(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111742301 B
(45)授权公告日 2024.11.08

(21)申请号 201980014496.9
(22)申请日 2019.02.14
(65)同一申请的已公布的文献号
申请公布号 CN 111742301 A
(43)申请公布日 2020.10.02
(30)优先权数据
15/904,072 2018.02.23 US
15/947,699 2018.04.06 US
(85)PCT国际申请进入国家阶段日
2020.08.20
(86)PCT国际申请的申请数据
PCT/US2019/017912 2019.02.14
(87)PCT国际申请的公布数据
WO2019/164730 EN 2019.08.29

(73)专利权人 微软技术许可有限责任公司
地址 美国华盛顿州
(72)发明人 J·莫拉 H·加布里杰尔斯基
(74)专利代理机构 北京世辉律师事务所 16093
专利代理师 李峥宇

(51)Int.Cl.
G06F 11/34(2006.01)
G06F 11/36(2006.01)
G06F 11/30(2006.01)
(56)对比文件
CN 104750459 A,2015.07.01
CN 107533507 A,2018.01.02
审查员 刘凤娇

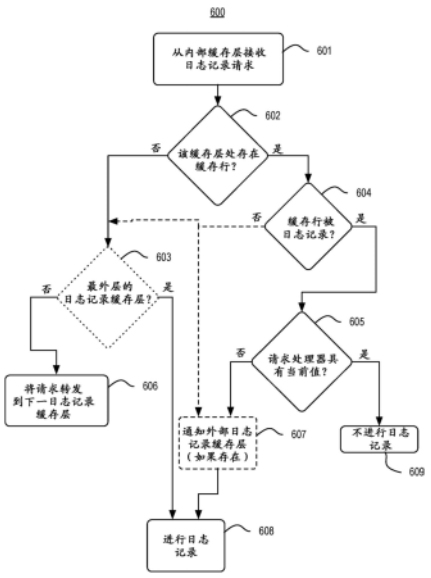
权利要求书3页 说明书25页 附图8页

(54)发明名称

通过请求来日志记录对更高级别的缓存的缓存流入量

(57)摘要

基于上层缓存层的跟踪日志记录,确定下层缓存层如何记录流入量。第二缓存从下层第一缓存接收引用存储器地址的日志记录请求。第二缓存确定它是否具有针对存储器地址的缓存行。当存在缓存行时,第二缓存将请求转发到下一日志记录缓存层,或者如果第二缓存是最外面的日志记录层,则使缓存行被日志记录。当不存在缓存行时,当缓存行不被第二缓存确定为被日志记录,或者当缓存行被第二缓存确定为被日志记录但未确定第一缓存是否知道第二缓存中缓存行的当前值时,第二缓存使缓存行被日志记录。



CN 111742301 B

1.一种微处理器,包括:

多个处理单元;

被布置到多个缓存层中的多个缓存,所述多个缓存包括第一缓存层内的多个第一缓存和第二缓存层内的一个或多个第二缓存,所述第二缓存层中的特定第二缓存用作针对所述第一缓存层中的至少特定第一缓存的后备存储库;以及

控制模块,将至少所述特定第二缓存配置为执行以下至少一项:

从所述特定第一缓存接收参考特定存储器地址的日志记录请求;以及

基于所述请求,确定所述特定第二缓存中是否存在与所述存储器地址相对应的缓存行,并且

当所述特定第二缓存中不存在所述缓存行时,执行以下中的一项:

当不存在参与日志记录并且充当针对至少所述特定第二缓存的后备存储库的第三缓存时,使所述缓存行被日志记录;或者

当所述第三缓存确实存在时,将所述请求转发到所述第三缓存;或者

当所述特定第二缓存中存在所述缓存行时,执行以下至少一项:

当(i)所述缓存行未被所述特定第二缓存确定为被日志记录,或(ii)所述缓存行被所述特定第二缓存确定为被日志记录,但所述特定第二缓存尚未确定所述第一缓存知道所述特定第二缓存的所述缓存行中存储的当前值时,使所述缓存行被日志记录;或

者

当(i)所述缓存行被所述特定第二缓存确定为被日志记录并且(ii)确定所述第一缓存知道所述特定第二缓存的所述缓存行中存储的所述当前值时,确定所述缓存行不需要被日志记录。

2.根据权利要求1所述的微处理器,其中使所述缓存行被日志记录包括:

将所述缓存行日志记录在跟踪缓冲器内;以及

将所述缓存行标记为被日志记录在所述特定第二缓存内。

3.根据权利要求1所述的微处理器,其中使所述缓存行被日志记录包括以下中的一项:

指令所述第一缓存直接日志记录所述特定存储器地址的值;或者

指令所述第一缓存日志记录对所述特定存储器地址的先前日志条目的参考。

4.根据权利要求1所述的微处理器,其中当所述缓存行未被所述特定第二缓存确定为被日志记录时,使所述缓存行被日志记录包括:

确定存在所述第三缓存,以及

通知所述第三缓存所述缓存行已由所述特定第二缓存按值日志记录。

5.根据权利要求1所述的微处理器,其中所述第一缓存层包括L1缓存层,并且其中所述第二缓存层包括L2缓存层或L3缓存层。

6.根据权利要求1所述的微处理器,所述控制模块还将至少所述特定第二缓存配置为:

从所述第一缓存接收消息,所述消息指示所述第一缓存中还与所述存储器地址相对应的另一缓存行被标记为未被日志记录在所述第一缓存内;以及

基于所述消息,将所述缓存行标记为未被日志记录在所述特定第二缓存内。

7.根据权利要求6所述的微处理器,其中基于所述另一缓存行由与所述第一缓存相对应的处理单元在针对所述处理单元禁用日志记录的情况下写入,所述另一缓存行被标记为

未被日志记录在所述第一缓存内。

8.根据权利要求1所述的微处理器,所述控制模块还将至少所述特定第二缓存配置为:

从所述第一缓存接收请求所述缓存行进行写入的消息;以及

将所述缓存行发送到所述第一缓存,至少基于针对所述第一缓存禁用日志记录,所述缓存行被标记为未被日志记录。

9.一种上层缓存层基于下层缓存层的日志记录请求来确定所述下层缓存层如何日志记录流入量的方法,所述方法在计算设备处被实现,所述计算设备包括(i)多个处理单元,(ii)被布置到多个缓存层中的多个缓存,所述多个缓存包括第一缓存层内的多个第一缓存和第二缓存层内的一个或多个第二缓存,所述第二缓存层中的特定第二缓存用作针对所述第一缓存层中的至少特定第一缓存的后备存储库,所述方法包括:

从所述特定第一缓存接收参考特定存储器地址的日志记录请求;以及

基于所述请求,确定所述特定第二缓存中是否存在与所述存储器地址相对应的缓存行,并且

当所述特定第二缓存中不存在所述缓存行时,执行以下中的一项:

当不存在参与日志记录并且充当针对至少所述特定第二缓存的后备存储库的第三缓存时,使所述缓存行被日志记录;或者

当所述第三缓存确实存在时,将所述请求转发到所述第三缓存;或者

当所述特定第二缓存中存在所述缓存行时,执行以下至少一项:

当(i)所述缓存行未被所述特定第二缓存确定为被日志记录,或(ii)所述缓存行被所述特定第二缓存确定为被日志记录,但所述特定第二缓存尚未确定所述第一缓存知道所述特定第二缓存的所述缓存行中存储的当前值时,使所述缓存行被日志记录;或者

当(i)所述缓存行被所述特定第二缓存确定为被日志记录并且(ii)确定所述第一缓存知道所述特定第二缓存的所述缓存行中存储的所述当前值时,确定所述缓存行不需要被日志记录。

10.根据权利要求9所述的方法,其中使所述缓存行被日志记录包括:

将所述缓存行日志记录在跟踪缓冲器内;以及

将所述缓存行标记为被日志记录在所述特定第二缓存内。

11.根据权利要求9所述的方法,其中使所述缓存行被日志记录包括以下中的一项:

指令所述第一缓存直接日志记录所述特定存储器地址的值;或者

指令所述第一缓存日志记录对所述特定存储器地址的先前日志条目的参考。

12.根据权利要求9所述的方法,其中当所述缓存行未被所述特定第二缓存确定为被日志记录时,使所述缓存行被日志记录包括:

确定存在所述第三缓存,以及

基于所述第三缓存的知识,使所述缓存行通过参考被日志记录。

13.根据权利要求9所述的方法,其中当所述缓存行未被所述特定第二缓存确定为被日志记录时,使所述缓存行被日志记录包括:

确定存在所述第三缓存,以及

通知所述第三缓存所述缓存行已由所述特定第二缓存按值日志记录。

14.根据权利要求9所述的方法,其中所述第一缓存层包括L1缓存层,并且其中所述第

二缓存层包括L2缓存层或L3缓存层。

15.根据权利要求9所述的方法,所述方法还包括:

从所述第一缓存接收消息,所述消息指示所述第一缓存中还与所述存储器地址相对应的另一缓存行被标记为未被日志记录在所述第一缓存内;以及

基于所述消息,将所述缓存行标记为未被日志记录在所述特定第二缓存内。

16.根据权利要求15所述的方法,其中基于所述另一缓存行由与所述第一缓存相对应的处理单元在针对所述处理单元禁用日志记录的情况下写入,所述另一缓存行被标记为未被日志记录在所述第一缓存内。

17.根据权利要求9所述的方法,所述方法还包括:

从所述第一缓存接收请求所述缓存行进行写入的消息;以及

将所述缓存行发送到所述第一缓存,至少基于针对所述第一缓存禁用日志记录,所述缓存行被标记为未被日志记录。

18.一种微处理器,包括:

多个处理单元;

被布置到多个缓存层中的多个缓存,所述多个缓存包括第一缓存层内的多个第一缓存和第二缓存层内的一个或多个第二缓存,所述第二缓存层用作针对所述第一缓存层中的至少特定第一缓存的后备存储库,所述特定第一缓存对应于所述多个处理单元中的特定处理单元;以及

控制模块,将至少所述特定第一缓存配置为执行以下至少一项:

检测对具有已被设置的日志记录状态的所述第一缓存中的缓存行的写入;

基于检测到所述写入,确定日志记录是否针对所述特定处理单元而被启用,以及

至少基于日志记录针对所述特定处理单元而被禁用,清除针对所述缓存行的所述日志记录状态,并且

向所述一个或多个第二缓存中的至少一个第二缓存通知所述至少一个第二缓存的针对所述缓存行的日志记录状态应被清除,或者

至少基于日志记录针对所述特定处理单元而被启用,保留针对所述缓存行的所述日志记录状态。

19.根据权利要求18所述的微处理器,其中所述特定处理单元具有处于拥有的状态的所述缓存行,并且其中所述控制模块还将至少所述特定第一缓存配置为向所述多个第一缓存中的至少一个第一缓存通知所述至少一个第一缓存的针对所述缓存行的日志记录状态应被清除。

20.根据权利要求18所述的微处理器,其中基于从所述多个第一缓存中的所述至少一个第一缓存接收的请求,所述特定第一缓存向所述多个第一缓存中的所述至少一个第一缓存通知所述至少一个第一缓存的针对所述缓存行的日志记录状态应被清除。

通过请求来日志记录对更高级别的缓存的缓存流入量

背景技术

[0001] 当在软件应用的开发期间编写代码时,开发人员通常花费大量时间来“调试”代码以查找运行时和其他源代码错误。这样做时,开发人员可以采用几种方法来重现和定位源代码缺陷(bug),例如,基于不同的输入来观察程序的行为、插入调试代码(例如,打印变量值、跟踪执行分支等)、临时删除代码部分等。跟踪运行时错误以查明代码错误可能会占用应用开发时间的很大一部分。

[0002] 为了协助开发人员进行代码调试过程,许多类型的调试应用(“调试器”)已经被开发。这些工具向开发人员提供了跟踪(trace)计算机代码的执行、将其可视化和对其进行改变的能力。例如,调试器可以将代码指令的执行可视化、可以在代码执行期间的不同时间呈现代码变量值、可以使得开发人员能够改变代码执行路径、和/或可以使得开发人员能够在感兴趣的代码元素上设置“断点”和/或“观察点”(“断点”和/或“观察点”在执行期间被到达时,使得将代码的执行被暂停)等。

[0003] 新兴的调试应用形式实现了“时间旅行(time travel)”、“反向”或“历史”调试。通过“时间旅行”调试,程序(例如,诸如线程的可执行实体)的执行由跟踪应用记录(record)/跟踪到一个或多个跟踪文件中。这些跟踪文件然后可以用于稍后重播程序的执行,以进行前向和后向分析。例如,“时间旅行”调试器可以使得开发人员能够设置前向断点/观察点(如常规调试器)以及反向断点/观察点。

[0004] 几个注意事项在记录跟踪文件时可以被考虑。最突出的是,在所记录的跟踪数据的稳健性与通过跟踪程序所产生的开销之间存在固有的权衡。这些权衡主要体现在跟踪文件的大小以及对所跟踪的程序的执行的性能影响。而且,由于跟踪可能在硬件协助下完成(或完全在软件中完成),因此可能还存在硬件设计和其他硬件成本方面的考虑。

发明内容

[0005] 本文描述的实施例涉及用于使用硬件辅助由处理器来创建比特精确的“时间旅行”跟踪记录的机制。这些机制基于使用至少两个层级(tier)或层(layer)的处理器缓存来跟踪跨多个处理单元的执行效果。一个机制修改了处理器的硬件和/或微代码,使得当它基于所跟踪的处理单元的活动而检测对内部或“下层”处理器缓存的流入量(influx)(即,缓存未命中(miss))时,该机制检查一个或多个外部或“上层”共享处理器缓存,来确定该流入量的数据是否已经代表另一个所跟踪的处理单元而被日志记录(log)。另一机制修改了处理器的硬件和/或微代码,使得一个或多个缓存层被配置为从(多个)下层缓存层接收日志记录请求,并使用其对所日志记录的缓存行的了解来确定对下层缓存层的流入量(如果存在)应如何被记录。任一机制可以使得流入量能够通过参考先前的日志条目而被日志记录,并且每个机制可以被扩展到“N”级(level,又译作级别)缓存。使用任一机制来记录跟踪文件可能只需要进行适度的处理器修改,并且与先前的跟踪记录方法相比,它们可以将跟踪记录的性能影响以及跟踪文件的大小减少若干数量级。

[0006] 第一实施例涉及包括多个处理单元、多个N级缓存和(N+i)级缓存的(多个)计算设

备。(N+i)级缓存与多个N级缓存中的两个或更多个N级缓存相关联,并且被配置为针对多个N级缓存的后备存储库。在这些实施例中,(多个)计算设备包括控制逻辑,控制逻辑将(多个)计算设备配置为检测对多个N级缓存中的第一N级缓存的流入量,并且其中流入量包括存储器位置处存储的数据。控制逻辑还将(多个)计算设备配置为检查(N+i)级缓存来确定针对存储器位置的数据是否先前已代表第二处理单元被日志记录。控制逻辑还基于该检查将(多个)计算设备配置为执行以下中的一项:通过参考先前已代表第二处理单元被日志记录的日志数据(即,当针对存储器位置的数据先前已代表第二处理单元被日志记录时),使针对存储器位置的数据代表第一处理单元被日志记录,或(ii)使针对存储器位置的数据代表第一处理单元按值被日志记录(即,当针对存储器位置的数据先前尚未代表第二处理单元被日志记录时)。

[0007] 第二实施例涉及包括多个处理单元以及被布置到多个缓存层中的多个缓存的(多个)计算设备。多个缓存包括第一缓存层内的多个第一缓存,以及第二缓存层内的一个或多个第二缓存。第二缓存层中的特定第二缓存用作针对第一缓存层中的至少特定第一缓存的后备存储库。在这些实施例中,(多个)计算设备包括控制逻辑,控制逻辑将至少特定第二缓存配置为从特定第一缓存接收参考特定存储器地址的日志记录请求。基于该请求,特定第二缓存确定特定第二缓存中是否存在与存储器地址相对应的缓存行。当特定第二缓存中不存在缓存行时,(i)当不存在参与日志记录并且充当针对至少特定第二缓存的后备存储库的第三缓存时,第二缓存使缓存行被日志记录;或者(ii)当第三缓存确实存在时,第二缓存将请求转发到第三缓存。

