(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 117785484 B
(45)授权公告日 2024.05.17

(21)申请号 202410205362.8
(22)申请日 2024.02.26
(65)同一申请的已公布的文献号
申请公布号 CN 117785484 A
(43)申请公布日 2024.03.29
(73)专利权人 北京卡普拉科技有限公司
地址 102600 北京市大兴区北兴路(东段)2号院5号楼4层A404
(72)发明人 孙超 李锐喆 赵彤
(74)专利代理机构 北京清大紫荆知识产权代理有限公司 11718
专利代理师 张梦龙
(51)Int.Cl.
G06F 9/50(2006.01)
G06F 12/0811(2016.01)

(56)对比文件
CN 102609362 A,2012.07.25
CN 102640127 A,2012.08.15
CN 104239228 A,2014.12.24
CN 109471734 A,2019.03.15
US 2011246995 A1,2011.10.06

审查员 王璐

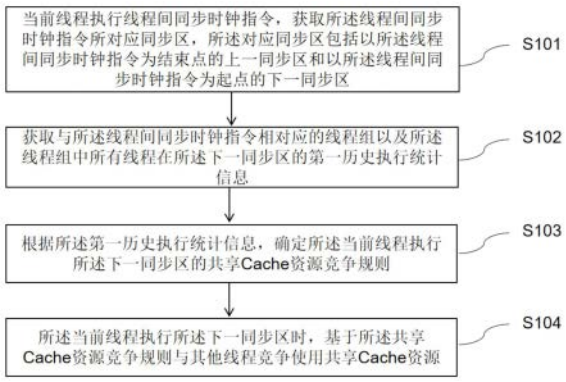
权利要求书2页 说明书10页 附图2页

(54)发明名称

共享Cache资源分配方法、系统、计算机设备及介质

(57)摘要

本发明实施例提供了一种共享Cache资源分配方法、系统、计算机设备及介质,涉及计算机技术领域,包括当前线程执行线程间同步时钟指令,获取线程间同步时钟指令所对应同步区,对应同步区包括以线程间同步时钟指令为结束点的上一同步区和以线程间同步时钟指令为起点的下一同步区;获取与线程间同步时钟指令相对应的线程组以及线程组中所有线程在下一同步区的第一历史执行统计信息;根据第一历史执行统计信息,确定当前线程执行下一同步区的共享Cache资源竞争规则;当前线程执行下一同步区时,基于共享Cache资源竞争规则与其他线程竞争使用共享Cache资源。该方案加速负载大线程的运行,进而提高了整个并行程序的运行速度。



CN 117785484 B

1.一种共享Cache资源分配方法,其特征在于,包括:

当前线程执行线程间同步时钟指令,获取所述线程间同步时钟指令所对应同步区,所述对应同步区包括以所述线程间同步时钟指令为结束点的上一同步区和以所述线程间同步时钟指令为起点的下一同步区;

获取与所述线程间同步时钟指令相对应的线程组以及所述线程组中所有线程在所述下一同步区的第一历史执行统计信息;

根据所述第一历史执行统计信息,确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则;

所述当前线程执行所述下一同步区时,基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源;

其中,所述确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则,包括:根据所述线程间同步时钟指令,获取所述线程组中每个线程的当前时间,并根据所述第一历史执行统计信息预测所述下一同步区的预测执行时长;针对每个线程,基于所述当前时间和所述预测执行时长,获得每个线程执行所述下一同步区的预测结束时间;对所述线程组中所述预测结束时间越晚的线程设置越大的竞争优势,基于所述竞争优势的设置规则形成所述共享Cache资源竞争规则。

2.如权利要求1所述的共享Cache资源分配方法,其特征在于,所述获取所述线程间同步时钟指令所对应同步区的信息,包括:

一个所述线程组中的各个线程在同一个同步区具有相同的起点指令的指令标记和相同的结束点指令的指令标记,一个所述线程组中的各个线程在不同的同步区具有不同的起点指令的指令标记和不同的结束点指令的指令标记;

根据所述起点指令的指令标记和所述结束点指令的指令标记,获取所述线程间同步时钟指令所对应同步区的信息。

3.如权利要求1所述的共享Cache资源分配方法,其特征在于,所述根据所述第一历史执行统计信息,确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则,包括:

根据所述第一历史执行统计信息,判断所述当前线程所对应的应用程序在所述下一同步区的执行特征是否具有可预测性;

若所述执行特征具有可预测性,则根据所述第一历史执行统计信息,判断所述应用程序的执行速度对共享Cache竞争是否敏感;

若所述应用程序的执行速度对共享Cache竞争敏感,则确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则。

4.如权利要求1所述的共享Cache资源分配方法,其特征在于,所述当前线程执行所述下一同步区时,基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源,包括:

所述当前线程在开始执行所述下一同步区时,将所述共享Cache资源竞争规则配置到共享Cache硬件设置上;在所述当前线程执行所述下一同步区的过程中,所述共享Cache硬件根据所述共享Cache资源竞争规则在所述线程组中分配Cache资源;在所述当前线程执行完所述下一同步区时,将所述共享Cache硬件设置恢复为无资源竞争的情况;

