仅单个实施方案时，这些实施方案也不旨在限制本公开的范畴。除非另外陈述，否则本公开中提供的特征的实例旨在是例证性的，而不是限制性的。如对受益于本公开的所属领域的技术人员将是显然的，上文的描述旨在涵盖这种替代、修改和等同物。

[0073] 本公开的范畴包括本文中公开(明示或暗示)的任何特征或特征组合,或其任何概括,无论其是否减轻本文中解决的任何或所有问题。因而,在起诉本申请案(或主张其优先权的申请案)时将对任何这种特征组合构成新的权利要求。特定而言,关于随附权利要求,来自附属权利要求的特征可与独立权利要求的那些特征组合,且来自各自独立权利要求的特征可以任何适当方式组合,且不仅仅是以随附权利要求中所列举的具体组合。

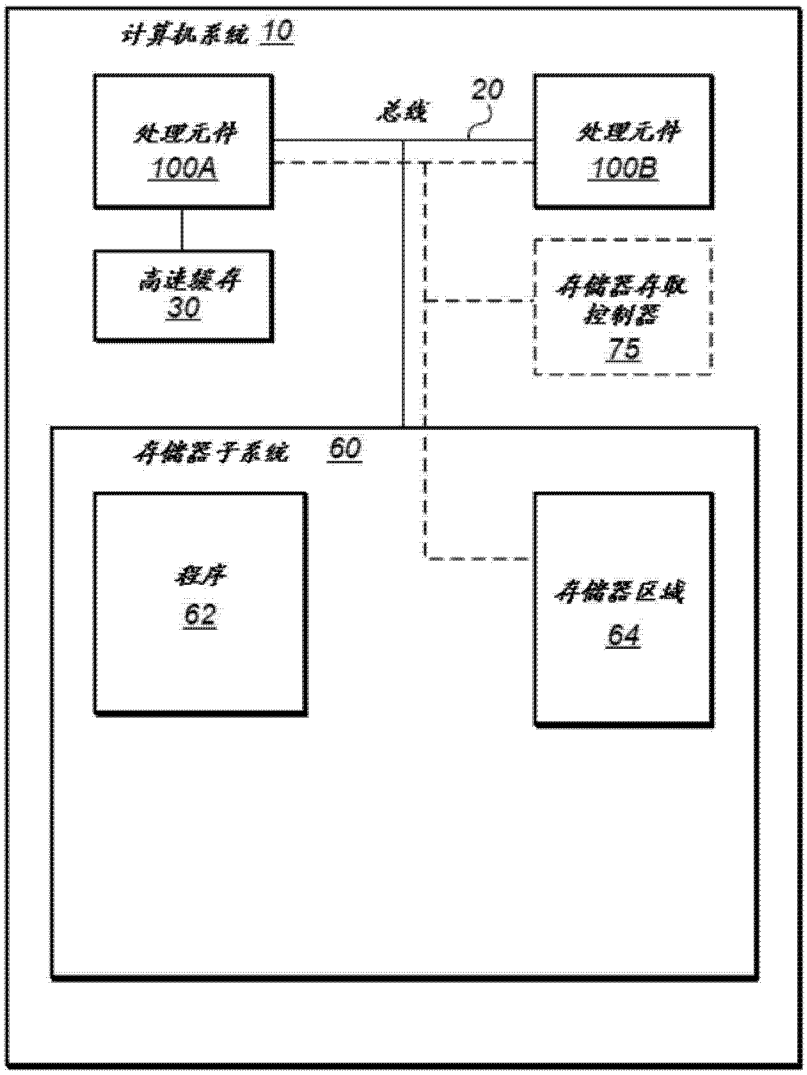

计算机系统 10
处理元件 100A
总线 20
处理元件 100B
高速缓存 30
存储器存取控制器 75
存储器子系统 60
程序 62
存储器区域 64

图 1

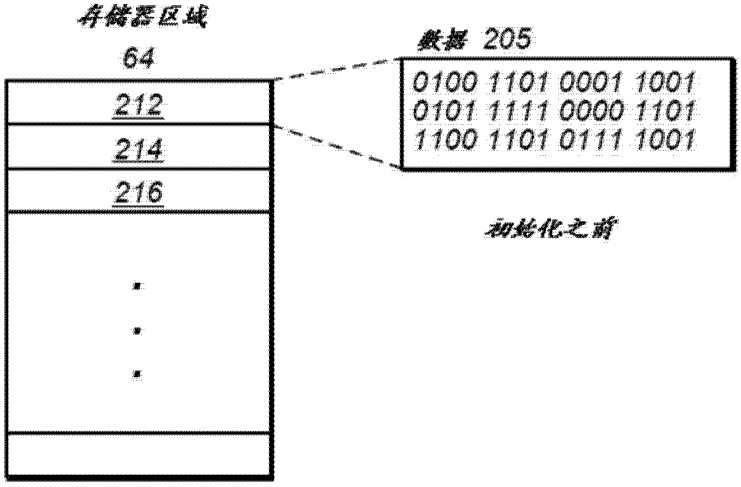

存储器区域
64
212
214
216
数据 205
0100 1101 0001 1001
0101 1111 0000 1101
1100 1101 0111 1001
初始化之前

图 2A

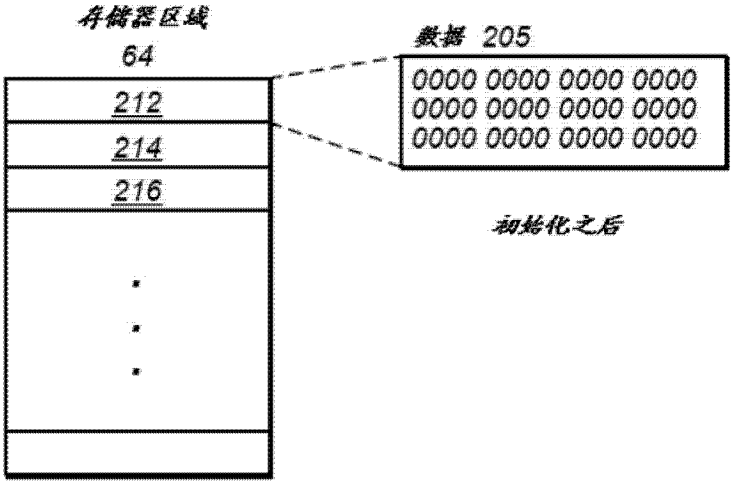

存储器区域
64
212
214
216
数据 205
0000 0000 0000 0000
0000 0000 0000 0000
0000 0000 0000 0000
初始化之后