[0008] 当特定第二缓存中存在缓存行时,(i)当缓存行未被特定第二缓存确定为被日志记录、或由特定第二缓存确定为被日志记录,但是特定第二缓存尚未确定第一缓存知道特定第二缓存的缓存行中存储的当前值时,第二缓存使缓存行被日志记录;或者(ii)当缓存行被特定第二缓存确定为被日志记录,并且确定第一缓存知道特定第二缓存的缓存行中存储的当前值时,第二缓存确定缓存行不需要被日志记录。

[0009] 本文描述的任何实施例也可以被实现为由(多个)计算设备(例如,诸如微处理器)执行的(多个)方法和/或在硬件存储设备上存储的并且可执行来执行(多个)方法的计算机可执行指令(例如,处理器微代码)。

[0010] 提供本发明内容是为了以简化的形式介绍一些概念,这些概念将在下面的具体实施方式中进一步描述。本发明内容既不旨在标识所要求保护的主题的关键特征或必要特征,也不旨在用于帮助确定所要求保护的主题的范围。

附图说明

[0011] 为了描述可以获得本发明的上述以及其他优点和特征的方式,将通过参考在附图中图示的本发明的特定实施例来对以上简要描述的本发明进行更具体的描述。理解这些附图仅描绘了本发明的典型实施例,因此不应认为是对本发明范围的限制,将通过使用附图以附加的特征和细节来描述和解释本发明,在附图中:

[0012] 图1图示了示例计算环境,示例计算环境有助于使用至少两个层级或层的处理器缓存来记录跨多个处理单元的“比特精确”执行跟踪;

[0013] 图2A图示了包括多层缓存的示例计算环境;

[0014] 图2B图示了缓存的一个示例;

[0015] 图3图示了用于跟踪记录的示例方法的流程图,该跟踪记录是基于以下来进行的:基于对一个或多个上级缓存的了解,通过参考先前的日志数据,而记录对下级缓存的流入量;

[0016] 图4A图示了示例共享缓存,其中每个缓存行包括一个或多个附加记账比特;

[0017] 图4B图示了共享缓存的一个示例,共享缓存包括一个或多个保留的缓存行,以用于存储应用于常规缓存行的记账比特;

[0018] 图5图示了系统存储器与缓存之间的组相联(set-associative)映射的示例;

[0019] 图6图示了上层缓存层的示例方法的流程图,该方法基于下层缓存层的日志记录请求来确定如何日志记录下层缓存层的流入量;

[0020] 图7图示了当处理单元在启用日志记录与禁用日志记录之间转换时,用于管理缓存行的日志记录状态的示例方法的流程图;

[0021] 图8图示了当具有禁用日志记录的处理单元仅从亲代(parent)缓存接收用于写入的缓存行时,用于管理缓存行的日志记录状态的示例方法的流程图;以及

[0022] 图9图示了当处理单元向处理单元已处于“拥有的”缓存一致性协议状态的缓存行写入时,用于管理缓存行的日志记录状态的示例方法的流程图。

具体实施方式

[0023] 本文中描述的实施例涉及用于使用硬件辅助由处理器来创建比特精确的“时间旅行”跟踪记录的机制。这些机制基于使用至少两个层级或层的处理器缓存来跟踪跨多个处理单元的执行效果。一个机制修改了处理器的硬件和/或微代码,使得当它基于所跟踪的处理单元的活动而检测对内部或“下层”处理器缓存的流入量(即,缓存未命中)时,它检查一个或多个外部或“上层”共享处理器缓存,来确定该流入量的数据是否已被另一个所跟踪的处理单元日志记录。另一机制修改了处理器的硬件和/或微代码,使得一个或多个缓存层被配置为从(多个)下层缓存层接收日志记录请求,并使用其对所日志记录的缓存行的了解来确定对下层缓存层的流入量(如果存在)应如何被日志记录。任一机制可以使得流入量能够通过参考先前的日志条目而被日志记录,并且每个机制可以被扩展到“N”级缓存。使用任一机制来记录跟踪文件可能只需要进行适度的处理器修改,并且当与先前的跟踪记录方法相比时,它们可以将跟踪记录的性能影响以及跟踪文件大小减少若干数量级。

[0024] 图1图示了示例计算环境100,示例计算环境100使用至少两个层级或层的处理器缓存来记录跨多个处理单元的“比特精确”的执行跟踪。如所描绘的,实施例可以包括或利用专用或通用计算机系统101,专用或通用计算机系统101包括计算机硬件,诸如,例如,一个或多个处理器102、系统存储器103、一个或多个数据存储库104和/或输入/输出硬件105。

[0025] 本发明范围内的实施例包括用于承载或存储计算机可执行指令和/或数据结构的物理和其他计算机可读介质。这样的计算机可读介质可以是可以由计算机系统101访问的任何可用介质。存储计算机可执行指令和/或数据结构的计算机可读介质是计算机存储设备。承载计算机可执行指令和/或数据结构的计算机可读介质是传输介质。因此,通过示例而非限制,本发明的实施例可以包括至少两个明显不同种类的计算机可读介质:计算机存储设备和传输介质。

[0026] 计算机存储设备是存储计算机可执行指令和/或数据结构的物理硬件设备。计算机存储设备包括各种计算机硬件,诸如,RAM、ROM、EEPROM、固态驱动器(“SSD”)、闪存、相变存储器(“PCM”)、光盘存储装置、磁盘存储装置或其他磁性存储设备、或可以用于以计算机可执行指令或数据结构的形式存储程序代码的任何其他(多个)硬件设备,并且该程序代码可以由计算机系统101访问和执行来实现本发明所公开的功能性。因此,例如,计算机存储设备可以包括可以存储计算机可执行指令和/或数据结构的所描绘的系统存储器103、所描绘的数据存储库104、或如稍后所讨论的其他存储装置(诸如,处理器上存储装置)。

[0027] 传输介质可以包括可以用于以计算机可执行指令或数据结构的形式承载程序代码的网络和/或数据链路,并且该程序代码可以由计算机系统101访问。“网络”被定义为使得能够在计算机系统和/或模块和/或其他电子设备之间传输电子数据的一个或多个数据链路。当信息通过网络或另一通信连接(硬连线、无线或硬连线或无线的组合)被传递或提供给计算机系统时,计算机系统可以将连接视为传输介质。上述的组合也应被包括在计算机可读介质的范围内。例如,输入/输出硬件105可以包括连接可以用于以计算机可执行指令或数据结构的形式承载程序代码的网络和/或数据链路的硬件(例如,网络接口模块(例如,“NIC”))。

[0028] 此外,在到达各种计算机系统组件时,计算机可执行指令或数据结构形式的程序代码可以从传输介质被自动传递到计算机存储设备(反之亦然)。例如,通过网络或数据链路接收的计算机可执行指令或数据结构可以被缓冲在NIC(例如,输入/输出硬件105)内的RAM中,然后最终被传递到系统存储器103和/或计算机系统101处的易失性较小的数据存储设备(例如,数据存储库104)。因此,应当理解,计算机存储设备可以被包括在也(或者甚至主要地)利用传输介质的计算机系统组件中。

[0029] 计算机可执行指令包括例如当在(多个)处理器102处被执行时,使计算机系统101执行特定功能或功能群组(group)的指令和数据。计算机可执行指令可以是例如二进制、中间格式指令(例如,汇编语言)或者甚至源代码。

[0030] 本领域技术人员将理解,本发明可以在具有许多类型的计算机系统配置的网络计算环境中被实践,计算机系统配置包括:个人计算机、台式计算机、膝上型计算机、消息处理器、手持式设备、多处理器系统、基于微处理器或可编程的消费类电子产品、网络PC、小型计算机、大型计算机、移动电话、PDA、平板电脑、寻呼机、路由器、交换机等。本发明也可以在分布式系统环境中被实践,在分布式系统环境中,借助网络链接(通过硬连线数据链路、无线数据链路或通过硬连线和无线数据链路的组合)的本地和远程计算机系统均执行任务。这样,在分布式系统环境中,计算机系统可以包括多个组成计算机系统。在分布式系统环境中,程序模块可以位于本地和远程存储器存储设备中。

[0031] 本领域技术人员还将理解,本发明可以在云计算环境中被实践。云计算环境可以是分布式的,但是这不是必需的。当是分布式时,云计算环境可以在组织内国际分布和/或具有跨多个组织拥有的组件。在本说明书和所附权利要求书中,“云计算”被定义为用于使得能够对可配置计算资源(例如,网络、服务器、存储装置、应用和服务)的共享池进行按需网络访问的模型。“云计算”的定义不限于在适当部署时可以从这样的模型获得的其他众多优势中的任何优势。

[0032] 云计算模型可以由各种特性组成,例如,按需自助服务、广泛网络访问、资源池、快

速弹性、测量的服务等。云计算模型也可以采用各种服务模型的形式,诸如例如,软件即服务(“SaaS”)、平台即服务(“PaaS”)和基础架构即服务(“IaaS”)。云计算模型还可以使用诸如私有云、社区云、公共云、混合云等不同的部署模型来部署。

[0033] 一些实施例(例如,云计算环境)可以包括具有各自能够运行一个或多个虚拟机的一个或多个主机的系统。在操作期间,虚拟机将模拟操作的计算系统,从而支持操作系统,也许还支持一个或多个其他应用。在一些实施例中,每个主机包括管理程序,管理程序使用从虚拟机的角度抽象的物理资源来模拟虚拟机的虚拟资源。管理程序还在虚拟机之间提供适当的隔离。因此,从任何给定虚拟机的角度来看,即使虚拟机仅与物理资源的外观(例如,虚拟资源)接口连接,管理程序也提供了虚拟机正在与物理资源接口连接的错觉。物理资源的示例包括处理能力、存储器、磁盘空间、网络带宽、介质驱动器等。

[0034] 图1包括(多个)处理器102的内部硬件组件的简化表示。如图所示,每个处理器102包括多个处理单元102a。每个处理单元可以是物理的(即,物理处理器核心)和/或逻辑的(即,由支持超线程的物理核心提供的逻辑核心,其中多于一个应用线程在物理核心处执行)。因此,例如,即使处理器102在一些实施例中仅包括单个物理处理单元(核心),它也可以包括由该单个物理处理单元呈现的两个或更多个逻辑处理单元102a。

[0035] 每个处理单元102a执行由应用(例如,跟踪器104a、调试器104b、操作内核104c、应用104d等)限定的处理器指令,并且该指令从预定的处理器指令集架构(ISA)中被选择。每个处理器102的特定ISA基于处理器制造方和处理器模型而变化。常见的ISA包括来自INTEL公司的IA-64和IA-32架构、来自ADVANCED MICRO DEVICES公司的AMD64架构以及来自ARM HOLDINGS,PLC的各种Advanced RISC Machine(“ARM”)架构,但是大量的其他ISA存在并且可以由本发明使用。通常,“指令”是由处理器可执行的最小的外部可见(即,在处理器外部)的代码单元。

[0036] 每个处理单元102a从一个或多个处理器缓存102b获得处理器指令,并且基于(多个)缓存102b中的数据、基于寄存器102d中的数据和/或在没有输入数据的情况下来执行处理器指令。一般而言,每个缓存102b是少量(即,相对于系统存储器103的典型数量而言为少)的随机存取存储器,该随机存取存储器存储后备存储库(诸如,系统存储器103和/或(多个)缓存102b中的另一缓存)的各部分在处理器上的副本。例如,当执行应用代码103a时,一个或多个缓存102b包含应用运行时数据103b的某些部分。如果(多个)处理单元102a请求尚未被存储在特定缓存102b中的数据,则发生“缓存未命中”,并且该数据从系统存储器103或另一缓存被获取,可能从该缓存102b中“逐出”一些其他数据。

[0037] 通常,(多个)处理器缓存102b被划分为分离的层级、层或级别,诸如,层1(L1)、层2(L2)、层3(L3)等。取决于处理器实现,层级可以是处理器102本身的一部分(例如,L1和L2)和/或可以与处理器102分离(例如,L3)。因此,图1的(多个)缓存102b可以包括这些层之一(L1),或者可以包括这些层中的多个(例如,L1和L2,甚至L3)。为了进一步理解这些概念,图2A图示了展示多层缓存的示例环境200。在图2A中,存在两个处理器201a和201b(例如,每个对应于图1的不同处理器102)以及系统存储器202(例如,对应于图1的系统存储器103)。在示例环境200中,每个处理器201包括四个物理处理单元(即,用于处理器201a的单元A1-A4和用于处理器210b的单元B1-B4)。

[0038] 示例环境200还在每个处理单元201内包括三层缓存层级结构。环境200仅是一个

示例布局,并且不限于本文中的实施例可以在其中操作的缓存层级结构。在环境200中,在最下或最内层,每个处理单元与它自己的专用L1缓存相关联(例如,处理器201a中的L1缓存“L1-A1”用于单元A1,处理器201a中的L1缓存“L1-A2”用于单元A2等)。向上移动一层,每个处理单元201包括两个L2缓存(例如,用作L1缓存L1-A1和L1-A2的后备存储库的处理器201a中的L2缓存“L2-A1”,用作L1缓存L1-A3和L1-A4的后备存储库的处理器201a中的L2缓存“L1-A2”等)。最终,在最高或最外层,每个处理单元201包括单个L3缓存(例如,用作L2缓存L2-A1和L2-A2的后备存储库的处理器201a中的L3缓存“L3-A”,以及用作L2缓存L2-B1和L2-B2的后备存储库的处理器201b中的L3缓存“L3-B”)。如图所示,系统存储器202用作L3缓存L3-A和L3-B的后备存储库。

[0039] 如图2A所示,当多个缓存层被使用时,(多个)处理单元102a通常与最下层(L1)直接交互。在大多数情况下,数据在各层之间流动(例如,在读取时,L3缓存与系统存储器103交互并将数据用于L2缓存,而L2缓存又将数据用于L1缓存)。当处理单元102a执行写入操作时,缓存进行协调以确保已影响了在(多个)处理单元102a之间共享的数据的那些缓存不再具有数据。该协调使用CCP执行。

[0040] 环境200中的缓存因此可以被视为“共享”缓存。例如,每个L2和L3缓存服务于给定处理器201内的多个处理单元,并且因此被处理单元共享。即使每个缓存对应于单个处理单元,给定处理器201中的L1缓存总体上也可以视为共享的,因为单独的L1缓存可以彼此协调(即,经由CCP)来确保一致性(即,使得每个缓存的存储器位置跨所有L1缓存被一致地查看)。类似地,每个处理器201内的L2缓存可以经由CCP协调。附加地,如果处理器201支持超线程,则每个单独的L1缓存可以被视为由两个或更多个逻辑处理单元共享,因此即使在单独的级别上也可以被“共享”。

[0041] 通常,每个缓存包括多个“缓存行”。每个缓存行存储来自其后备存储库(例如,系统存储器202或更高层的缓存)的存储器块。例如,图2B图示了缓存203的至少一部分的示例,缓存203包括多个缓存行206,每个缓存行206至少包括地址部分204和值部分205。每个缓存行206的地址部分204被配置为在缓存行所对应的系统存储器202中存储地址,并且值部分205最初存储从系统存储器202接收的值。值部分205可以被处理单元修改,并且最终被逐出回到后备存储库。如省略号所指示,缓存203可以包括大量的缓存行。例如,当代的64位INTEL处理器可以包含具有512个或更多缓存行的单独L1缓存。在这样的缓存中,每个缓存行通常可用于存储6字节(48比特)至8字节(64比特)存储器地址的64字节(512比特)值。如图2A直观所示,缓存大小通常随每一层增加(即,L2缓存通常大于L1缓存,L3缓存通常大于L2缓存等)。

[0042] 每个缓存行206的地址部分204中存储的地址可以是物理地址诸如,系统存储器202中的实际存储器地址。备选地,地址部分204中存储的地址可以是虚拟地址,虚拟地址是被映射到物理地址以提供抽象(例如,使用操作系统管理的页表(page table))的地址。这样的抽象可以用于例如促进在(多个)处理器102处执行的不同过程之间的存储器隔离,包括用户模式过程和与操作系统内核104b相关联的内核模式过程之间的隔离。当虚拟地址被使用时,处理器102可以包括转译旁视(translation lookaside)缓冲器(TLB)102f(通常是存储器管理单元(MMU)的一部分),转译旁视缓冲器(TLB)102f维持物理地址和虚拟地址之间最近使用的存储器地址映射。