所述根据所述共享Cache资源竞争规则在所述线程组中分配Cache资源包括:让竞争优

势越大的线程获得越大的共享Cache竞争容量,或者让竞争优势越大的线程的数据在共享Cache中被替换时的比率越低。

5.如权利要求3所述的共享Cache资源分配方法,其特征在于,所述判断所述应用程序的执行速度对共享Cache竞争是否敏感的方式包括:

若所述应用程序由共享Cache访问缺失所引发的等待时间在所述下一同步区执行时长中所占比例超过第一预设阈值,则判断所述应用程序的执行速度对共享Cache竞争敏感;

或者,若使用或改变所述共享Cache资源竞争规则后,共享Cache访问缺失次数以及所述下一同步区的执行时长的变化值超过第二预设阈值,则判断所述应用程序的执行速度对共享Cache竞争敏感。

6.如权利要求3所述的共享Cache资源分配方法,其特征在于,所述方法还包括:

判断所述应用程序的执行速度对内存总线的竞争是否敏感;

若所述应用程序的执行速度对所述内存总线的竞争敏感,则确定所述当前线程在所述下一同步区的内存总线竞争规则,并让所述当前线程执行所述下一同步区时,基于所述内存总线竞争规则与其他线程竞争使用内存总线。

7.一种共享Cache资源分配装置,其特征在于,包括:

第一获取模块,用于当前线程执行线程间同步时钟指令,获取所述线程间同步时钟指令所对应同步区的信息,所述对应同步区包括以所述线程间同步时钟指令为结束点的上一同步区和以所述线程间同步时钟指令为起点的下一同步区;

第二获取模块,用于获取与所述线程间同步时钟指令相对应的线程组以及所述线程组中所有线程在所述下一同步区的第一历史执行统计信息;

竞争规则确定模块,用于根据所述第一历史执行统计信息,确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则;

执行模块,用于所述当前线程执行所述下一同步区时,基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源;

其中,所述确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则,包括:根据所述线程间同步时钟指令,获取所述线程组中每个线程的当前时间,并根据所述第一历史执行统计信息预测所述下一同步区的预测执行时长;针对每个线程,基于所述当前时间和所述预测执行时长,获得每个线程执行所述下一同步区的预测结束时间;对所述线程组中所述预测结束时间越晚的线程设置越大的竞争优势,基于所述竞争优势的设置规则形成所述共享Cache资源竞争规则。

8.一种计算机设备,包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序,其特征在于,所述处理器执行所述计算机程序时实现权利要求1至6中任一项所述的共享Cache资源分配方法。

9.一种计算机可读存储介质,其特征在于,所述计算机可读存储介质存储有执行权利要求1至6中任一项所述的共享Cache资源分配方法的计算机程序。

共享Cache资源分配方法、系统、计算机设备及介质

技术领域

[0001] 本发明涉及计算机技术领域,特别涉及一种共享Cache资源分配方法、系统、计算机设备及介质。

背景技术

[0002] Cache即高速缓存,是从过去到现在通用处理器(CPU)上的重要部件,其通过缓存部分热点数据来减少处理器对内存访问,从而降低访存延迟,提高整个程序的运行速度。此外,Cache支持访存地址的虚实映射,处理器的寄存器与Cache之间、Cache与内存之间的数据存取均由硬件系统自行完成,使得应用程序无需控制对Cache的使用,这能保持程序员对应用程序代码的便捷编写。为了提高访存效率,现代通用处理器通常采用了多级Cache,例如Intel公司的很多CPU都有三个层级的Cache,标记为L1~L3,其中L1离处理器计算单元最近、访问速度最快、容量最小,而L3离处理器计算单元最远、容量最大、访问速度最慢。

[0003] 相对于应用程序所处理的数据量而言,Cache的容量总是有限,Cache中会不断发生数据替换的现象,即有的数据从内存进入Cache,而有的数据则从Cache中被挤出去。Cache替换是以固定Cache块大小为单位的,一个Cache块通常为64字节。

[0004] 随着众核技术的快速发展,一块芯片上通常有十多个、数十个甚至上百个处理器核心,其中每个处理器核心具有独立的私有Cache,而多个处理器核心通常会共享最后一级Cache(如L3)。多个处理器核不会竞争使用各自的私有Cache,但会竞争使用共享Cache。尽管很多并行应用程序的多个线程(一个进程包含一个或多个线程)具有同样的指令流、甚至处理的数据量都相近,但对共享Cache的竞争使用可能导致计算量和数据量相近的多个线程的运行速度不同,甚至有较大差异,这就使得本身计算负载平衡的并行程序表现出负载明显不平衡且难以预测的现象,从而造成并行程序性能的明显下降。

[0005] 很多应用程序的计算负载是动态变化的,这样程序的并行化设计很难做到不同线程间的计算负载平衡。例如,对于预报天气或模拟气候的应用程序而言,在有云与无云、白天与晚上、下雨与不降水等情况下的计算量会有很大的变化;还有些科学计算程序采用了隐式迭代求解的方法,迭代收敛的步数也往往是在积分过程中动态变化的。对于线程间计算负载明显不均衡的情况,常会出现同步时负载小线程等待负载大线程的情况。试想,当负载大线程与负载小线程竞争使用共享Cache时,如果能让负载大线程具有竞争优势,则有可能加速负载大线程的运行,从而加速整个并行程序的运行。