图 2B

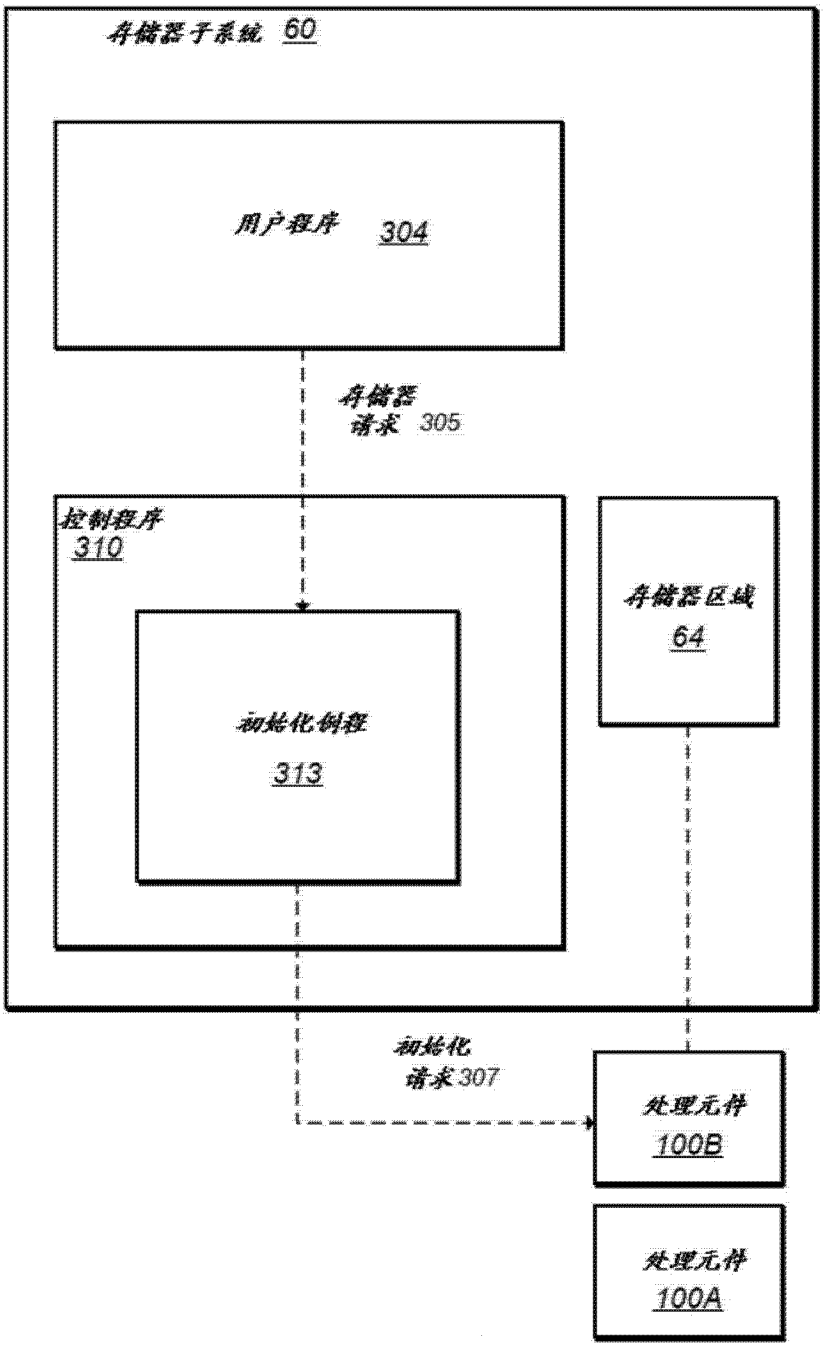

存储器子系统 60
用户程序 304
存储器请求 305
控制程序 310
初始化例程 313
存储器区域 64
初始化请求 307
处理元件 100B
处理元件 100A