[0043] (多个)缓存102b可以包括代码缓存部分和数据缓存部分。当执行应用代码103a时,(多个)缓存102b的(多个)代码部分可以存储应用代码103a中存储的处理器指令的至少一部分并且(多个)缓存102b的(多个)数据部分可以存储应用运行时数据103b的数据结构的至少一部分。另外,缓存可以是包含性的、独占的(exclusive)、也可以包括包含性和独占的行为。例如,在包含性缓存中,L3层通常将数据的超集存储在L3层下面的L2层中,而L2层将L1层的超集存储在它们之下。在独占缓存中,各层可以不相交,例如,如果L3缓存中存在L1缓存需要的数据,则它们可以交换信息,诸如,数据、地址等。

[0044] 返回到图1,每个处理器102还包括微代码102c,微代码102c包括控制处理器102的操作的控制逻辑(即,可执行指令),并且该控制逻辑通常用作处理器的硬件和由处理器102向执行应用公开的处理器ISA之间的解释器。微代码102通常体现在诸如ROM、EEPROM等的处理器上存储装置中。

[0045] 寄存器102d是基于(多个)处理器102的ISA被定义并通过处理器指令被读取和/或写入的基于硬件的存储位置。例如,寄存器102d通常用于存储从(多个)缓存102b获取的值以供指令使用、存储执行指令的结果和/或存储状况(status)或状态(诸如,执行指令的一些副作用(例如,值改变的符号、值达到零、进位的发生等))、处理器循环计数等。因此,一些寄存器102d可以包括用于发信号通知由执行处理器指令引起的某种状态改变的“标志(flag)”。在一些实施例中,处理器102还可以包括用于控制处理器操作的不同方面的控制寄存器。尽管图1将寄存器102d描绘为单个框,但是将理解,每个处理单元102a通常包括该处理单元专用的一个或多个对应寄存器组102d。

[0046] 在一些实施例中,(多个)处理器102可以包括一个或多个缓冲器102e。如下文将要讨论的,(多个)缓冲器102e可以用作跟踪数据的临时存储位置。因此,例如,(多个)处理器102可以将跟踪数据的各部分存储在(多个)缓冲器102e中,并且在适当的时间(例如,当存在可用的存储器总线带宽和/或空闲的处理器循环时),将该数据刷新到跟踪数据存储库104e。

[0047] 如上所述,处理器根据一个或多个CCP在(多个)缓存102b上操作。通常,CCP限定当各种处理单元102a从各种缓存102b中读取和写入数据时,各种缓存102b之间的数据之间的一致性如何被保持,以及如何确保各种处理单元102a始终从(多个)缓存102b中的给定位置读取有效数据。CCP与处理器102的ISA定义的存储器模型相关并启用该存储器模型。

[0048] 通用CCP的示例包括MSI协议(即,修改的、共享的和无效的)、MESI协议(即,修改的、独占的、共享的和无效的)和MOESI协议(即,修改的、拥有的、独占的、共享的和无效的)。这些协议中的每一个为(多个)缓存102b中的单独位置(例如,行)限定状态。“修改的”缓存位置包含已在(多个)缓存102b中被修改并且因此可能与后备存储库(例如,系统存储器103或另一缓存)中的对应数据不一致的数据。当具有“修改的”状态的位置从(多个)缓存102b被逐出时,通用CCP需要缓存来保证其数据被写回到后备存储库,或者由另一缓存接管该责任。“共享的”缓存位置包含未从后备存储库中的数据进行修改、以只读状态存在并且被(多个)处理单元102a共享的数据。(多个)缓存102b可以在无需将其写入后备存储库的情况下,逐出该数据。“无效的”缓存位置不包含有效数据并且可以被认为是空的并且可用于存储缓存未命中的数据。“独占的”缓存位置包含与后备存储库相匹配并且仅由单个处理单元102a使用的数据。该数据可以在任何时间(即,响应于读取请求)被改变为“共享的”状态,或者可

以在对其进行写入时被改变为“修改的”状态。“拥有的”缓存位置由两个或更多个处理单元102a共享,但是处理单元之一具有对其进行改变的独占权。当该处理进行改变时,因为其他处理单元可能需要基于CCP实现来使其自身的缓存无效或更新,所以它可以通知其他处理单元。

[0049] 数据存储库104可以存储表示应用程序的计算机可执行指令,应用程序诸如例如,跟踪器104a、调试器104b、操作系统内核104c和应用104d(例如,作为由跟踪器104a进行跟踪的主题的应用)。当这些程序正在执行时(例如,使用(多个)处理器102),系统存储器103可以存储对应的运行时数据,诸如,运行时数据结构、计算机可执行指令等。因此,图1将系统存储器103图示为包括应用代码103a和应用运行时数据103b(例如,各自与应用104g相对应)。数据存储库104可以进一步存储数据结构,诸如,被存储在一个或多个跟踪数据存储库104e内的跟踪数据。如省略号104f所示,数据存储库104还可以存储其他计算机可执行指令和/或数据结构。

[0050] 跟踪器104a可用于记录一个或多个实体(例如,应用104d或内核104c的一个或多个线程)的比特精确执行跟踪,并将跟踪数据存储到跟踪数据存储库104e中。在一些实施例中,跟踪器104a是独立应用,而在其他实施例中,跟踪器104a被集成到另一软件组件(诸如,内核104c、管理程序、云结构等)中。尽管跟踪数据存储库104e被描绘为数据存储库104的一部分,但是跟踪数据存储库104e也可以至少部分地体现在系统存储器103中、(多个)缓存102b中、(多个)缓冲器102e中或某个其他存储设备处。

[0051] 如所提及的,跟踪器104a记录一个或多个实体的比特精确执行跟踪。如本文中所使用的,“比特精确”跟踪是包括以下数据的跟踪:该数据足以使得先前在一个或多个处理单元102a处被执行的代码能够被重播、使得其在重播时以与跟踪期间基本上相同的方式执行。存在跟踪器104a可以使用来记录比特精确跟踪的多种方法,每个方法都有各种优点和缺点(例如,在跟踪开销、跟踪文件大小、所需的处理器修改量等方面)。用于记录这样的数据的一些特定实施例稍后结合图3至图9来讨论。

[0052] 不论跟踪器104a使用哪种记录方法,它都可以将跟踪数据记录到一个或多个跟踪数据存储库104e中。作为示例,跟踪数据存储库104e可以包括一个或多个跟踪文件、系统存储器103的一个或多个区域、处理器缓存102b的一个或多个区域(例如,L2或L3缓存)、处理器102中的缓冲器102d或其任何组合或其中多个。跟踪数据存储库104e可以包括一个或多个跟踪数据流(stream)。在一些实施例中,例如,多个实体(例如,过程、线程等)可以各自被跟踪到分离跟踪文件或给定跟踪文件内的跟踪数据流。备选地,与每个实体相对应的数据分组可以被标记为使得它们被标识为与该实体相对应。如果多个相关实体(例如,同一过程的多个线程)将被跟踪,则针对每个实体的跟踪数据可以被独立跟踪(使得它们能够独立地重播),但是跨实体可排序的任何事件(例如,访问共享存储器)可以利用跨独立跟踪为全局的顺序号(例如,单调递增的编号)来标识。跟踪数据存储库104e可以被配置为用于对跟踪数据流的灵活管理、修改和/或创建。例如,对现有跟踪数据流的修改可能涉及对现有跟踪文件的修改、对现有文件内的跟踪数据的各部分的替换和/或对包括修改的新跟踪文件的创建。

[0053] 在一些实现中,跟踪器104a可以连续地附加到(多个)跟踪数据流,使得跟踪数据在跟踪期间连续增长。然而,在其他实现中,跟踪数据流可以被实现为一个或多个环形缓冲

器。在这样的实现中,随着新的跟踪数据被添加到跟踪数据存储库104e,最旧的跟踪数据被从(多个)数据流中移除。这样,当跟踪数据流被实现为(多个)缓冲器时,它们包含在所跟踪的(多个)过程处最近执行的滚动跟踪。环形缓冲器的使用可以使得跟踪器104a能够甚至在生产系统中也可以始终参与跟踪。在一些实现中,诸如通过设置或清除一个或多个控制寄存器中的一个或多个比特,跟踪实际上可以在任何时间被启用和禁用。这样,无论是跟踪到环形缓冲器还是附加到传统的跟踪数据流,跟踪数据都可以包括在其期间针对一个或多个处理单元102a启用跟踪的时段之间的间隙。

[0054] 调试器104b可用于将由跟踪器104a生成的跟踪数据消费(例如,重播)到跟踪数据存储库104e中,以便帮助用户对跟踪数据(或其派生)执行调试动作。例如,调试器104b可以呈现一个或多个调试接口(例如,用户接口和/或应用编程接口)、在应用104d的一个或多个部分的执行之前重播、设置断点/观察点(包括反向断点/观察点)、启用在跟踪数据上的查询/搜索等。

[0055] 返回到跟踪器104a,在本文的实施例中,跟踪器104a利用处理器102的(多个)缓存102b有效地记录应用104d和/或操作系统内核104c的比特精确的执行跟踪。这些实施例基于发明人的以下观察而被建立:处理器102(包括(多个)缓存102b)形成了半闭合或准闭合系统。例如,一旦用于过程的数据的各部分(即,代码数据和运行时应用数据)被加载到(多个)缓存102b中,则处理器102可以在没有任何输入的情况下,在时间突发内以半闭合或准闭合系统自行运行。特别地,一旦(多个)缓存102b被加载了数据,一个或多个处理单元102a就使用(多个)缓存102b的(多个)数据部分中存储的运行时数据并使用寄存器102d来执行来自(多个)缓存102b的(多个)代码部分的指令。

[0056] 当处理单元102a需要一些信息流入量时(例如,由于其正在执行、将要执行或可能执行的指令访问尚未在(多个)缓存102b中的代码或运行时数据),“缓存未命中”发生并且该信息从系统存储器103被引入(多个)缓存102b。例如,如果在所执行的指令在应用运行时数据103b内的存储器地址处执行存储器操作时发生数据缓存未命中,则来自该存储器地址的数据被引入(多个)缓存102b的数据部分的缓存行之一。类似地,如果当指令在系统存储器103中存储的存储器地址应用代码103a处执行存储器操作时发生代码缓存未命中,则来自该存储器地址的代码被引入(多个)缓存102b的(多个)代码部分的缓存行之一。处理单元102a然后使用(多个)缓存102b中的新信息继续执行,直到新信息再次被引入(多个)缓存102b中(例如,由于另一缓存未命中或未缓存的读取)。

[0057] 发明人还观察到,为了记录应用的执行的比特精确表示,当处理单元执行该应用的(多个)线程时,跟踪器104a可以记录能够将信息的流入量再现到(多个)缓存102b中的足够的数据。例如,一种记录这些流入量的方法在每个处理单元的基础上并且在最里面的缓存层(例如,L1)处操作。该方法可以涉及针对正被跟踪的每个处理单元,记录与该处理单元的L1缓存相关联的所有缓存未命中和未缓存的读取(即,来自硬件组件和不可缓存的存储器的读取)、以及执行期间每个数据片段被引入该处理单元的L1缓存中的时间(例如,使用所执行的指令计数或某个其他计数器)。如果存在可以跨处理单元被排序的事件(例如,访问共享存储器),则这些事件可以跨所产生的数据流被日志记录(例如,通过跨数据流使用单调递增(或递减)编号(MIN))。

[0058] 然而,由于L1缓存层可以包括多个不同的L1缓存,L1缓存各自与不同的物理处理

单元相关联(例如,如图2A所示),因此以这种方式进行记录可能记录重复数据,因此针对“全保真”跟踪严格需要更多数据。例如,如果多个物理处理单元从同一存储器位置读取(这在多线程应用中可能经常发生),则该方法可以日志记录针对同一存储器位置的缓存未命中以及针对多个物理处理单元中的每一个的数据。值得注意的是,如本文所使用的,“全保真”跟踪是包含足够信息以实现被跟踪实体的完全重播的任何跟踪(即使与使用替代跟踪技术可能被记录的数据相比,特定的“全保真”跟踪实际上可能包含封装相同信息的较少数据)。

[0059] 为了进一步减小跟踪文件大小,发明人开发了利用一个或多个上层缓存来避免记录该重复数据的至少一部分的改进的记录技术。相反,这些改进的技术可以通过参考先前被日志记录的数据进行日志记录,或者在许多情况下避免日志记录。

[0060] 基于处理器检查一个或多个更上层缓存层的知识,在下层缓存层处日志记录缓存未命中

[0061] 在第一实施例中,处理器基于第一处理单元的活动(诸如,从特定存储器地址读取)来检测到对内部或“下层”处理器缓存(例如,L1)的流入量(即,缓存未命中),然后检查一个或多个外部或“上层”共享处理器缓存,以确定相同数据流入量(即,相同存储器地址和被第一处理单元读取的相同值)是否已经代表第二所跟踪的处理单元被日志记录。如果是,则在可能的情况下,处理器可以通过参考第二处理单元的先前流入量,通过第一处理来日志记录该稍后的流入量。

[0062] 为了理解这些实施例,请注意,在大多数环境中,上层缓存大于其之下的下层缓存,并且它通常是多个下层缓存的后备存储库。例如,在图2A的示例环境中,每个L2缓存是两个L1缓存的后备存储库,并且每个L3缓存是两个L2缓存(并扩展为四个L1缓存)的后备存储库。因此,上层缓存可以保留有关多个下层缓存的知识(例如,在图2A中,L2缓存L1-A1可以保留关于L1缓存L1-A1和L1-A2的知识,L2缓存L1-A2可以保留关于L1缓存L1-A3和L1-A4的知识,并且L3缓存L3-A可以保留关于L2缓存L2-A1和L2-A1以及L1缓存L1-A1、L1-A2、L1-A3和L1-A4的知识)。通过利用一个或多个上层缓存层的知识,本文的实施例通过参考已代表其他处理单元被日志记录的流入量,使得能够有很多机会来日志记录由一个处理单元引起的流入量。

[0063] 根据这些第一实施例,图3图示了用于跟踪记录的方法300的示例,该跟踪记录是以下来进行的:基于一个或多个上级缓存的知识,通过参考先前的日志数据,而记录对下级缓存的流入量。现在图3在图1和图2A、图2B的上下文中被描述。

[0064] 特别地,图3在诸如处理器102或201a的环境中操作,处理器102或201a包括多个处理单元、多个N级缓存以及与多个N级缓存中的两个或更多个相关联并且被配置为多个N级缓存的后备存储库的(N+i)级缓存。在方法300(以及在权利要求中)中,N和i是正整数,即,N≥1,使得N等于1、2、3等;并且i≥1,使得i等于1、2、3等。例如,参考图2A的处理器201a,该处理器包括多个处理单元A1、A2等。处理器201a还包括多个N级缓存L1-A1、L1-A2等(即,其中N等于1)。处理器201a还包括与多个N级缓存中的两个或更多个相关联并且被配置为多个N级缓存的后备存储库的(N+i)级缓存。例如,处理器201a包括作为N级缓存L1-A1和L1-A2的后备存储库的(N+i)级缓存L2-A1(即,其中N等于1,并且i等于1)。在另一示例中,处理器201a包括作为N级缓存L1-A1、L1-A2等的后备存储库的(N+i)级缓存L3-A(即,其中N等于1,并且i

等于2)。处理器102/201a基于诸如微代码102c和/或电路逻辑的控制逻辑来操作方法300。

[0065] 如图所示,方法300包括动作301,动作301是在第一处理单元处执行期间检测对N级缓存的流入量。在一些实施例中,动作301包括检测对多个N级缓存中的第一N级缓存的流入量,该流入量包括存储器位置处存储的数据。例如,基于处理单元A1的活动(诸如,对系统存储器202的所请求的存储器访问(例如,由于对应用104c的第一线程的正常或推测性执行而引起)),在缓存L1-A1中可能发生缓存未命中(即,当N等于1时)。这样,缓存L1-A1的行获得数据流入量(包括所请求的存储器位置的过去-当前值(then-current value))。根据缓存属性(例如,存在哪些较高级的层、缓存架构是包含性的还是独占的等)和当前缓存状态,流入量可以源自系统存储器202或更高级的缓存(例如,L2-A1和/或L3-A)。