[0006] 综上所述,至少有两个问题值得关注:1)对于负载平衡的多个线程,如何保证它们在竞争使用共享Cache资源的公平性;2)对于负载不平衡的多个线程,如何让负载大的线程在竞争使用共享Cache资源时具有竞争优势。为解决上述问题,本申请提出了基于线程间同步时钟的共享Cache资源分配方法。

发明内容

[0007] 有鉴于此,本发明实施例提供了一种共享Cache资源分配方法,以解决现有技术中

时钟同步时线程间负载不平衡的技术问题。该方法包括：

[0008] 当前线程执行线程间同步时钟指令，获取所述线程间同步时钟指令所对应同步区的信息，所述对应同步区包括以所述线程间同步时钟指令为结束点的上一同步区和以所述线程间同步时钟指令为起点的下一同步区；

[0009] 获取与所述线程间同步时钟指令相对应的线程组以及所述线程组中所有线程在所述下一同步区的第一历史执行统计信息；

[0010] 根据所述第一历史执行统计信息，确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则；

[0011] 所述当前线程执行所述下一同步区时，基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源。

[0012] 本发明实施例还提供了一种共享Cache资源分配装置，以解决现有技术中时钟同步时线程间负载不平衡的技术问题。该装置包括：

[0013] 第一获取模块，用于当前线程执行线程间同步时钟指令，获取所述线程间同步时钟指令所对应同步区的信息，所述对应同步区包括以所述线程间同步时钟指令为结束点的上一同步区和以所述线程间同步时钟指令为起点的下一同步区；

[0014] 第二获取模块，用于获取与所述线程间同步时钟指令相对应的线程组以及所述线程组中所有线程在所述下一同步区的第一历史执行统计信息；

[0015] 竞争规则确定模块，用于根据所述第一历史执行统计信息，确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则；

[0016] 执行模块，用于所述当前线程执行所述下一同步区时，基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源。

[0017] 本发明实施例还提供了一种计算机设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序，所述处理器执行所述计算机程序时实现上述任意的共享Cache资源分配方法，以解决现有技术中时钟同步时线程间负载不平衡的技术问题。

[0018] 本发明实施例还提供了一种计算机可读存储介质，所述计算机可读存储介质存储有执行上述任意的共享Cache资源分配方法的计算机程序，以解决现有技术中时钟同步时线程间负载不平衡的技术问题。

[0019] 与现有技术相比，本说明书实施例采用的上述至少一个技术方案能够达到的有益效果至少包括：当前线程执行线程间同步时钟指令，获取线程间同步时钟指令所对应同步区，同步区包括以线程间同步时钟指令为结束点的上一同步区和以线程间同步时钟指令为起点的下一同步区；获取与线程间同步时钟指令相对应的线程组以及线程组中所有线程在下一同步区的第一历史执行统计信息；根据第一历史执行统计信息，确定当前线程执行下一同步区的共享Cache资源竞争规则；当前线程执行下一同步区时，基于共享Cache资源竞争规则与其他线程竞争使用共享Cache资源。本申请通过对多个线程执行同步区时设置共享Cache资源竞争规则，实现了对于负载不平衡的多个线程，让负载不同的线程在竞争使用共享Cache资源时具有不同的竞争优势，从而使共享Cache资源得到合理利用，进而加速负载大线程的运行，从而加速整个并行程序的运行。

附图说明

[0020] 为了更清楚地说明本申请实施例的技术方案,下面将对实施例中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本申请的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其它的附图。

[0021] 图1是本发明实施例提供的共享Cache资源分配方法的流程图;

[0022] 图2是本发明实施例提供的一个执行程序的示例图;

[0023] 图3是本发明实施例提供的一种计算机设备的结构框图;

[0024] 图4是本发明实施例提供的一种共享Cache资源分配装置的结构框图。

具体实施方式

[0025] 下面结合附图对本申请实施例进行详细描述。

[0026] 以下通过特定的具体实例说明本申请的实施方式,本领域技术人员可由本说明书所揭露的内容轻易地了解本申请的其他优点与功效。显然,所描述的实施例仅仅是本申请一部分实施例,而不是全部的实施例。本申请还可以通过另外不同的具体实施方式加以实施或应用,本说明书中的各项细节也可以基于不同观点与应用,在没有背离本申请的精神下进行各种修饰或改变。需说明的是,在不冲突的情况下,以下实施例及实施例中的特征可以相互组合。基于本申请中的实施例,本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例,都属于本申请保护的范围。

[0027] 在本发明实施例中,提供了一种共享Cache资源分配方法,如图1所示,该方法包括以下步骤:

[0028] 步骤S101、当前线程执行线程间同步时钟指令,获取所述线程间同步时钟指令所对应同步区的信息,所述对应同步区包括以所述线程间同步时钟指令为结束点的上一同步区和以所述线程间同步时钟指令为起点的下一同步区;

[0029] 步骤S102、获取与所述线程间同步时钟指令相对应的线程组以及所述线程组中所有线程在所述下一同步区的第一历史执行统计信息;

[0030] 步骤S103、根据所述第一历史执行统计信息,确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则;

[0031] 步骤S104、所述当前线程执行所述下一同步区时,基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源。