图 3A

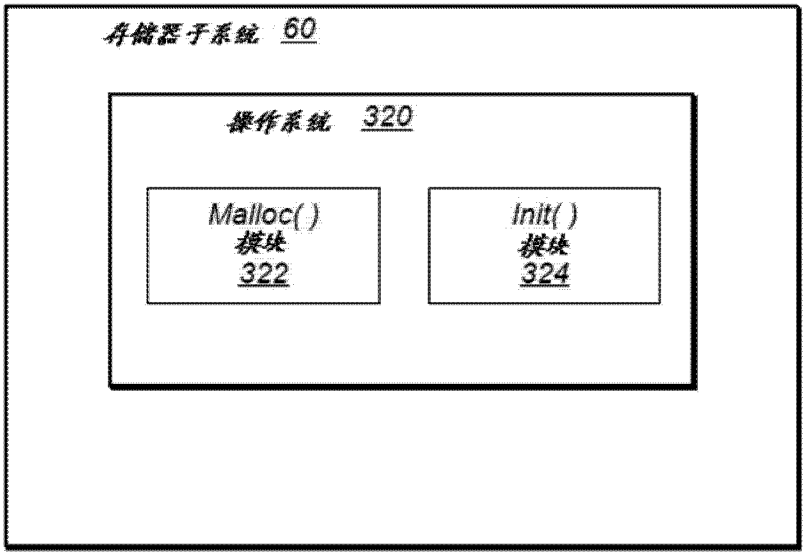

存储器子系统 60
操作系统 320
Malloc( )
模块
322
Init( )
模块
324

图 3B

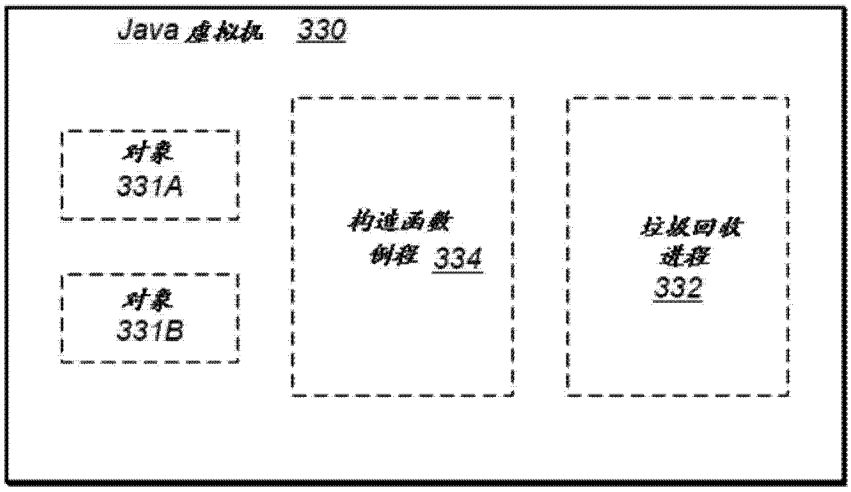

Java 虚拟机 330
对象
331A
对象
331B
构造函数
例程 334
垃圾回收
进程
332

图 3c

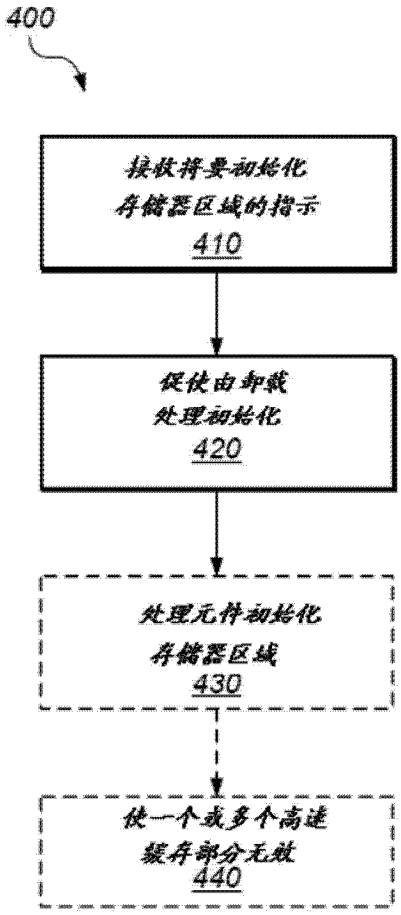