[0066] 方法300还包括基于在第二处理单元处的执行来检查(N+i)级缓存,以确定流入量的数据是否已被日志记录的动作302。在一些实施例中,动作302包括:基于检测对第一N级缓存的流入量,检查(N+i)级缓存以确定用于存储器位置的数据先前是否已被第二处理单元日志记录。例如,如果i等于1,使得(N+i)级缓存包括(N+1)级缓存,则处理器201可以检查L2缓存(诸如,L2-A1)(其具有缓存L1-A2和处理单元A2的知识)。该检查可以用于确定针对存储器位置的数据先前是否已代表处理单元A2被日志记录。例如,基于在引起缓存L1-A2中的缓存未命中的处理单元A2处应用104c的第二线程的先前执行,该数据先前已被日志记录。在备选示例中,如果i等于2,使得(N+i)级缓存包括(N+2)级缓存,则处理器201可以检查L2缓存,诸如,缓存L3-A(其具有处理器201中的所有其他缓存的知识)。该检查可以用于确定针对存储器位置的数据先前是否已代表处理单元A2-A4中的任何处理单元被日志记录(例如,基于引起缓存L1-A2、L1-A3和/或L1-A4中的(多个)缓存未命中的一个或多个处理单元A2-A4处应用104c的一个或多个其他线程的先前执行)。注意,在该第二示例中,L2缓存可以在检查中被跳过。

[0067] 如图所示,动作302可以被重复任意次数,同时每次递增i的值。尽管i每次通常递增1,但是可能存在将i递增大于1的正整数的实施例。重复动作302的效果是,递增i时将检查多个上级缓存。例如,如果i＝1,则当动作302最初运行时,处理器201可以检查L2缓存层(例如,L2-A1和/或L2-A2)。如果在L2缓存中未找到关于可适用的存储器位置的足够知识,则处理器201可以用i＝2重复动作302,从而检查L3缓存层(例如,L3-A)。对于计算环境所提供的尽可能多的缓存层,这种情况可以继续。如果i递增的值大于1,则沿途的一个或多个缓存层可以被跳过。将理解,在提供独占缓存或提供呈现混合包含性/独占行为的缓存的架构中检查多个缓存层可能是有益的。这是因为在这些架构中,可能无法保证外部缓存层包含(多个)内部缓存层中数据的完整超集。

[0068] 鉴于前述内容,将理解,方法300可以在诸如i等于1的处理器102或201a等环境中操作,使得(N+i)级缓存包括(N+1)级缓存,并且处理器还包括被配置为(N+1)级缓存的后备存储库的(N+2)级缓存。在这些环境中,检查(N+1)级缓存来确定针对存储器位置的数据先前是否已代表第二处理单元被日志记录(即,动作302)可以包括:确定(N+1)级缓存中没有缓存行与存储器位置相对应。此外,检查(N+2)级缓存来确定针对存储器位置的数据先前是否已代表第二处理单元被日志记录。

[0069] 如所示,基于动作302的结果,方法包括动作303:当数据已被日志记录时,通过参考来日志记录流入量;或者动作304:当数据尚未被日志记录时,按值日志记录流入量。

[0070] 在一些实施例中,动作303包括:当针对存储器位置的数据先前已代表第二处理单元被日志记录时,通过参考先前已代表第二处理单元被日志记录的日志数据,使针对存储器位置的数据代表第一处理单元被日志记录。继续上面的示例,例如,如果检查(N+1)级缓存L2-A1和/或检查(N+2)级缓存L3-A引起以下确定:数据/存储器位置已代表处理单元A2被日志记录(基于对缓存L1-A2的流入量),则处理器201a可以通过参考针对处理单元A2所创建的日志条目,使对缓存L1-A1的流入量代表处理单元A1被日志记录。稍后将给出如何完成通过参考进行日志记录的示例。

[0071] 转向动作302的备选结果,在一些实施例中,动作304包括:当针对存储器位置的数据尚未代表第二处理单元被日志记录时,使针对存储器位置的数据代表第一处理单元按值被日志记录。例如,如果检查(N+1)级缓存L2-A1和/或检查(N+2)级缓存L3-A引起以下确定:数据/存储器位置尚未代表另一处理单元被日志记录,则处理器201a可以使对缓存L1-A1的流入量代表处理单元A1按值被日志记录。按值日志记录可以包括例如日志记录用于处理单元A1的数据分组中的存储器地址和存储器值。请注意,按值日志记录可以包括任何数量的压缩技术来减少完成实际日志记录所需的比特数。

[0072] 如结合图1所描述的,(多个)处理器102可以包括可以用于临时存储跟踪数据的(多个)缓冲器102e。因此,在方法300中,“引起”不同类型的数据被日志记录可以包括处理器102将这样的数据存储到(多个)缓冲器102e中。附加地或备选地,其可以包括处理器102将这样的数据传送到跟踪器104a、将这样的数据写入到跟踪数据存储库104e和/或通知跟踪器104a数据在(多个)缓冲器102d中可用。在一些实施例中,(多个)缓冲器102d可以包括(多个)缓存102b的一个或多个保留部分。因此,使用缓冲器102e,在动作304/304中,使针对存储器位置的数据代表第一处理单元(通过参考或按值)被日志记录可以包括:基于诸如处理器循环、存储器位置、总线带宽等的资源的可用性来延迟日志记录。在(多个)缓冲器102d包括(多个)缓存102b的一个或多个保留部分的实施例中,所延迟的日志记录可以包括使得缓存行无效(在N级缓存和/或(N+i)级缓存中),而不是逐出它,以便出于延迟的日志记录的目的而保留针对存储器位置的数据。

[0073] 方法300的描述已经涉及具有关于下层缓存的“知识”的上层缓存。上层缓存保留的关于下级缓存的“知识”的特定形式可能会有所不同,示例如下。

[0074] 在基本形式中,该“知识”可以仅仅是上级缓存中存在与(多个)下级缓存中的(多个)缓存行相对应的缓存行(即,对应于相同的存储器位置和存储器数据的缓存行)。如上文所提及,在包含性缓存中,(多个)上层将数据的超集存储在它们下面的(多个)层中。例如,假设图2A中的缓存是包含性的。在这种情况下,当处理单元A2的活动引起来自系统存储器202的位置被导入到缓存L1-A2中时,该相同的存储器位置也被缓存在缓存L2-A1和L3-A中。如果处理单元A2的活动正被追踪,则实施例可以使存储器位置及其值代表处理单元A2被日志记录。稍后,如果处理单元A1的活动引起来自系统存储器202的相同位置被导入到缓存L1-A1中,并且该位置仍存储相同的数据,则该位置从缓存L2-A1被提供,因为缓存L2-A1已具有该数据。现有技术可以基于对缓存L2-A1的流入量而再次将该数据日志记录以用于处理单元A1。然而,本文的实施例可以替代地认识到存储器位置及其值已存在于缓存L2-A1中,并且因此已存在于缓存L1-A2中。因为处理单元A2正在被日志记录,所以实施例可以认识到存储器位置及其值已代表处理单元A2被日志记录,并因此使处理单元A1的这一新活动

参考先前代表处理单元A2被日志记录的日志数据而被日志记录。

[0075] 上层缓存的更精细形式的“知识”也是可能的。例如,实施例可以利用附加的“记账”(或日志记录)比特在一个或多个缓存层中扩展缓存行,这些附加的“记账”(或日志记录)比特使得处理器102能够为实现记账比特的每个缓存行标识该缓存行是否已被日志记录(可能具有日志记录了缓存行的(多个)处理单元的身份)。为了理解这些概念,图4A图示了类似于图2B的共享缓存203的示例共享缓存400a,其中缓存行404中的每一个包括一个或多个附加记账比特401。因此,每个缓存行404包括(多个)记账比特401、常规地址比特402和值比特403。

[0076] 备选地,图4B图示了共享缓存400b的示例,共享缓存400b包括存储存储器地址402和值403的常规缓存行405,以及用于存储适用于常规缓存行405的记账比特的一个或多个保留缓存行406。(多个)保留缓存行405的比特被分配到不同的记账比特群组,记账比特各自对应于多个常规缓存行405中的一个不同的常规缓存行。

[0077] 在示例图4B的变型中,(多个)保留缓存行406可以作为一个(或多个)路(way)保留在组相联缓存的每个索引中(其稍后将进行详细讨论)。例如,在8路组相联缓存中,组(set)中的一个路可以被保留用于适用于该组中其他七个路的记账比特。由于给定组中的所有路通常被大多数处理器并行读取,因此这可以降低实现保留缓存行的复杂度,并可以加快对保留缓存行的访问。

[0078] 不论记账比特是如何被实际存储的,每个缓存行的(多个)记账比特401可以包括用作标志(即,开启或关断)的一个或多个比特,该标志由(多个)处理器102用来指示缓存行中的当前值是否代表处理单元被日志记录(或者备选地,是否由参与日志记录的处理单元消耗)。因此,动作302中的检查可以包括使用该标志来确定缓存行是否已被参与日志记录的处理单元日志记录。

[0079] 备选地,每个缓存行的记账比特401可以包括多个比特。多个比特可以以用若干方式来使用。使用在本文中被称为“单位比特”的一个方法,每个缓存行的记账比特401可以包括与处理器102的处理单元102a的数量相等的单位比特数量(例如,如果处理器102支持超线程,则是逻辑处理单元的数量,或者如果超线程不被支持,则是物理处理单元的数量)。这些单位比特可以由处理器102用来跟踪哪个或哪些特定处理单元已日志记录了缓存行(如果存在)。因此,例如,由两个处理单元102a共享的缓存可以将两个单位比特与每个缓存行相关联。

[0080] 在使用多个记账比特401(本文中被称为“索引比特”)的另一方法中,每个缓存行的记账比特401可以包括多个索引比特,该多个索引比特足以表示对参与日志记录的计算机系统101的处理器102的每个处理单元102a的索引以及可能的“保留”值(例如,-1)。例如,如果处理器102包括128个处理单元102a,则这些处理单元可以按照每个缓存行由使用仅七个索引比特的索引值(例如,0-127)来标识。在一些实施例中,一个索引值被保留(例如,“无效”),以指示没有处理器已经日志记录缓存行。因此,这将意味着七个索引比特实际上将能够表示127个处理单元102a,再加上保留值。例如,二进制值0000000-1111110可以对应于索引位置0-126(十进制),并且二进制值1111111(例如,根据解释,为十进制的-1或127)可以对应于“无效”,以指示没有处理器已经记录了对应缓存行,但是这种表示方式(notation)可以根据实现而变化。因此,单位比特可以由处理器102用来指示缓存行是否已被日志记录

(例如,不同于-1的值),并且作为到日志记录了缓存行的特定处理单元(例如,最近消耗缓存行的处理单元)的索引。使用多个记账比特401的该第二方法的优点是在缓存102b中以很少的开销支持大量的处理单元,缺点是比第一方法的粒度小(即,一次仅一个处理单元被标识)。

[0081] 鉴于前述内容,将理解,在动作302中,检查(N+i)级缓存来确定针对存储器位置的数据先前是否已代表第二处理单元被日志记录包括:确定(N+i)级缓存中与存储器位置相对应的缓存行是否设置了一个或多个记账比特。

[0082] 可以用于确定缓存行是否已被日志记录的另一机制是利用组相联缓存和路锁定(way-locking)。由于处理器的缓存102b通常比系统存储器103小得多(通常为几个数量级),因此系统存储器103中的存储器位置通常比缓存102b的任何给定层中的行多得多。这样,一些处理器限定了用于将系统存储器的多个存储器位置映射到一个或多个缓存层的(多个)行的机制。处理器通常采用两种通用技术之一:直接映射和关联(或组相联)映射。使用直接映射,系统存储器103中的不同存储器位置被映射到缓存层中的仅一个行,使得每个存储器位置只能被缓存到该层中的特定行中。

[0083] 另一方面,使用组相联映射,系统存储器103中的不同位置可以被缓存到缓存层中的多个行之一。图5图示了系统存储器和缓存之间的组相联映射的示例500。在此,缓存层502的缓存行504在逻辑上被划分为两个缓存行的不同组,包括两个第一缓存行504a和504b的第一组(标识为索引0),以及两个缓存行504c和504d的第二组(标识为索引1)。组中的每个缓存行被标识为不同的“路”,使得缓存行504a由索引0、路0标识,缓存行504b由索引0、路1标识,依此类推。如进一步描绘的,存储器位置503a、503c、503e和503g(存储器索引0、2、4和6)被映射到索引0。这样,系统存储器中的这些位置中的每一个可以被缓存到该组内索引0处的任何缓存行(即,缓存行504a和504b)。所描绘的映射的特定模式仅出于图示和概念目的,并且不应被解释为存储器索引可以被映射到缓存行的唯一方式。

[0084] 组相联缓存通常被称为N路组相联缓存,其中N是每个组中的“路”的数量。因此,图5的缓存500将被称为2路组相联缓存。处理器通常实现N路缓存,其中N是2的幂(例如,2、4、8等),其中N的值通常被选择为4和8(但是本文的实施例不限于任何特定的N值或N值的子集)。值得注意的是,1路组相联缓存通常等价于直接映射的缓存,因为每个组仅包含一个缓存行。附加地,如果N等于缓存中的行数,则其被称为完全关联缓存,因为它包括包含缓存中所有行的单个组。在完全关联缓存中,任何存储器位置可以被缓存到缓存中的任何行。

[0085] 注意,图5表示系统存储器和缓存的简化视图,以便图示一般原理。例如,尽管图5将单独存储器位置映射到缓存行,但是应当理解,缓存中的每一行可以在系统存储器中存储与多个可寻址位置有关的数据。因此,在图5中,系统存储器(501)中的每个位置(503a-503h)实际上可以表示多个可寻址的存储器位置。附加地,映射可以在系统存储器501中的实际物理地址与缓存502中的行之间,或者可以使用虚拟地址的中间层。

[0086] 组相联缓存可以用于确定缓存行是否已通过使用路锁定而被日志记录。出于某种目的,路锁定会在缓存中锁定或保留一个或多个路。特别地,本文的实施例利用路锁定来为正被追踪的处理单元保留一个或多个路,使得锁定/保留的路被专用于存储与该单元的执行有关的缓存未命中。因此,再次参考图5,如果“路0”针对所跟踪的处理单元被锁定,则缓存行504a和504c(即,索引0、路0和索引1、路0)将被专用于与该单元的执行相关的缓存未命

中,并且其余的缓存行将被用于所有其他缓存未命中。因此,为了确定特定缓存行是否已被日志记录,处理器102仅需要确定“N+1”缓存层中存储的缓存行是否是已被保留用于所跟踪的处理单元的路的一部分。

[0087] 鉴于前述内容,将理解,在动作302中,检查(N+i)级缓存来确定针对存储器位置的数据先前是否已代表第二处理单元被日志记录包括:确定(N+i)级缓存中与存储器位置相对应的缓存行是否被存储在与所日志记录的处理单元相对应的路中。

[0088] 如先前所解释的,缓存根据CCP来操作,CCP定义了当处理单元从缓存数据读取和写入缓存数据时一致性如何被维持在各种缓存之间,以及如何确保处理单元始终从缓存中的给定位置读取有效数据。这样,结合操作缓存,处理器102维持并存储CCP状态数据。不同处理器和/或不同CCP利用其跟踪缓存一致性状态并且使得该缓存一致性数据可用于跟踪器104a的粒度可以变化。例如,一方面,一些处理器/CCP跟踪每个缓存行以及每个处理单元的缓存一致性。因此,这些处理器/CCP可以跟踪与每个处理单元相关的每个缓存行的状态。这意味着因为它与每个处理单元102a有关,所以单个缓存行可以具有关于其状态的信息。其他处理器/CCP的粒度较小,并且仅跟踪缓存行级别的缓存一致性(并且缺少每个处理单元的信息)。另一方面,因为一次只有一个处理器可以独占行(独占的、修改的等),所以处理器制造方可以仅出于效率考虑而选择在缓存行级别跟踪缓存一致性。作为中间粒度的示例,处理器/CCP可以跟踪每个缓存行的缓存一致性,以及具有当前缓存行状态的处理单元的索引(例如,四处理单元处理器的索引为0、1、2、3)。