[0032] 本实施例中通过对多个线程执行同步区时设置共享Cache资源竞争规则,实现了对于负载不平衡的多个线程,让负载不同的线程在竞争使用共享Cache资源时具有不同的竞争优势,从而使共享Cache资源得到合理利用,进而加速负载大线程的运行,从而加速整个并行程序的运行。

[0033] 在一个实施例中,共享Cache资源分配方法包括以下步骤:

[0034] 步骤S201、当前线程执行线程间同步时钟指令,获取所述线程间同步时钟指令所对应同步区的信息,所述对应同步区包括以所述线程间同步时钟指令为结束点的上一同步区和以所述线程间同步时钟指令为起点的下一同步区;

[0035] 步骤S202、更新所述当前线程在所述上一同步区的第二历史执行统计信息;

[0036] 步骤S203、获取与所述线程间同步时钟指令相对应的线程组以及所述线程组中所

有线程在所述下一同步区的第一历史执行统计信息；

[0037] 步骤S204、根据所述第一历史执行统计信息，确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则；

[0038] 步骤S205、所述当前线程执行所述下一同步区时，基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源。

[0039] 通过增加更新所述当前线程在所述上一同步区的第二历史执行统计信息的步骤，使得历史统计信息可用作后续预测的基础。

[0040] 具体实施时，关于“当前线程执行线程间同步时钟指令”，一条线程间同步时钟指令不会引起线程间的同步，但能表明一个线程组在执行该指令后将进行同步，或线程组中各线程执行到该指令时的时间应尽量相近。线程组中的各个线程需要使用相同的同步标记去调用线程间同步时钟指令。

[0041] 因此，线程间同步时钟指令需要以线程组信息和指令标记为输入。换而言之，可以从一条线程间同步时钟指令中获取到相应线程组及其中各线程的相关信息，并能获取到相应指令标记。一个应用程序通常会迭代地执行其指令流，在迭代执行过程中，对应着同一线程组标记和同一指令标记的线程间同步时钟指令会执行多次。

[0042] 具体的，对于同步区进行解释说明。一个同步区可理解为执行时长或执行结束时间在多个线程间需要尽量接近的一段指令流，其通过线程间同步时钟指令得以识别，即一个同步区是在两条线程间同步时钟指令之间执行的指令流，其中一条指令为同步区的起点，而另一条指令则为同步区的结束点。因此，一条线程间同步时钟指令通常是上一同步区的结束点，也是下一同步区的起点。

[0043] 在一个实施例中，所述获取所述线程间同步时钟指令所对应同步区的信息，包括：

[0044] 一个所述线程组中的各个线程在同一个同步区具有相同的起点指令的指令标记和相同的结束点指令的指令标记，一个所述线程组中的各个线程在不同的同步区具有不同的起点指令的指令标记和不同的结束点指令的指令标记；

[0045] 根据所述起点指令的指令标记和所述结束点指令的指令标记，获取所述线程间同步时钟指令所对应同步区的信息。

[0046] 具体的，对于一组线程的一个同步区，各线程在所述下一同步区的起点指令需具有相同的指令标记，各线程在所述下一同步区的结束点指令也需具有相同的指令标记。对于一个同步区，起点指令的指令标记和结束点指令的指令标记在大多数情况下互不相同(只有当循环程序中只有一条线程间同步时钟指令时，会出现起点指令的指令标记和结束点指令的指令标记相同的情况)；而对于不同的两个同步区，它们两者之间的起点指令的指令标记和结束点指令的指令标记都应不相同。在应用程序的迭代执行过程中，同一个同步区也会执行多遍。因此，可以利用一组线程在执行一个同步区的历史信息，去确定下一次执行时的共享Cache竞争设置。

[0047] 图2中示出了一个执行程序示例，该循环内共有四条线程间同步时钟指令(do_sync_clock)，共形成四个同步区，分别是“sync1”-“sync2”、“sync2”-“sync3”、“sync3”-“sync4”、“sync4”-“sync1”。其中，在同步区“sync1”-“sync2”中，对do_comp1进行计算；在同步区“sync2”-“sync3”中，对do_comp2进行计算；在同步区“sync3”-“sync4”中，进行了一次线程间的同步(thread_synchronize，线程同步)；在同步区“sync4”-“sync1”中，没有进

行计算,也没有进行线程间同步。由于循环要执行100次(int i = 0; i<100; i++),每条线程间同步时钟指令都会执行100次。

[0048] 在一个实施例中,所述更新所述当前线程在所述上一同步区的第二历史执行统计信息,具体包括:首先需要获取到当前线程在所述上一同步区从起点指令到结束点指令之间的此次执行统计信息,然后将此次执行统计信息更新到第二历史执行统计信息中。所述第二历史执行统计信息可包含最近若干次执行统计信息。执行统计信息除包含同步区的运行时间信息(包括开始时间、结束时间和运行时长)外,还可以包含同步区的一些处理器事件,例如,执行的指令数量、访问各级私有Cache的次数、访问各级私有Cache的缺失次数、访问共享Cache的次数、访问共享Cache的缺失次数、因共享Cache访问缺失而引发的等待时间等;另外,执行统计信息还可以包括历史的共享Cache资源竞争规则。

[0049] 在一个实施例中,所述根据所述第一历史执行统计信息,确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则,包括:

[0050] 根据所述第一历史执行统计信息,判断所述当前线程所对应的应用程序在所述下一同步区的执行特征是否具有可预测性;