400
接收将要初始化
存储器区域的指示
410
促使由卸载
处理初始化
420
处理元件初始化
存储器区域
430
使一个或多个高速
缓存部分无效
440

图 4

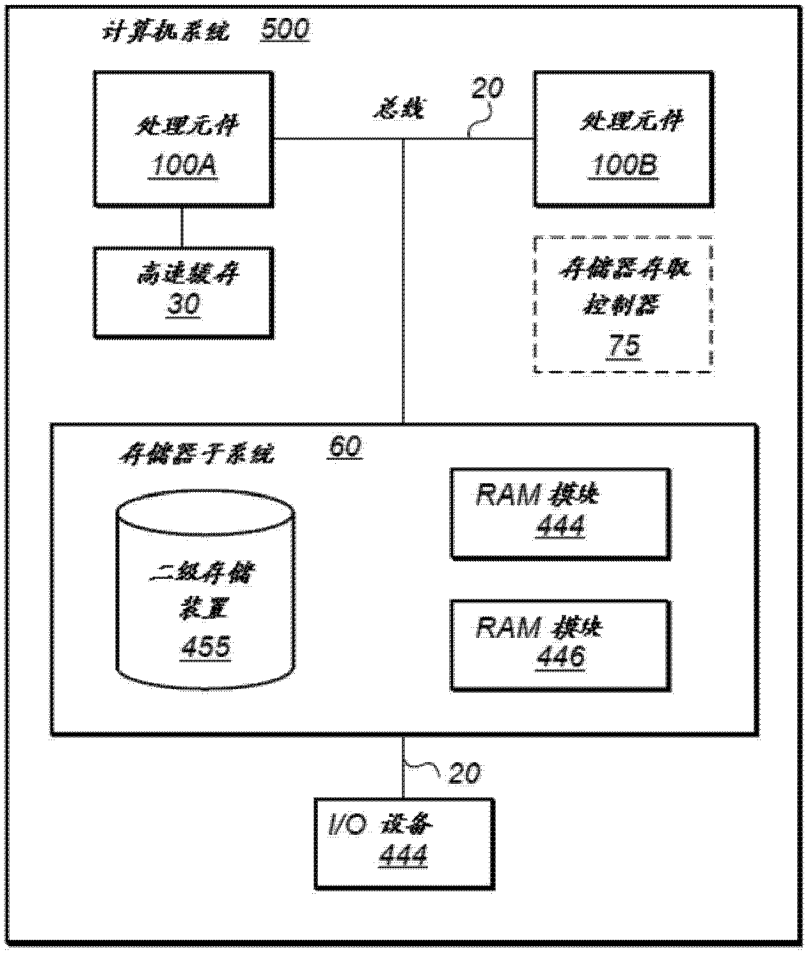

计算机系统 500
处理元件 100A
总线
20
处理元件 100B
高速缓存 30
存储器存取控制器 75
存储器子系统 60
RAM 模块 444
二级存储装置 455
RAM 模块 446
20
I/O 设备 444

图 5