[0089] 不管在给定处理器处CCP状态数据以何种粒度被维持,该CCP状态数据均可以被包括在(N+i)级缓存所具有的关于缓存的数据的“知识”中。特别地,与(N+i)级缓存中的给定缓存行相关联的CCP状态数据可以用于确定该缓存行是否已被处理单元之一日志记录。例如,如果CCP状态数据指示特定处理单元已将给定缓存行视为“共享的”,则该数据又可以用于确定处理单元已日志记录了来自缓存行的读取。因此,将理解,在动作302中,检查(N+i)级缓存来确定针对存储器位置的数据先前是否已代表第二处理单元被日志记录可以包括:确定(N+i)级缓存中与存储器位置相对应的缓存行是否具有可用于确定缓存行已被日志记录的相关联的CCP状态数据。

[0090] 在动作303中,数据流入量可以通过参考先前被日志记录的数据(通常由引起当前流入量的不同处理单元日志记录的数据)而被日志记录。通过参考进行日志记录可以使用多种方法(包括其组合)中的一个或多个来完成,下面将介绍其中的一些方法。

[0091] 第一方法通过参考先前日志记录的存储器地址来进行日志记录。例如,假设图2A中的处理单元A2具有表示特定存储器地址(即,在系统存储器202中)的所记录的数据和在该存储器地址处存储的特定数据。稍后,如果该特定存储器地址/特定数据是针对处理单元A1的流入量,则处理单元A1可以存储日志条目,该日志条目标识(i)特定存储器地址和(ii)处理单元A2。在此,处理单元A1避免了重新日志记录存储在存储器地址处的实际数据(其可能具有相当大的大小)。该第一方法的一些变体还可以存储排序数据,诸如,来自跨处理单元A1和A2的数据流递增的一系列的MIN。该MIN可以稍后用于对处理单元A1针对处理单元A2处的一个或多个事件(例如,还与来自同一系列的MIN相关联的那些事件)的流入量进行排序。相应地,在动作303中,通过参考先前已代表第二处理单元被日志记录的日志数据,使针对存储器位置的数据代表第一处理单元被日志记录可以包括以下中的一项或多项:日志记

录存储器位置的地址、或日志记录存储器位置的地址和排序数据(诸如,MIN)。

[0092] 第二方法通过参考存储数据的缓存行的先前所有者进行日志记录。例如,假设图2A中的处理单元A2已经日志记录了数据的第一流入量。还假设第一流入量使数据被缓存在(N+i)级缓存(例如,缓存L2-A1)的缓存行中,其中处理单元A2被标识为缓存行的所有者。稍后,如果处理单元A1引起相同数据的第二流入量,则处理单元A1可能成为(N+i)级缓存中该缓存行的所有者。处理单元A1然后可以存储标识缓存行的先前所有者(即,处理单元A2)的日志条目,使得A2的日志条目可以稍后被用来获得数据。这意味着通过参考进行的日志记录可以涉及记录缓存行的身份以及缓存行的先前所有者(例如,可能避免记录存储器地址和存储器值)。因此,在动作303中,通过参考先前已代表第二处理单元被日志记录的日志数据,使针对存储器位置的数据代表第一处理单元被日志记录可以包括:将第二处理单元日志记录为与存储器位置相对应的缓存行的先前所有者。

[0093] 第三方法通过参考CCP数据来进行日志记录。例如,如上文所提及,CCP可以存储关于不同处理单元用于读取和写入的每个缓存行的缓存一致性状态。该数据的粒度可以根据处理器的实现而变化,但是例如可以跟踪与每个处理单元相关的每个缓存行的缓存一致性状态,跟踪每个缓存行的缓存一致性状态以及拥有当前缓存行状态的处理单元的索引(例如,0、1、2、3等)等。第三方法利用可用的CCP数据来跟踪哪个(哪些)处理单元先前拥有缓存行的缓存一致性状态、哪个缓存一致性状态然后可以被用来标识哪个(哪些)处理单元已日志记录了缓存行的值。这意味着通过参考进行日志记录可以涉及记录用于缓存行的CCP数据(例如,再次可能避免记录存储器地址和存储器值)。因此,在动作303中,通过参考先前已代表第二处理单元被日志记录的日志数据,使针对存储器位置的数据代表第一处理单元被日志记录可以包括日志记录参考第二处理单元的CCP数据。

[0094] 第四方法通过参考缓存路来进行日志记录。如所提及,组相联缓存可以用于确定缓存行是否已通过使用路锁定而被日志记录。例如,假定路锁定用于为处理单元P2保留一个或多个路,并且P2日志记录数据的第一流入量。第一流入量还引起(N+i)级缓存(例如,缓存L2-A1)将第一流入量的数据存储在与该路相关联的缓存行中。当另一处理单元(例如,P1)具有相同数据的第二流入量时,在(N+i)级缓存中该缓存行的存在指示P2已日志记录了该数据。实施例可以基于注释存储缓存行的路来日志记录对P2的日志数据的参考,并且可以再次潜在地避免日志记录存储器地址和存储器值。该实施例还可以与记录排序信息(例如,MIN)结合使用,以对P1和P2之间的事件进行排序。因此,在动作303中,通过参考先前已代表第二处理单元被日志记录的日志数据,使针对存储器位置的数据代表第一处理单元被日志记录可以包括以下中的一项或多项:日志记录对缓存路的参考、或日志记录对缓存路的参考和排序数据。

[0095] 除了基于第二处理单元的先前流入量来日志记录第一处理单元的流入量之外,实施例还包括当存在相同数据的多个流入量时,用于通过单个处理单元来减少(甚至消除)日志记录的优化。例如,参考图2A,处理单元A1可以在存储器位置处的特定数据的N级缓存(例如,L1-A1缓存)中引起缓存未命中。作为响应,缓存层级结构可以将该数据导入L1-A1缓存中,并且潜在地还导入(N+i)级缓存(例如,L2-A1缓存和/或L3-A缓存)中。另外,针对处理单元A1的流入量可以按值被日志记录。稍后,该数据可以从L1-A1缓存中被逐出。在典型的缓存环境中,这可能引起数据也被主动从L2-A1缓存和/或L3-A缓存中被逐出。然而,不是在

L2-A1和/或L3-A缓存中引起(多个)逐出,而是各实施例可以在这些(N+i)级缓存中的一个或多个中保留适当的(多个)缓存行。因此,方法300可以包括在与存储器位置相对应的第一N级缓存中逐出第一缓存行,同时在与存储器位置相对应的(N+i)级缓存中保留第二缓存行。

[0096] 稍后,如果处理单元A1引起L1-A1缓存中针对同一数据的后续缓存未命中,则(N+i)级缓存(例如,L2-A1和/或L3-A缓存)中的(多个)保留缓存行可以用于确定该数据已代表处理单元A1被日志记录。因此,在一些实施例中,该后续缓存未命中由处理单元A1参考先前的日志条目来日志记录。在其他实施例中,日志条目可以针对该后续缓存未命中而被完全省略,因为处理单元A1已在其跟踪中具有了该数据。因此,方法300可以包括:基于检测到对第一N级缓存的后续流入量,后续流入量还包括存储器位置处存储的数据,基于第二缓存行的存在,使后续流入量通过参考被日志记录。附加地或备选地,方法300可以包括(i)基于在第一处理单元处执行的附加代码来检测对第一N级缓存的后续流入量,后续流入量还包括存储器位置处存储的数据,以及(ii)至少基于检测到对第一N级缓存的后续流入量,并且至少基于第二缓存行的存在,确定后续流入量不需要被日志记录。

[0097] 将理解,基于处理器检查一个或多个上层缓存层而在下层缓存层处进行日志记录的第一实施例可以被实现为实现图3的方法300的处理器控制逻辑(例如,电路和/或微代码)。这样,实现该实施例的处理器102可以包括处理器控制逻辑,如方法300所述,处理器控制逻辑检测对下层(例如,L1)缓存的流入量,然后(潜在地逐步)检查一个或多个上层缓存来确定流入量是否可以通过参考来日志记录,或者甚至流入量是否根本不需要被日志记录。

[0098] 基于下层缓存层将(多个)日志记录请求发送到(多个)上层缓存层,在下层缓存层处记录缓存未命中

[0099] 在第二实施例中,处理器基于第一处理单元的活动(诸如,从特定存储器地址的读取)来检测对下层处理器缓存(例如,L1)的流入量(即,缓存未命中),并且该下层处理器缓存然后请求上层缓存日志记录流入量和/或请求上层缓存通知下层如何日志记录流入量。该上层缓存然后确定流入量是否需要被日志记录以及如何(例如,按值或通过参考)日志记录流入量,和/或在请求缺乏必要知识的情况下将请求传递给另一上层缓存(如果存在)来确定如何日志记录流入量。这可以继续到N个缓存级。

[0100] 实现该第二实施例的处理器102可以通过在所有(多个)上层缓存处或至少参与日志记录过程的所有(多个)上层缓存处实现公共的(或至少非常相似的)控制逻辑来潜在地这样做。在一些实现中,实现第二实施例所需的控制逻辑可能不如实现第一实施例所需的控制逻辑扩展,但是基于利用(多个)上层缓存级的知识,提供了源自在下层缓存级处日志记录流入量的许多(或全部)相同优点。附加地,由于缓存级已在大多数处理器中彼此之间传递了CCP消息,因此实现第二实施例所需的控制逻辑可以潜在地被实现为对现有控制逻辑的扩展。

[0101] 根据该第二实施例,图6图示了上层缓存层基于下层缓存层的日志记录请求来确定如何通过下层缓存层日志记录流入量的示例方法600的流程图。类似于方法300,方法600可以在微处理器环境(诸如,图2A的示例环境)被中实现,该微处理器环境描绘了处理器201a,处理器201a包括多个处理单元(例如,处理单元A1-A4中的两个或更多个处理单元),

并且其包括被布置到多个缓存层中的多个缓存。这些缓存可以包括第一缓存层内的多个第一缓存(例如,缓存L1-A1至L1-A4中的两个或更多个缓存)以及第二缓存层内的一个或多个第二缓存(例如,缓存L2-A1或L2-A2或缓存L3A中的一个或多个缓存)。这些缓存可以包括第二缓存层中的特定第二缓存(例如,L2-A1或L3-A),该特定第二缓存用作至少第一缓存层中的特定第一缓存(例如,L1-A1)的后备存储库。为了简单起见,方法600将特定第一缓存称为“第一缓存”,并将特定第二缓存称为“第二缓存”。微处理器环境可以包括用于执行方法的控制逻辑(例如,微代码102c和/或电路)。在一些实施例中,这样的控制逻辑在一个或多个上层缓存层(例如,图2A中的缓存层L2和/或L3)处被实现。

[0102] 方法600在上面介绍的第二缓存处被执行,第二缓存参与日志记录,并从动作601开始,在动作601中,第二缓存接收来自内部缓存层的日志记录请求。在一些实施例中,动作601可以包括第二缓存从第一缓存接收引用特定存储器地址的日志记录请求。例如,L2缓存层中的缓存L2-A1(如果方法600在L3缓存层处被执行,则为缓存L3-A)可以从L1缓存层中的缓存L1-A1接收日志记录请求。该日志记录请求可以基于处理单元A1的活动(诸如,对特定存储器地址(例如,在系统存储器202中)的读取),该活动引起对第一缓存L1-A1的数据流入量。在图2A的环境中,该流入量中的数据可以从缓存L2-A2、缓存L3-A或系统存储器202被提供。

[0103] 基于该请求,方法600前进至动作602,在动作602中,第二缓存确定在该缓存层处是否存在针对存储器地址的缓存行。在一些实施例中,动作602可以包括基于请求,确定在第二缓存中是否存在与存储器地址相对应的缓存行。例如,基于接收到请求,L2缓存层中的缓存L2-A1(如果方法600在L3缓存层处被执行,则为缓存L3-A)可以确定其是否包含从日志记录请求缓存特定存储器地址的缓存行。尽管在缓存层级结构包括包含性缓存(例如,其中第二缓存在其下的第一缓存层中的(多个)缓存中存储数据的超集)的情况下,通常将存在这样的缓存行,但是将理解,如果缓存层级结构是独占的或表现出一些独占行为,则可能不是这种情况。

[0104] 在动作602的“否”分支之后(即,当特定第二缓存中不存在缓存行时),方法600到达动作603,在动作603中,第二缓存可以确定它是否是最外层日志记录缓存层。如将要讨论的,基于动作603的结果,方法600可以包括(i)当不存在参与日志记录并充当至少第二缓存的后备存储库的第三缓存(例如,在第三缓存层内)时,第二缓存使缓存行被日志记录(即,遵循动作608的路径),或者(ii)当第三缓存确实存在时,第二缓存将请求转发到第三缓存(即,遵循动作606的路径)。

[0105] 例如,如果第二缓存是缓存L2-A1,则在动作603处,第二缓存可以确定缓存L3-A是否存在并且是否参与日志记录(因此L2-A1不是最外日志记录缓存层)。如果确实存在缓存L3-A,则将理解,在一些实现中,根据处理器的当前配置,该缓存可以在某一时刻参与日志记录,而在另一时刻不参与日志记录。在另一示例中,如果第二缓存是缓存L3-A,则在动作603处,第二缓存可以确定不存在外部缓存层,因此第二缓存是最外日志记录缓存层。请注意,在日志记录缓存层之间可以存在中间的非日志记录缓存层。例如,如果动作603由缓存L2-A1执行,并且某个L4缓存层将存在,则L3缓存层可以是非日志记录的,而L4缓存层可以是日志记录的。

[0106] 如果来自动作603的判定是第二缓存不是外部日志记录缓存层(即,来自动作603

的“否”分支),则方法600进行到动作606,在动作606中,第二缓存将日志记录请求转发到下一日志记录缓存层。然后,方法600在该层的缓存处被重复。例如,如果第二缓存是缓存L2-A1,则它可以将请求转发到缓存L3-A,并且缓存L3-A可以重复方法600。这可以被扩展到存在的许多日志记录缓存级。在一些实现中,当到达动作606时,第二缓存层可以将一个或多个答复消息发送到第一缓存,以指令其将其日志记录请求直接发送到下一日志记录缓存层,而不是将日志记录请求转发到下一日志记录缓存层。

[0107] 另一方面,如果来自动作603的判定是第二缓存是最外日志记录缓存层(即,来自动作603的“是”分支),则方法600进行到动作608,在动作608中,第二缓存使流入量被日志记录。如稍后将讨论的,动作608处的日志记录可以按值或通过参考(根据特定情况)来执行,并且动作608处的实际日志记录可以在当前缓存层和/或下层缓存层处被执行。

[0108] 注意,如动作603的判定框中的虚线所示,根据执行方法600的计算环境,动作603可以是可选动作。例如,如果缓存层级结构包括参与日志记录(并执行方法600)的仅一个上层缓存层,则该缓存层将始终是“最外层”的日志记录缓存层。在这些环境中,动作603可能不是必需的。此外,即使存在多个日志记录缓存层,最外层的日志记录缓存层也可以具有其是最外层的固有知识。在任一种情况下,动作602处的“否”判定因此可以简单地进行至动作608。

[0109] 返回到动作602,并且在“是”分支(即,当第二缓存中存在缓存行时)之后,方法600到达动作604,在动作604中,第二缓存确定缓存行是否被日志记录。该确定可以包括确定缓存行是否由第二缓存日志记录,或者缓存行是否由某个其他缓存日志记录,并且第二缓存是否知晓该日志记录。第二缓存确定缓存行已被日志记录(并且潜在地通过(多个)处理单元)的方式可以依赖于结合第一缓存日志记录实施例(例如,包括例如结合图4A、图4B和图5描述的实施例)所描述的任何机制。例如,第二缓存可以如结合图4A和图4B所描述的那样存储记账比特(即,标志比特、单位比特和/或索引比特),第二缓存可以利用如结合图5所描述的路锁定,和/或第二缓存可以存储并依赖于CCP数据。

[0110] 如将要讨论的,如果缓存行不由第二缓存确定为被日志记录,则方法600可以包括第二缓存将请求转发到下一日志记录缓存层(即,遵循动作606的路径)和/或日志记录缓存行(即,遵循动作608的路径)。另一方面,如果缓存行由第二缓存确定为被日志记录,则方法600可以包括:当第二缓存尚未确定第一缓存是否知道该缓存的缓存行中所存储的当前值时,第二缓存使缓存行被日志记录(即,遵循操作608的路径),或者当确定请求处理器知道第二缓存的缓存行中所存储的当前值时,第二缓存确定缓存行不需要被日志记录(即,遵循动作609的路径)。