[0051] 若所述执行特征具有可预测性,则根据所述第一历史执行统计信息,判断所述应用程序的执行速度对共享Cache竞争是否敏感;

[0052] 若所述应用程序的执行速度对共享Cache竞争敏感,则确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则。

[0053] 具体实施时,所述下一同步区的第一历史执行统计信息包括相应线程组中各线程在最近若干次执行所述下一同步区时的执行统计信息。根据第一历史执行统计信息,可以首先确定应用程序在所述下一同步区的执行特征是否稳定或有规律,即确定执行特征是否具有可预测性;并确定执行速度对共享Cache竞争是否敏感。例如,当具有稳定执行特征时,各个线程在最近两次甚至更多次执行所述下一同步区时会保持相近的指令数量、访问各级私有Cache的次数和访问各级私有Cache的缺失次数;则可利用所述稳定执行特征,有效预测同步区在一次运行时的执行统计信息。对于有规律的执行特征中的规律,可以是递增、递减或周期性波动等形式的规律。

[0054] 在一个实施例中,所述判断所述应用程序的执行速度对共享Cache竞争是否敏感的方式包括:

[0055] 若所述应用程序由共享Cache访问缺失所引发的等待时间在所述下一同步区执行时长中所占比例超过第一预设阈值,则判断所述应用程序的执行速度对共享Cache竞争敏感;

[0056] 或者,若使用或改变所述共享Cache资源竞争规则后,共享Cache访问缺失次数以及所述下一同步区的执行时长的变化值超过第二预设阈值,则判断所述应用程序的执行速度对共享Cache竞争敏感。

[0057] 具体实施时,对于如何确定执行速度对共享Cache竞争是否敏感,可从两方面入手:1)判断由共享Cache访问缺失所引发的等待时间在同步区执行时长中所占有的比例是否足够大,这里可采用一个第一预设阈值进行判断;2)判断执行速度对共享Cache资源竞争规则的调整是否敏感,即查看使用或改变共享Cache资源竞争规则后,共享Cache访问缺失次数及同步区的执行时长是否发生了足够的变化。因此,通过这两种方式即可查看执行速

度对共享Cache竞争是否敏感,进而可对执行速度对共享Cache竞争敏感的线程分配更具有竞争优势的共享Cache资源,从而加快负载大线程的运行。

[0058] 可设计与上述类似的方法,以判断所述应用程序的执行速度对内存总线竞争是否敏感。

[0059] 在一个实施例中,所述确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则,包括:

[0060] 根据所述线程间同步时钟指令,获取所述线程组中每个线程的当前时间,并根据所述第一历史执行统计信息预测所述下一同步区的预测执行时长;

[0061] 针对每个线程,基于所述当前时间和所述预测执行时长,获得每个线程执行所述下一同步区的预测结束时间;

[0062] 对所述线程组中所述预测结束时间越晚的线程设置越大的竞争优势,基于所述竞争优势的设置规则形成所述共享Cache资源竞争规则。

[0063] 具体实施时,在确定所述下一同步区的执行特征具有可预测性且执行速度对共享Cache竞争足够敏感时,将确定共享Cache资源竞争规则。对于一个线程,可以根据当前时间(即执行当前线程间同步时钟指令的时间)和所预测的当前线程运行所述下一同步区的执行时长,得到该线程在执行完所述下一同步区时的预测结束时间,通过该方法获得线程组中所有线程的预测结束时间。对于线程组中的所有线程,预测结束时间越晚的线程的共享Cache竞争优势应越大,形成所述共享Cache资源竞争规则。

[0064] 本实施例中,通过对负载不平衡的多个线程,通过设置共享Cache资源竞争规则,可以让负载大的线程在竞争使用共享Cache资源时具有竞争优势,从而加速负载大线程的运行,从而加速整个并行程序的运行,提高了程序的运行速度。

[0065] 在一个实施例中,所述当前线程执行所述下一同步区时,基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源,包括:

[0066] 所述当前线程在开始执行所述下一同步区时,将所述共享Cache资源竞争规则配置到共享Cache硬件设置上;在所述当前线程执行所述下一同步区的过程中,所述共享Cache硬件根据所述共享Cache资源竞争规则在所述线程组中分配Cache资源;在所述当前线程执行完所述下一同步区时,将所述共享Cache硬件设置恢复为无资源竞争的情况;

[0067] 所述根据所述共享Cache资源竞争规则在所述线程组中分配Cache资源包括:让竞争优势越大的线程获得越大的共享Cache竞争容量,或者让竞争优势越大的线程的数据在共享Cache中被替换时的比率越低。

[0068] 本实施例中,对于多个负载不平衡的线程,当负载大线程与负载小线程竞争使用共享Cache时,基于所述共享Cache资源竞争规则,使负载大的线程在竞争使用共享Cache资源时具有更高的竞争优势。本实施例中,通过对线程的预测结束时间的早晚分辨该线程的负载大小,线程的预测结束时间越晚表明线程的负载越大,对其分配的共享Cache资源使用的竞争优势越大,也即基于所述共享Cache资源竞争规则,对所述当前线程分配与所述当前线程的预测结束时间相适应的竞争优势。