[0111] 例如,如果来自动作604的判定是缓存行未由第二缓存确定未被日志记录时(即,来自动作604的“否”分支),则在动作608处,第二缓存可以使流入量被日志记录,同时潜在地通知外部日志记录缓存层(如果存在)所发生的日志记录(即,动作607)。如果动作607被执行,则在动作608处在缓存行未由特定第二缓存确定为被日志记录的情况下,使缓存行被日志记录可以包括:确定第三缓存层存在,并通知第三缓存该缓存行已经由特定第二缓存按值日志记录。请注意,在方法600中,动作607和608可以相对于彼此以任何顺序执行(包括并行执行的动作)。注意,动作607可以使得方法600在下一日志记录缓存层处被执行。

[0112] 备选地,图6示出了,如果来自动作604的判定是缓存行未由第二缓存确定为被日

志记录,则第二缓存在动作603处可以确定它是否是最外层的日志记录缓存层,并基于结果,在操作606处将请求转发到下一日志记录缓存层或者在操作608处日志记录流入量。本质上,这些替换路径传达的信息是,如果在操作604处存在“否”判定,则第二缓存可以(i)日志记录流入量并通知下一日志记录缓存层(如果存在)(即,使得下一层知道日志记录事件以供以后使用),和/或(ii)将请求转发到下一日志记录缓存层,因为第二缓存(甚至更高的层)可能包含可能引起流入量通过参考被日志记录或根本不被日志记录的知识。

[0113] 返回到动作604,如果第二缓存知道缓存行正在被日志记录(即,来自动作604的“是”分支),则在动作605处,第二缓存确定引起日志记录请求的处理单元是否在缓存行中具有当前值。应理解,第二缓存对于所请求的存储器地址可以具有比第一缓存当前拥有的最新的值。例如,如果第一缓存是缓存L1-A1且第二缓存是缓存L2-A1,则可能是缓存L2-A2针对特定存储器地址的所具有的值比动作601中,处理单元A1执行引起日志记录请求的读取时(例如,由于处理单元A2的活动)的L1-A1所具有的值更加新。如果可以肯定地知道第一缓存具有第二缓存处的缓存行中的当前值(即,动作605的“是”分支),则第二缓存可以选择在动作609处不日志记录任何内容(即,因为当前值已被日志记录)。另一方面,如果不能肯定地知道第一缓存具有第二缓存处的缓存行中的当前值(即,动作605的“否”分支),则第二缓存可以在动作608处使流入量被日志记录,同时潜在地通知外部日志记录缓存层(如果存在)所发生的日志记录(即,动作607)。再次,动作607和608可以相对于彼此以任何顺序被执行(包括并行执行的动作)。

[0114] 如所提及的,动作608处的流入量的日志记录可以根据具体情况按值或通过参考来执行。通常,如果流入量的值不能基于跟踪来定位,则流入量按值来日志记录(例如,通过重播期间的处理器活动或先前日志记录的缓存行)。如果流入量的值可以通过重播已日志记录的处理器活动来获得,或者如果流入量的值被存储在先前已日志记录的缓存行中,则流入量可以能够通过参考来日志记录。值得注意的是,即使在通过参考日志记录流入量可能合法的情况下,按值日志记录流入量仍是合法的。可以做出按值进行日志记录的判定,例如以节省跟踪期间的处理时间、创建更容易被重播的跟踪等。因此,将理解,动作608可以包括基于直接日志记录特定存储器地址的值和/或日志记录对特定存储器地址的先前日志条目的参考,使缓存行被日志记录。

[0115] 在方法600中日志记录可以能够通过参考来执行的一种情况是,动作608是否已经由动作605被达到(即,第二缓存知道值已被日志记录,但是请求处理器无法明确地知道当前值)。在此,动作608处的日志记录可以参考已知由第二缓存日志记录的值而针对第一缓存被执行。例如,缓存L1-A2可能已日志记录了当前值,并且缓存L2-A1知道此事,因此流入量可以参考L1-A2的日志记录而针对缓存L1-A1被日志记录。

[0116] 在这些情况下,如果重播代码可以通过其他方式恢复该值,则不日志记录任何内容也是有意义的。例如,由处理单元A1引起的当前流入量可能已与处理单元A2的先前活动有关地被日志记录,因此在动作608处有可能通过参考A2的日志来日志记录当前流入量。然而,如果此时方法600拒绝为A1日志记录任何内容,则跟踪可能仍然是正确的。这减少了跟踪的大小,并且权衡需要在重放期间在其他处理单元的跟踪中定位该先前日志记录的值。

[0117] 在重播期间定位先前日志记录的值的任务取决于能够在不同的日志记录的处理单元之间再现事件的至少部分顺序。一些内容可以被包括在跟踪中,以帮助对先前日志记

录的值进行定位。例如,日志记录缓存逐出有助于在重播时确定满足A1的读取所需的值在缓存A1-L1中不可用(即,因为它已被逐出)。因此,该值然后可以在(多个)跟踪中被搜索以用于(多个)其他处理单元。在另一示例中,日志记录CCP数据还可以帮助确定在重播期间满足A1的读取所需的值不可用或当前不在缓存A1-L1中。因此,该值然后可以在(多个)跟踪中被搜索以用于(多个)其他处理单元。请注意,CCP数据可以潜在地指示在哪里寻找当前值。在另一示例中,缓存的几何形状知识可以帮助对所需的日志条目进行定位。例如,可能已知处理单元A1和A2共享相同的L2缓存(即,L2-A1)。因此,例如与在跟踪中搜索A3和A4相反,首先搜索A2的跟踪来得到所需的日志条目是有意义的。

[0118] 日志记录可以能够通过参考来执行的另一情况是,当基于下层缓存层在动作607处向当前缓存层发送通知,方法600在当前缓存层处被执行时,是否已达到动作608。此处,下层缓存层将以及日志记录了流入量(按值或通过参考),因此当前缓存层可以参考下层缓存层的日志来进行日志记录。

[0119] 如还提到的,动作608处的日志记录可以在当前缓存层或下层缓存层处被执行。例如,在一些实现中,不是第二缓存层在到达动作608时自行执行日志记录,而是可以将一个或多个答复消息向下发送回第一缓存,以指令第一缓存流入量应被日志记录以及如何日志记录流入量(即,按值或通过参考,如果通过参考,则参考日志条目在何处)。(多个)答复消息还可以指示第一缓存如何设置记账比特、保存CCP数据等。类似地,在动作609处,第二缓存层可以向第一缓存层发送答复消息,以通知其没有必要进行日志记录。如果原始日志记录请求已通过多于一个日志记录缓存层传播,则这些答复消息可以向下通过各层被往回传播,或者可以被直接发送给原始请求者。鉴于前述内容,将理解,动作608可以包括基于指令第一缓存直接记录特定存储器地址的值,或者指令第一缓存记录对特定存储器地址的先前日志条目的参考来引起缓存行被日志记录。

[0120] 不论日志记录如何被执行,动作608可以包括第二缓存层设置任何适当的日志记录记帐比特(例如,标志比特、单位比特或索引比特),或者保存任何适当的CCP消息,以便文档记录流入量被日志记录的事实。这样,将理解,在动作608中,使缓存行被日志记录可以包括将缓存行标记为被日志记录在特定第二缓存内(例如,通过适当地设置与缓存行相关联的记账比特)。

[0121] 在一些实施例中,在动作608处的日志记录可以包括第二缓存主动通知一个或多个下层缓存层其已日志记录了缓存行,以及可能的其如何日志记录缓存行。例如,如果缓存L2-A1已在动作608处进行了日志记录,则它可以向缓存L1-A2至L1-A4中的一个或多个发送一个或多个消息(即,除了发起日志记录请求的缓存之外的L1缓存)来通知他们其已日志记录了缓存行。该信息可以包括缓存L2-A1是否已按值或通过参考日志记录了缓存行。如果缓存行通过参考日志记录,则缓存L2-A1甚至可以发送有关原始日志数据存在位置的信息。作为响应,缓存(例如,L1-A2至L1-A4中的一个或多个)可以存储记录缓存L2-A1已日志记录了缓存行(潜在地包括缓存L2-A1如何已日志记录了缓存行)的事实的信息。该信息可以例如被存储在这些L1缓存内的附加记账比特中。以这种方式,如果这些L1缓存中的一个缓存稍后确定它们需要日志记录缓存行,则它们可以提前知道它已被日志记录,并且避免将日志记录请求发送到上层缓存或将如何日志记录的问题发送到上层缓存。

[0122] 值得注意的是,以上结合第一实施例讨论的用于使用(多个)缓冲器102(e)和/或

(多个)缓存102b的专用部分来完成延迟日志记录的任何技术也适用于第二实施例。这样,将理解,在动作608中,使缓存行被日志记录可以包括在诸如(多个)缓冲器102e和/或(多个)缓存102b的一部分的跟踪缓冲器内日志记录缓存行。

[0123] 虽然方法600集中于在上层日志记录缓存层处执行动作,但是图7-图9示出了可以在下层缓存层(例如,发起了原始日志记录请求的(多个)L1缓存层)处执行的一些示例方法。特别地,尽管方法600集中于执行日志记录(包括设置缓存行的日志记录状态来指示缓存行的值已被日志记录(例如,设置与缓存行相关联的记账比特、存储CCP数据等)),但是这些方法与以后该缓存行的值不再被日志记录时清除该日志记录状态有关。

[0124] 缓存可以包含设置了其日志记录状态的缓存行,这是因为使用该缓存的处理单元在为该处理单元启用日志记录的同时执行了存储器读取。缓存还可以从上级缓存接收缓存行,并且该缓存行上已设置了日志记录状态。如上文所提及,由于较高级缓存已主动通知它缓存行已被日志记录,因此缓存行也可能设置了日志记录状态。通常,当处理单元对缓存行执行写入操作而禁用该处理单元的日志记录时,缓存行的日志记录状态通常会被清除。

[0125] 首先,图7图示了当处理单元在启用日志记录到禁用日志记录之间转换时,用于管理缓存行的日志记录状态的示例方法700的流程图。与方法600一样,方法700可以在诸如图2A的示例环境的微处理器环境中被实现。通常,方法700在处理单元(例如,A1)已在启用日志记录的情况下进行操作之后操作,并且其使用现在包括已被日志记录的一个或多个缓存行的缓存(例如,L1-A1)。如果处理单元写入这些已日志记录的缓存行之一,则方法700根据当前针对处理单元启用还是禁用日志记录来保留或清除该日志记录状态。

[0126] 方法700在动作701处开始,在动作701中,它检测到对标记为已日志记录的缓存行的写入。在一些实施例中,动作701可以包括检测对具有已设置的日志记录状态的第一缓存中的缓存行的写入。例如,第一缓存可以是缓存L1-A1。该缓存可以具有基于处理单元A1的存储器读取而先前被标记为已日志记录(例如,通过适当地设置其记账比特)的缓存行。例如,该缓存行可以对应于上面结合方法600讨论的特定存储器地址。

[0127] 接下来,方法700包括动作702,在动作702中,确定是否启用日志记录。在该上下文中,动作702确定与第一缓存相关联的处理单元是否启用了日志记录。在一些实施例中,动作702可以包括基于检测到写入而确定是否为特定处理单元启用了日志记录。例如,针对缓存L1-A1的控制逻辑可以确定处理单元A1是否已启用日志记录。如果启用了日志记录(即,来自动作702的“是”分支),则针对缓存行的日志记录状态可以在动作703处被保留。因此,在一些实施例中,动作703可以包括至少基于针对特定处理单元启用日志记录而保留针对缓存行的日志记录状态。

[0128] 备选地,如果日志记录被禁用(即,动作702的“否”分支),则针对缓存行的日志记录状态可以在动作704处被清除。因此,在一些实施例中,该动作可以包括至少基于针对特定处理单元禁用日志记录,清除针对缓存行的日志记录状态。例如,缓存L1-A1可以根据需要清除缓存行的记帐比特。

[0129] 如图所示,除了清除日志记录状态之外,方法700还包括通知下一日志记录缓存层。在一些实施例中,动作705可以包括至少基于针对特定处理单元禁用日志记录,通知一个或多个第二缓存中的至少一个针对第二缓存行的其日志记录状态应被清除。例如,第二缓存之一可以是缓存L2-A1,并且这样缓存L1-A1可以通知缓存L2-A1清除其缓存行副本的

日志记录状态。注意,动作704和705可以相对于彼此以任何顺序执行(包括并行执行)。

[0130] 尽管未在图6中描绘,但是方法600可以对应地包括特定第二缓存从第一缓存接收消息,消息指示第一缓存中还与存储器地址相对应的另一缓存行被标记为未被日志记录在第一缓存内。方法600还可以包括基于消息而将缓存行标记为未被日志记录在特定第二缓存内。

[0131] 图8图示了当禁用了日志记录的处理单元仅从亲代缓存接收用于写入的缓存行时,用于管理缓存行的日志记录状态的示例方法800的流程图。与方法600和700一样,方法800可以在微处理器环境(诸如,图2A的示例环境)中被实现。通常,方法800在处理单元(例如,A1)在启用日志记录的情况下操作时进行操作,并且其使用(多个)亲代缓存(例如,L2-A1和/或L3-A)包含已设置日志记录状态的缓存行的缓存(例如,L1-A1),并且缓存采用缓存行来从亲代缓存进行写入。

[0132] 方法800从动作801处开始,在动作801中,在禁用日志记录的情况下,缓存从上层缓存请求缓存行以进行写入。例如,基于来自处理单元A1写入特定存储器地址的请求,可能在缓存L1-A1中发生缓存未命中。结果,缓存L1-A1可以从缓存L2-A1或缓存L3-A请求适当的缓存行的副本。

[0133] 在一些情况下,缓存L1-A1可以接收清除了日志记录状态的缓存行。这样,方法800可以包括动作802,在动作802中,缓存仅从上层缓存接收清除了日志记录状态的缓存行。日志记录状态可以在所接收的缓存行中被清除,例如是因为(i)日志记录状态未在上层缓存中被设置,或者(ii)上层缓存知道在处理单元A1处禁用了日志记录,因此在将缓存行提供给缓存L1-A1时,上层缓存清除了日志记录状态。例如,方法600可以包括:从第一缓存接收请求缓存行进行写入的消息,并将缓存行发送到第一缓存,至少基于针对第一缓存禁用的日志记录,缓存行被标记为未被日志记录。

[0134] 在其他情况下,缓存L1-A1可以接收设置了日志记录状态的缓存行。这样,方法800可以包括动作803,在动作803中,缓存仅从上层缓存接收设置了日志记录状态的缓存行。例如,因为日志记录状态在上层缓存中被设置,因此日志记录状态可以在所接收的缓存行中被设置。

[0135] 接下来,方法800可以包括动作804,在动作804中,在仍然禁用日志记录的情况下,缓存执行对缓存行的写入。例如,缓存L1-A1可以通过将适当的值写入到缓存行来完成来自处理单元A1的原始写入请求。接下来,方法800可以包括动作805和动作806,在动作805中,针对缓存行的日志记录状态被清除,在动作806中,上层缓存被通知清除其针对缓存行的日志记录状态。尽管动作805与动作804被分开描述,但应注意,清除针对缓存行的日志记录状态可以是在动作804处执行写入操作的自然部分。例如,在禁用日志记录的情况下,任何写入操作均可能引起日志记录状态针对所写入的缓存行被清除。这样,动作804和805之间的箭头以虚线示出,以指示动作804实际上可以是可选的。动作806可以按照与以上结合方法700讨论的动作705类似的方式操作。

[0136] 类似于方法700的动作705,当动作806被执行时,方法600可以对应地包括特定第二缓存从第一缓存接收消息,消息指示第一缓存中还与存储器地址相对应的另一缓存行被标记为未被日志记录在第一缓存内。方法600还可以包括基于消息,将缓存行标记为未被日志记录在特定第二缓存内。

[0137] 图9图示了当处理单元对处理单元已占用“拥有的”CCP状态的缓存行进行写入时，用于管理缓存行的日志记录状态的示例方法900的流程图。与方法600-800一样，方法900可以在诸如图2A的示例环境的微处理器环境中被实现。通常，方法900在CCP提供状态的情况下操作，在该状态下，在一个处理单元已占用了缓存行进行写入的时段期间，(多个)其他处理单元可以请求该缓存行的当前值。该方面的一个示例是较早介绍的MOESI CCP中的“拥有的”状态。

[0138] 方法900从动作901开始，在动作901中，在禁用日志记录的情况下，缓存修改处于拥有的状态的缓存行。例如，处理单元A1可能已将缓存行视为缓存L1-A1中“拥有的”状态。在此时间期间，处理单元A1可以执行对该缓存行的写入。如结合图8所讨论的，当禁用日志记录时，清除针对缓存行的日志记录状态可以是执行写入操作的自然部分。这样，方法900没有描绘用于清除日志记录状态的任何快速动作，但是在一些实现中可以存在快速动作。

[0139] 基于动作901，方法900示出了一个(或多个)动作可以被采取来传达缓存行的日志状态也应在(多个)其他缓存上被清除。在动作902中，基于请求，缓存通知同级缓存来清除针对缓存行的日志记录状态。例如，在对缓存L1-A1中拥有的缓存行的写入已被执行之后，缓存L1-A1可以从诸如缓存L2-A2的同级缓存接收请求(例如，CCP消息)，以请求缓存行的当前值。作为该请求的结果，如果日志记录状态已被设置，则缓存L1-A1可以通知缓存L2-A2应清除其对应缓存行中的日志记录状态。该通知可以与传达缓存L1-A1中的缓存行的当前值的CCP消息一起被发送，也可以作为分离的消息的一部分被发送。

[0140] 在动作903中，基于修改缓存行，缓存通知一个或多个同级缓存来清除针对缓存行的日志记录状态。例如，在对缓存L1-A1中所拥有的缓存行执行写入操作之后，缓存L1-A1可以向其同级缓存(例如，L1-A2至L1A4)广播通知，以使他们知道应清除针对该缓存行的日志记录状态(如果这些缓存中存在缓存行并且设置了日志记录状态)。因此，尽管动作902相对地通知同级缓存来清除日志记录状态，但是动作903主动地通知同级缓存。

[0141] 在动作904中，基于修改缓存行，缓存通知上层缓存层来清除针对缓存行的日志记录状态。例如，在对缓存L1-A1中所拥有的缓存行执行写入操作之后，缓存L1-A1可以向其(多个)亲代缓存(诸如，缓存L2-A1和/或L3-A)广播通知，使得他们知道应清除针对该缓存行的日志记录状态(如果这些缓存中存在该缓存行并且设置了日志记录状态)。因此，类似于动作903，动作904执行主动通知，但是这次是针对(多个)亲代缓存，而不是(多个)同级缓存。

[0142] 值得注意的是，一些实现可以执行动作902-904中的多于一个的动作。例如，一个实现可以在执行写入时主动通知(多个)上层缓存(即，动作904)，但是仅反应性地通知(多个)同级缓存(即，动作902)。在另一示例中，一个实现可以主动通知(多个)上层缓存(即，动作904)和(多个)同级缓存(即，动作903)。

[0143] 附加地，类似于方法700的动作705和方法800的动作806，当执行903时，方法600可以对应地包括特定第二缓存从第一缓存接收消息，消息指示第一缓存中与存储器地址相对应的另一缓存行被标记为未被日志记录在第一缓存内。方法600还可以包括基于消息，将缓存行标记为未被日志记录在特定第二缓存内。

[0144] 基于对(多个)上层缓存(例如，L2、L3等)的了解，将流入量记录到下层缓存(例如，L1)可以提供超越实现通过参考来进行日志记录和在一些情况下避免日志记录的若干优

点。例如,只有当来自缓存未命中的数据实际上已被处理单元消耗时,下层才启动日志记录过程。例如,这可以避免记录因推测性执行而引起的缓存未命中。另外,下层可以与引起缓存活动的指令的退出同时执行日志记录。这可以引起捕获更高精度时序的跟踪。最后,在记录下层时,如果需要,记录可以基于虚拟存储器寻址、而不是物理存储器寻址。值得注意的是,如果基于虚拟存储器寻址进行日志记录,则可能存在多个虚拟地址映射到同一物理地址的情况。在这些情况下,缓存可能不会引起通过不同虚拟地址对同一物理地址的访问表现为缓存未命中。如果发生这种情况,则该跟踪器104a可以日志记录来自TLB 102f的数据。在一些实现中,虚拟或物理地址可以进一步通过附加标识符(例如,虚拟处理器ID、存储器地址的安全设置等)来区分。在这些实现中的至少一些实现中,缓存可能引起对具有不同附加标识符(或具有更高、更低或不同安全级别)的同一地址的访问表现为缓存未命中。

[0145] 因此,本文的实施例提供了用于基于使用至少两个层级或层的处理器缓存来跨多个处理单元跟踪执行效果来记录比特精确的“时间旅行”跟踪记录的不同实施例。以这种方式记录跟踪文件可能只需要进行适度的处理器修改,并且当与现有的跟踪记录方法相比时,它可以将跟踪记录的性能影响以及跟踪文件的大小减少几个数量级。

[0146] 在不脱离本发明的精神或基本特征的情况下,本发明可以以其他特定形式体现。所描述的实施例在所有方面仅应被认为是说明性的而非限制性的。因此,本发明的范围由所附权利要求书而不是前面的描述指示。落入权利要求的等同含义和范围内的所有改变均应被包含在其范围之内。

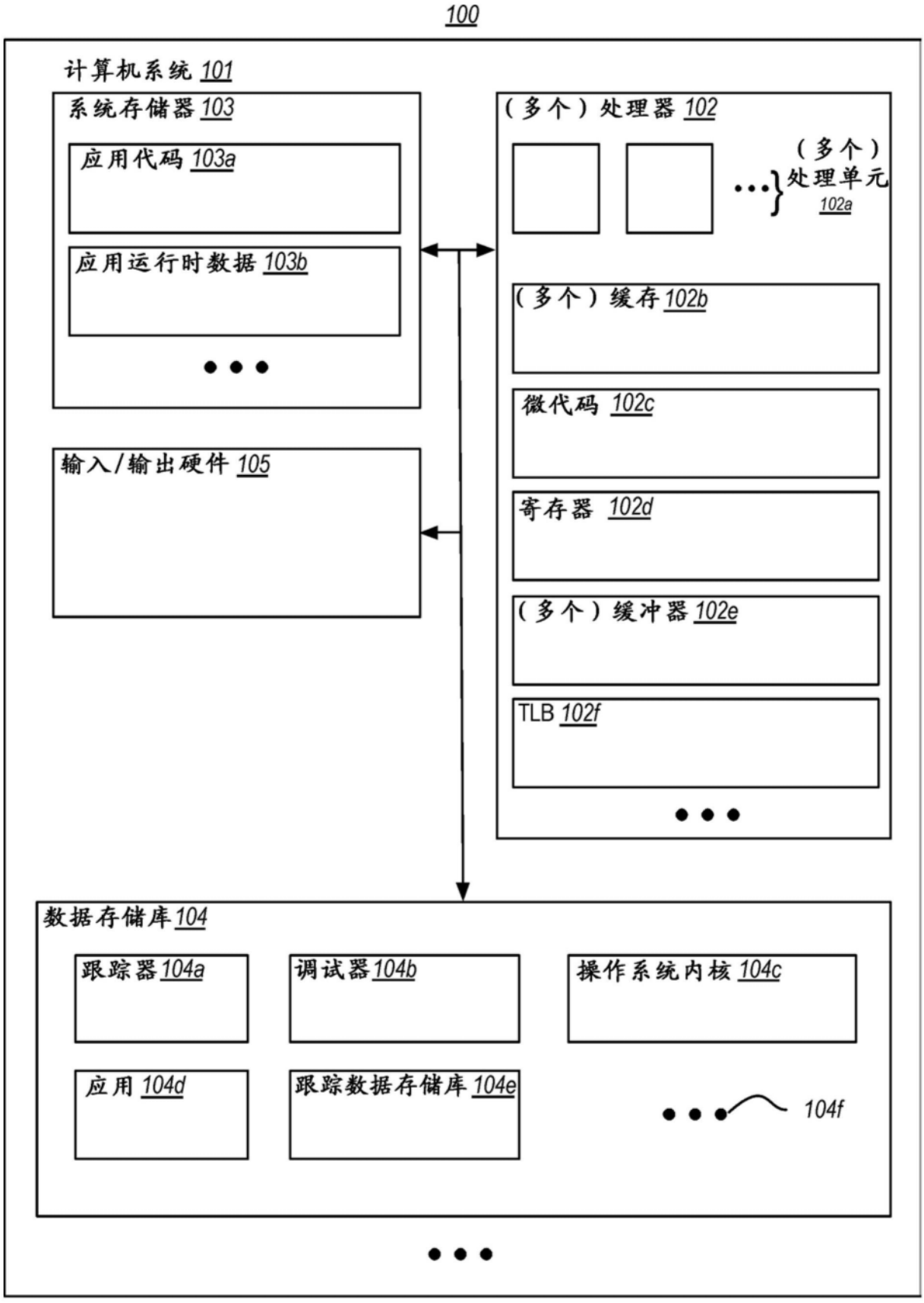

100
计算机系统 101
系统存储器 103
应用代码 103a
应用运行时数据 103b
输入/输出硬件 105
(多个)处理器 102
(多个)处理单元 102a
(多个)缓存 102b
微代码 102c
寄存器 102d
(多个)缓冲器 102e
TLB 102f
数据存储库 104
跟踪器 104a
调试器 104b
操作系统内核 104c
应用 104d
跟踪数据存储库 104e
104f

图1

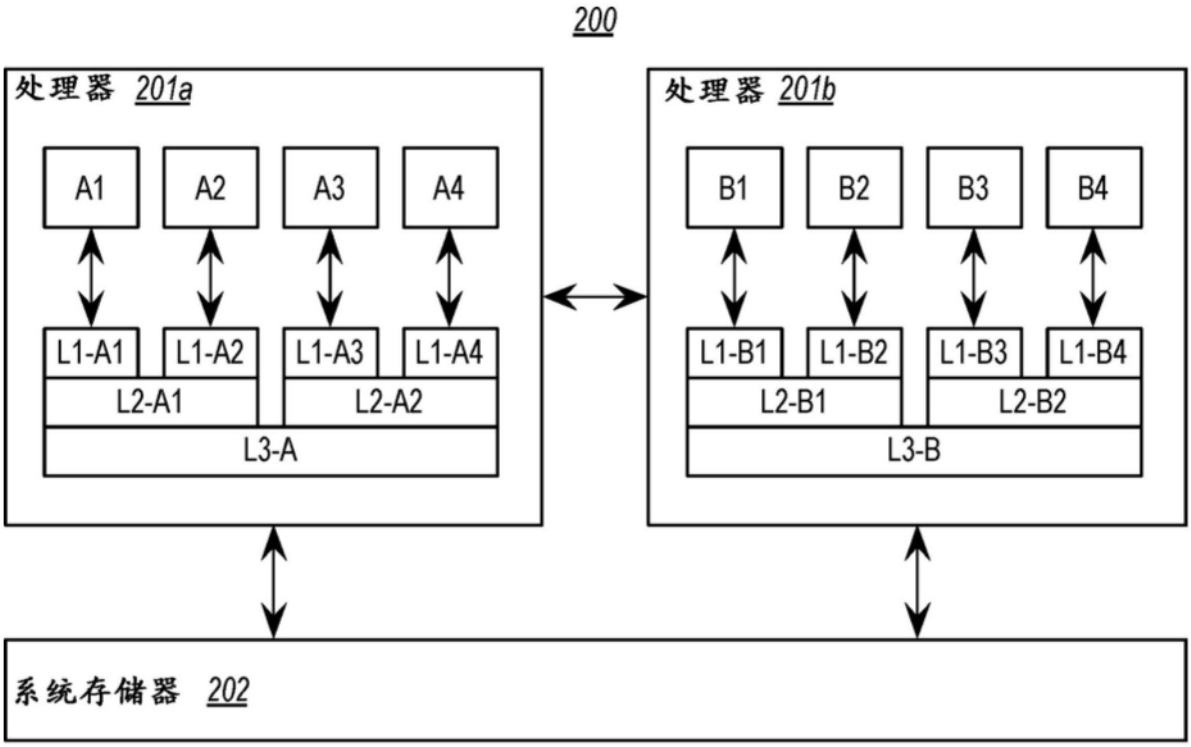

200
处理器 201a
A1
A2
A3
A4
L1-A1
L1-A2
L1-A3
L1-A4
L2-A1
L2-A2
L3-A
处理器 201b
B1
B2
B3
B4
L1-B1
L1-B2
L1-B3
L1-B4
L2-B1
L2-B2
L3-B
系统存储器 202

图2A

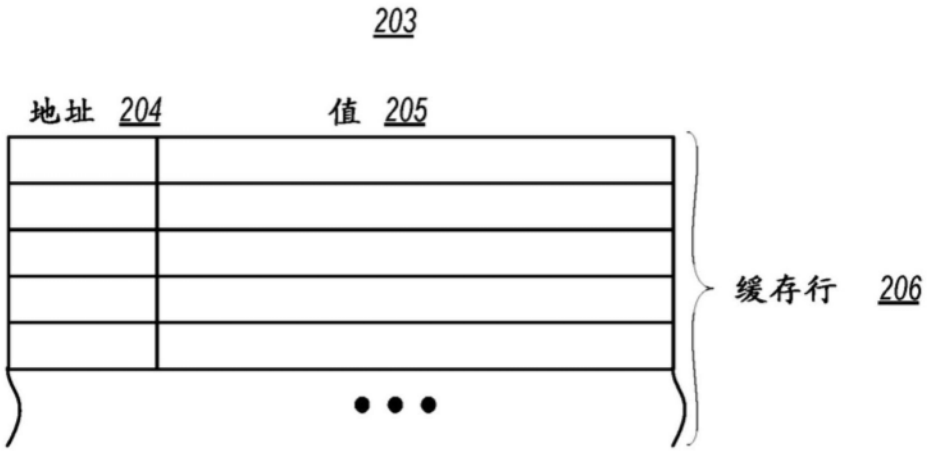

203
地址 204
值 205
缓存行 206

图2B

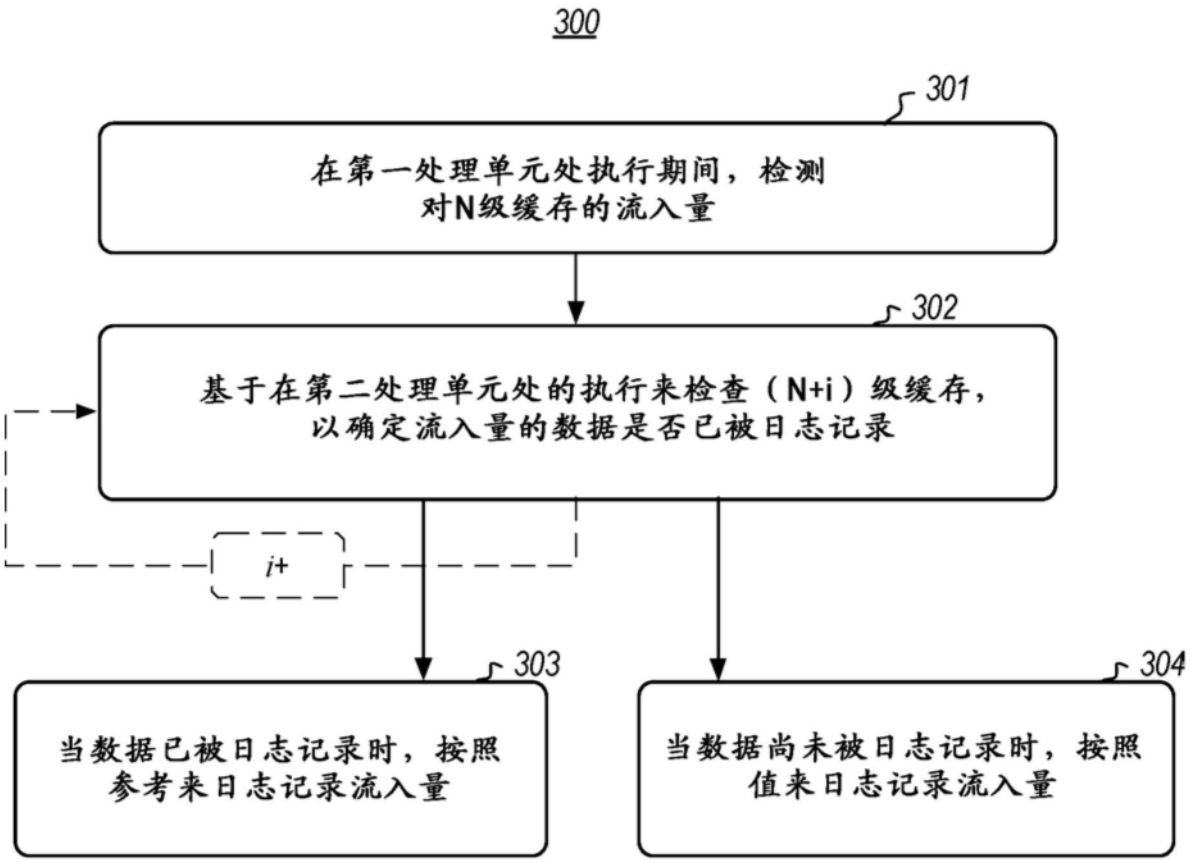

300
301
在第一处理单元处执行期间，检测
对N级缓存的流入量
302
基于在第二处理单元处的执行来检查（N+i）级缓存，
以确定流入量的数据是否已被日志记录
i+
303
当数据已被日志记录时，按照
参考来日志记录流入量
304
当数据尚未被日志记录时，按照
值来日志记录流入量