[0069] 共享Cache资源竞争优势大的直接效果可以体现在两个方面:1)在共享Cache中,竞争优势越大的线程所占用的Cache容量越大或Cache块被替换的概率越低;2)在竞争使用内存总线时,竞争优势越大的线程使用内存总线的优先级越高。

[0070] 在一个实施例中,所述方法还包括:

[0071] 判断所述应用程序的执行速度对内存总线的竞争是否敏感;

[0072] 若所述应用程序的执行速度对所述内存总线的竞争敏感,则确定所述当前线程在所述下一同步区的内存总线竞争规则,并让所述当前线程执行所述下一同步区时,基于所述内存总线竞争规则与其他线程竞争使用内存总线。

[0073] 具体实施时,例如,存在采用流式数据访问的应用程序,其运行速度可能对Cache容量不敏感,但对内存总线的竞争敏感;对于这类应用,Cache竞争的优势应体现在对内存总线的竞争使用上;这类应用程序的一个重要特点是访问最后一级私有Cache(离共享Cache最近的一级私有Cache)的缺失次数与访问共享Cache的缺失次数比较相近,因此可以通过该特点识别出运行速度对Cache容量不敏感,但对内存总线竞争敏感的应用程序。

[0074] 在一个实施例中,所述方法还包括:

[0075] 若出现线程组内所有线程对共享Cache竞争优势应相同的情况,此时处理器硬件系统可通过预设机制确保让所有线程公平地使用共享Cache资源以及内存总线。

[0076] 通过本实施例的方法,可有效保证负载平衡的多个线程在竞争使用共享Cache资源的公平性。

[0077] 在一个实施例中,所述方法还包括:

[0078] 将对线程间同步时钟指令的调用添加到应用程序中,添加的方式包括以下两种方式:1)通过编译器自动添加,例如编译器可检测到诸如OpenMP等线程级并行方式下的线程间同步操作,然后在同步操作的前和后自动插入对线程间同步时钟指令的调用;2)提供添加线程间同步时钟指令调用的程序接口,使得应用程序研发人员可以根据应用程序的特点自行添加调用。

[0079] 需要说明的是,由于不能把一个线程从一个计算节点迁移到另一计算节点上执行,但在同一计算节点内可以把一个线程从一个处理器核迁移到另一个处理器核,因此本申请的共享Cache资源分配方法在一个计算节点内使用即可。

[0080] 本申请的上述实施例,通过对多个线程执行同步区时设置共享Cache资源竞争规则,实现了对于负载不平衡的多个线程,让负载不同的线程在竞争使用共享Cache资源时具有不同的竞争优势,从而使共享Cache资源得到合理利用,进而加速负载大线程的运行,从而加速整个并行程序的运行。

[0081] 在本实施例中,提供了一种计算机设备,如图3所示,包括存储器301、处理器302及存储在存储器上并可在处理器上运行的计算机程序,所述处理器执行所述计算机程序时实现上述任意的共享Cache资源分配方法。

[0082] 具体的,该计算机设备可以是计算机终端、服务器或者类似的运算装置。

[0083] 在本实施例中,提供了一种计算机可读存储介质,所述计算机可读存储介质存储有执行上述任意的共享Cache资源分配方法的计算机程序。

[0084] 具体的,计算机可读存储介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机可读存储介质的例子包括,但不限于相变内存(PRAM)、静态随机存取存储器(SRAM)、动态随机存取存储器(DRAM)、其他类型的随机存取存储器(RAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、快闪记忆体或其他内存技术、只读光盘只读

存储器(CD-ROM)、数字多功能光盘(DVD)或其他光学存储、磁盒式磁带,磁带磁盘存储或其他磁性存储设备或任何其他非传输介质,可用于存储可以被计算设备访问的信息。按照本文中的界定,计算机可读存储介质不包括暂存电脑可读媒体(transitory media),如调制的数据信号和载波。

[0085] 基于同一发明构思,本发明实施例中还提供了一种共享Cache资源分配装置,如下面的实施例所述。由于共享Cache资源分配装置解决问题的原理与共享Cache资源分配方法相似,因此共享Cache资源分配装置的实施可以参见共享Cache资源分配方法的实施,重复之处不再赘述。以下所使用的,术语“单元”或者“模块”可以实现预定功能的软件和/或硬件的组合。尽管以下实施例所描述的装置较佳地以软件来实现,但是硬件,或者软件和硬件的组合的实现也是可能并被构想的。

[0086] 图4是本发明实施例的共享Cache资源分配装置的一种结构框图,如图4所示,包括:第一获取模块401、第二获取模块402、竞争规则确定模块403和执行模块404,下面对该结构进行说明。

[0087] 第一获取模块401,用于当前线程执行线程间同步时钟指令,获取所述线程间同步时钟指令所对应同步区的信息,所述对应同步区包括以所述线程间同步时钟指令为结束点的上一同步区和以所述线程间同步时钟指令为起点的下一同步区;

[0088] 第二获取模块402,用于获取与所述线程间同步时钟指令相对应的线程组以及所述线程组中所有线程在所述下一同步区的第一历史执行统计信息;

[0089] 竞争规则确定模块403,用于根据所述第一历史执行统计信息,确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则;

[0090] 执行模块404,用于所述当前线程执行所述下一同步区时,基于所述共享Cache资源竞争规则与其他线程竞争使用共享Cache资源。

[0091] 在一个实施例中,第一获取模块401,还用于:

[0092] 一个所述线程组中的各个线程在同一个同步区具有相同的起点指令的指令标记和相同的结束点指令的指令标记,一个所述线程组中的各个线程在不同的同步区具有不同的起点指令的指令标记和不同的结束点指令的指令标记;

[0093] 根据所述起点指令的指令标记和所述结束点指令的指令标记,获取所述线程间同步时钟指令所对应同步区的信息。

[0094] 在一个实施例中,竞争规则确定模块403,还用于:

[0095] 根据所述第一历史执行统计信息,判断所述当前线程所对应的应用程序在所述下一同步区的执行特征是否具有可预测性;

[0096] 若所述执行特征具有可预测性,则根据所述第一历史执行统计信息,判断所述应用程序的执行速度对共享Cache竞争是否敏感;

[0097] 若所述应用程序的执行速度对共享Cache竞争敏感,则确定所述当前线程执行所述下一同步区的共享Cache资源竞争规则。

[0098] 在一个实施例中,竞争规则确定模块403,还用于:

[0099] 根据所述线程间同步时钟指令,获取所述线程组中每个线程的当前时间,并根据所述第一历史执行统计信息预测所述下一同步区的预测执行时长;

[0100] 针对每个线程,基于所述当前时间和所述预测执行时长,获得每个线程执行所述

下一同步区的预测结束时间；

[0101] 对所述线程组中所述预测结束时间越晚的线程设置越大的竞争优势，基于所述竞争优势的设置规则形成所述共享Cache资源竞争规则。

[0102] 在一个实施例中，执行模块404，还用于：

[0103] 所述当前线程在开始执行所述下一同步区时，将所述共享Cache资源竞争规则配置到共享Cache硬件设置上；在所述当前线程执行所述下一同步区的过程中，所述共享Cache硬件根据所述共享Cache资源竞争规则在所述线程组中分配Cache资源；在所述当前线程执行完所述下一同步区时，将所述共享Cache硬件设置恢复为无资源竞争的情况；

[0104] 所述根据所述共享Cache资源竞争规则在所述线程组中分配Cache资源包括：让竞争优势越大的线程获得越大的共享Cache竞争容量，或者让竞争优势越大的线程的数据在共享Cache中被替换时的比率越低。

[0105] 在一个实施例中，竞争规则确定模块403，还用于：

[0106] 若所述应用程序由共享Cache访问缺失所引发的等待时间在所述下一同步区执行时长中所占比例超过第一预设阈值，则判断所述应用程序的执行速度对共享Cache竞争敏感；

[0107] 或者，若使用或改变所述共享Cache资源竞争规则后，共享Cache访问缺失次数以及所述下一同步区的执行时长的变化值超过第二预设阈值，则判断所述应用程序的执行速度对共享Cache竞争敏感。

[0108] 在一个实施例中，所述装置还包括：

[0109] 判断模块，判断所述应用程序的执行速度对内存总线的竞争是否敏感；

[0110] 内存总线竞争规则确定模块，用于若所述应用程序的执行速度对所述内存总线的竞争敏感，则确定所述当前线程在所述下一同步区的内存总线竞争规则，并让所述当前线程执行所述下一同步区时，基于所述内存总线竞争规则与其他线程竞争使用内存总线。

[0111] 本发明实施例实现了如下技术效果：当前线程执行线程间同步时钟指令，获取线程间同步时钟指令所对应同步区，同步区包括以线程间同步时钟指令为结束点的上一同步区和以线程间同步时钟指令为起点的下一同步区；获取与线程间同步时钟指令相对应的线程组以及线程组中所有线程在下一同步区的第一历史执行统计信息；根据第一历史执行统计信息，确定当前线程执行下一同步区的共享Cache资源竞争规则；当前线程执行下一同步区时，基于共享Cache资源竞争规则与其他线程竞争使用共享Cache资源。本申请通过对多个线程执行同步区时设置共享Cache资源竞争规则，实现了对于负载不平衡的多个线程，让负载不同的线程在竞争使用共享Cache资源时具有不同的竞争优势，从而使共享Cache资源得到合理利用，进而加速负载大线程的运行，从而加速整个并行程序的运行。

[0112] 显然，本领域的技术人员应该明白，上述的本发明实施例的各模块或各步骤可以用通用的计算装置来实现，它们可以集中在单个的计算装置上，或者分布在多个计算装置所组成的网络上，可选地，它们可以用计算装置可执行的程序代码来实现，从而，可以将它们存储在存储装置中由计算装置来执行，并且在某些情况下，可以以不同于此处的顺序执行所示出或描述的步骤，或者将它们分别制作成各个集成电路模块，或者将它们中的多个模块或步骤制作成单个集成电路模块来实现。这样，本发明实施例不限制于任何特定的硬件和软件结合。

[0113] 以上所述仅为本发明的优选实施例而已,并不用于限制本发明,对于本领域的技术人员来说,本发明实施例可以有各种更改和变化。凡在本发明的精神和原则之内,所作的任何修改、等同替换、改进等,均应包含在本发明的保护范围之内。

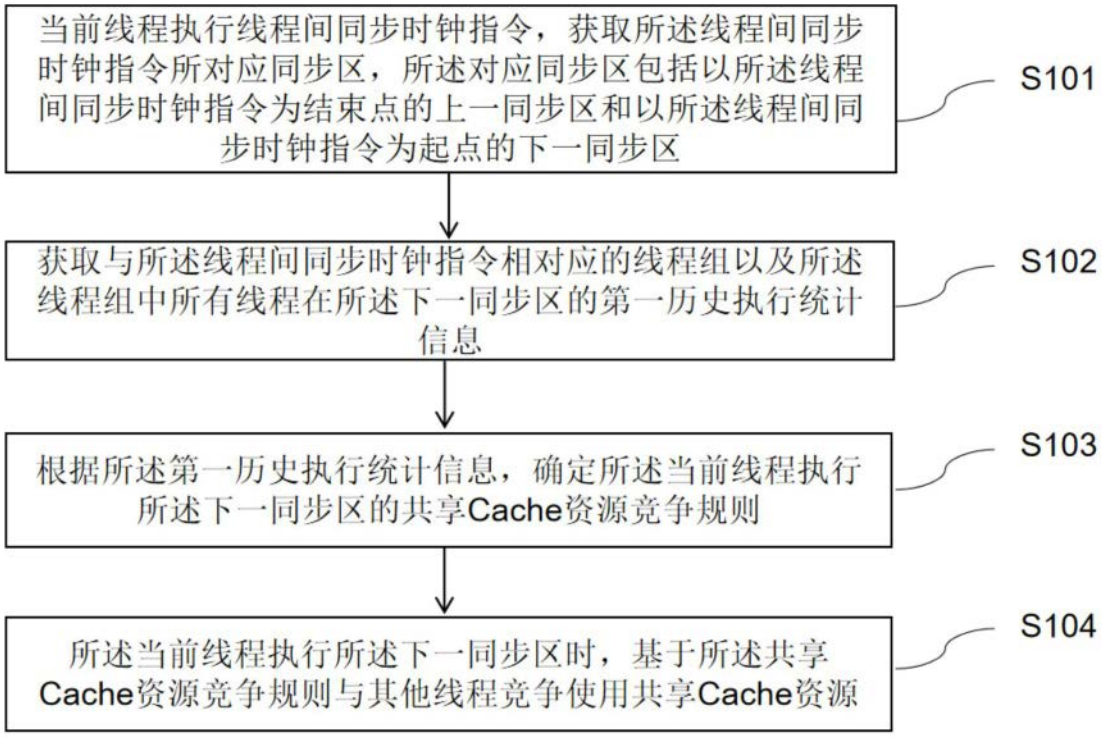


图1

```
for (int i = 0; i < 100; i++) {
    do_sync_clock(t_group, “sync1”);
    do_comp1;
    do_sync_clock(t_group, “sync2”);
    do_comp2;
    do_sync_clock(t_group, “sync3”);
    thread_synchronize;
    do_sync_clock(t_group, “sync4”);
}
```

图2

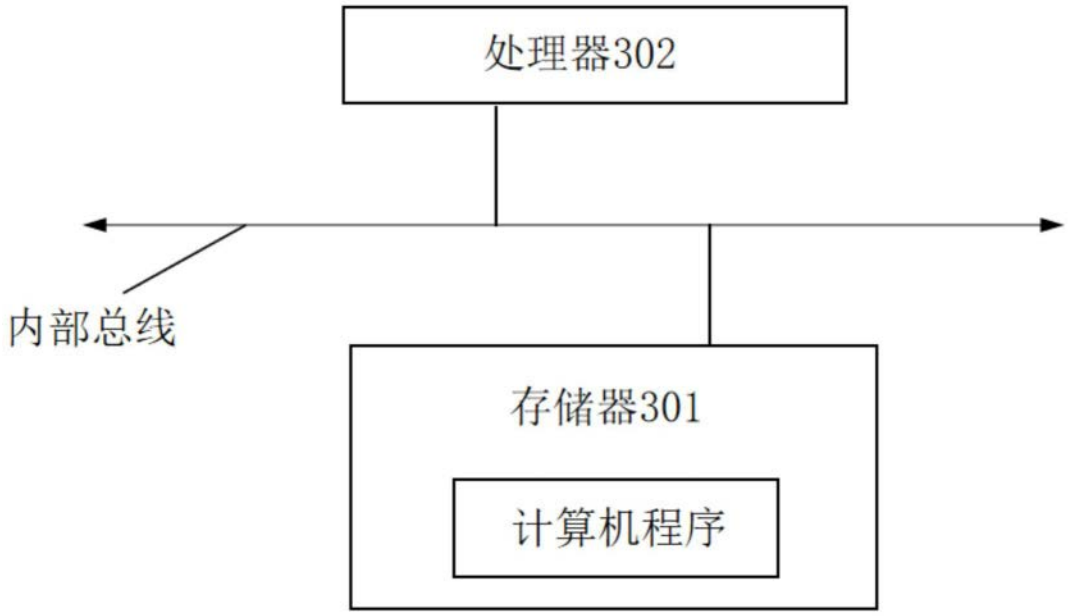

处理器302
内部总线
存储器301
计算机程序

图3


第一获取模块
401
第二获取模块
402
竞争规则确定模块
403
执行模块
404

图4