图3

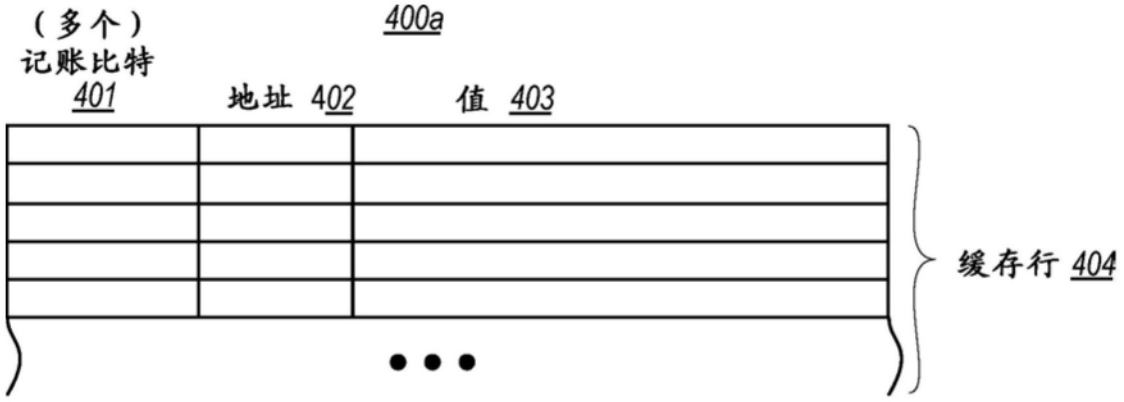

（多个）
记账比特
401
400a
地址 402
值 403
缓存行 404

图4A

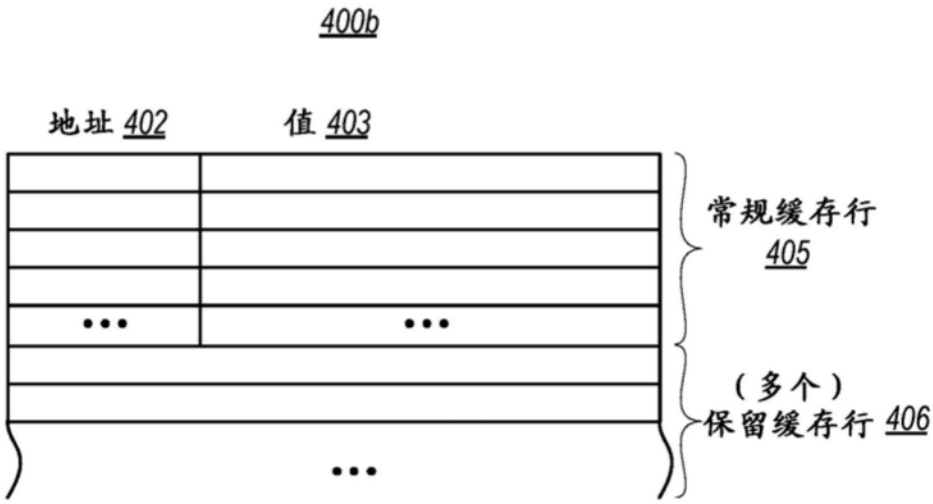

400b
地址 402
值 403
常规缓存行
405
（多个）
保留缓存行 406

图4B

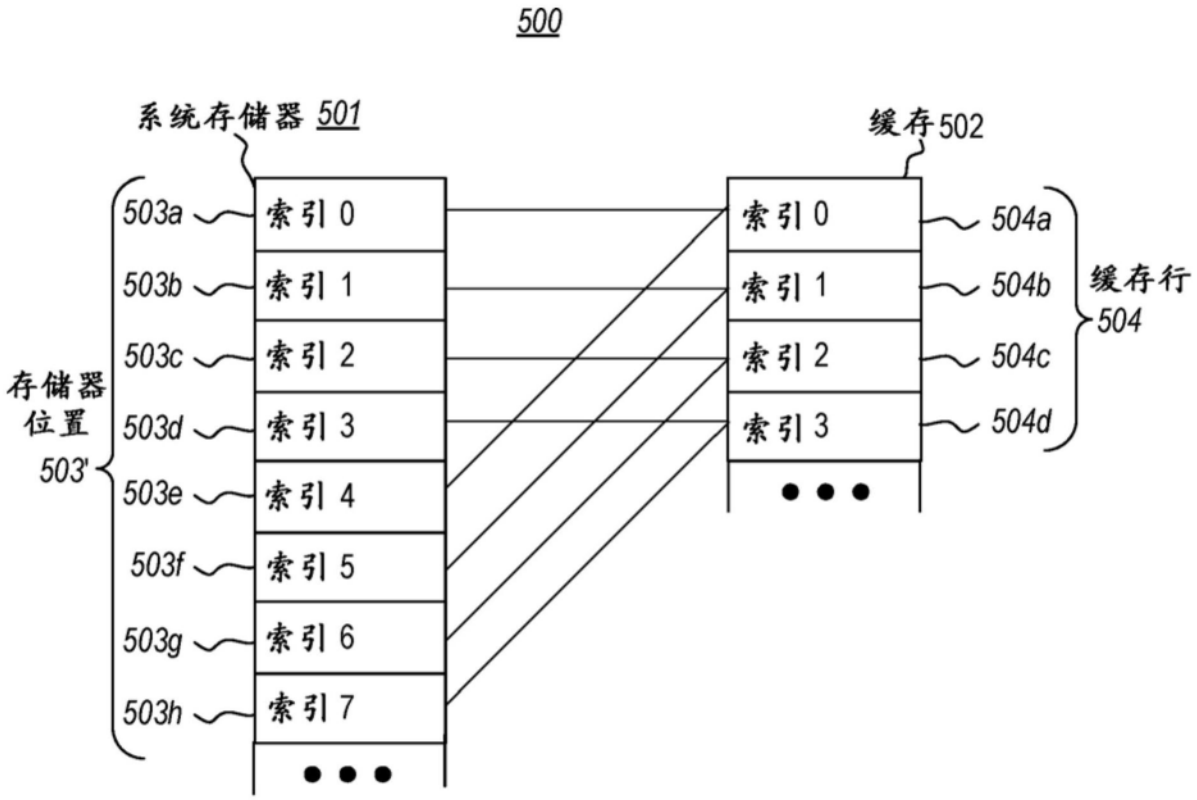

500
系统存储器 501
缓存 502
503a
503b
503c
503d
503e
503f
503g
503h
存储器
位置
503'
索引 0
索引 1
索引 2
索引 3
索引 4
索引 5
索引 6
索引 7
索引 0
索引 1
索引 2
索引 3
504a
504b
504c
504d
缓存行
504

图5

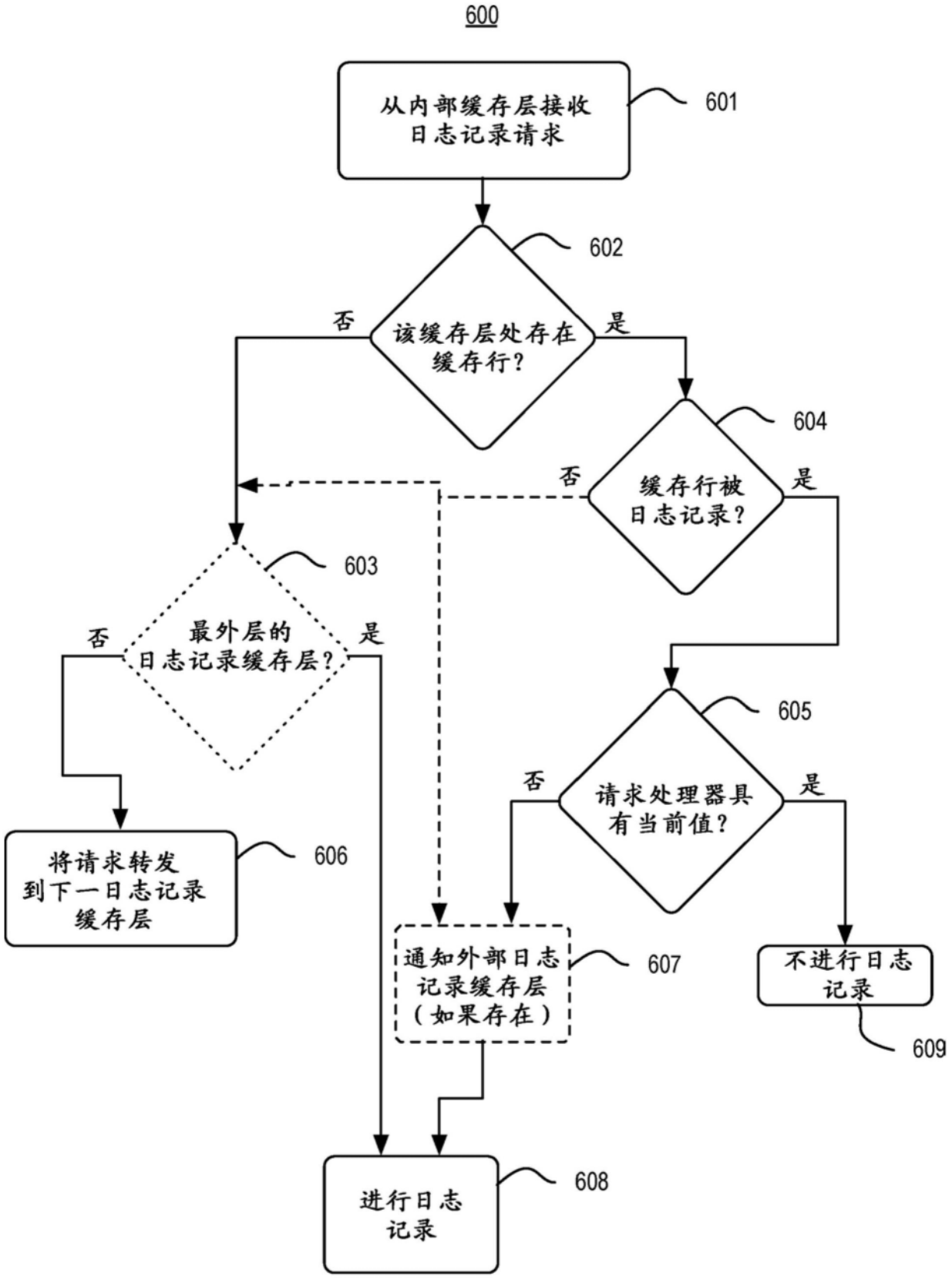

600
从内部缓存层接收日志记录请求
601
602
该缓存层处存在缓存行?
否
是
604
缓存行被日志记录?
否
是
603
最外层的日志记录缓存层?
否
是
605
请求处理器具有当前值?
否
是
将请求转发到下一日志记录缓存层
606
通知外部日志记录缓存层(如果存在)
607
不进行日志记录
609
进行日志记录
608

图6

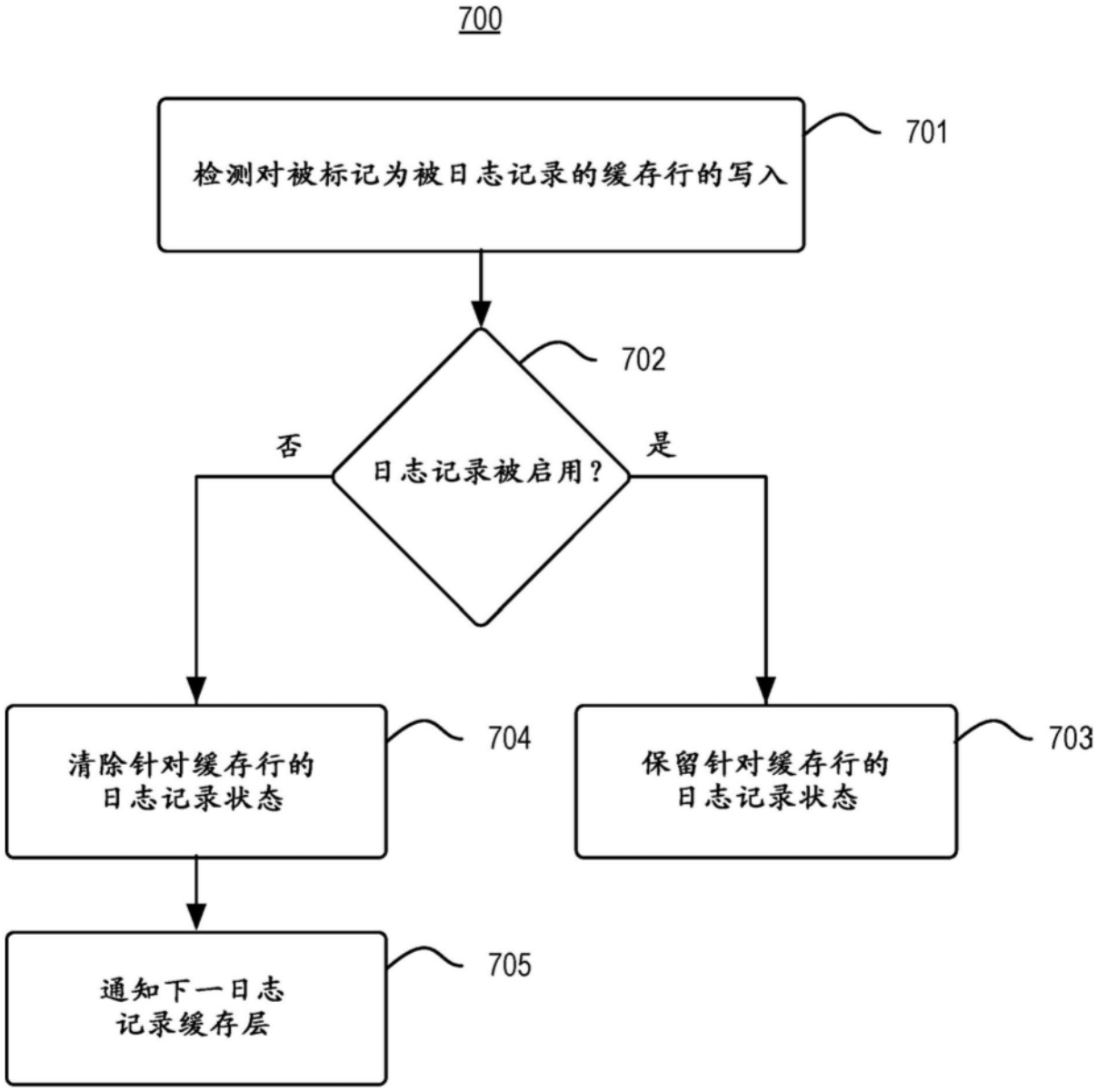

700
检测对被标记为被日志记录的缓存行的写入
701
日志记录被启用？
702
否
是
清除针对缓存行的
日志记录状态
704
保留针对缓存行的
日志记录状态
703
通知下一日志
记录缓存层
705

图7

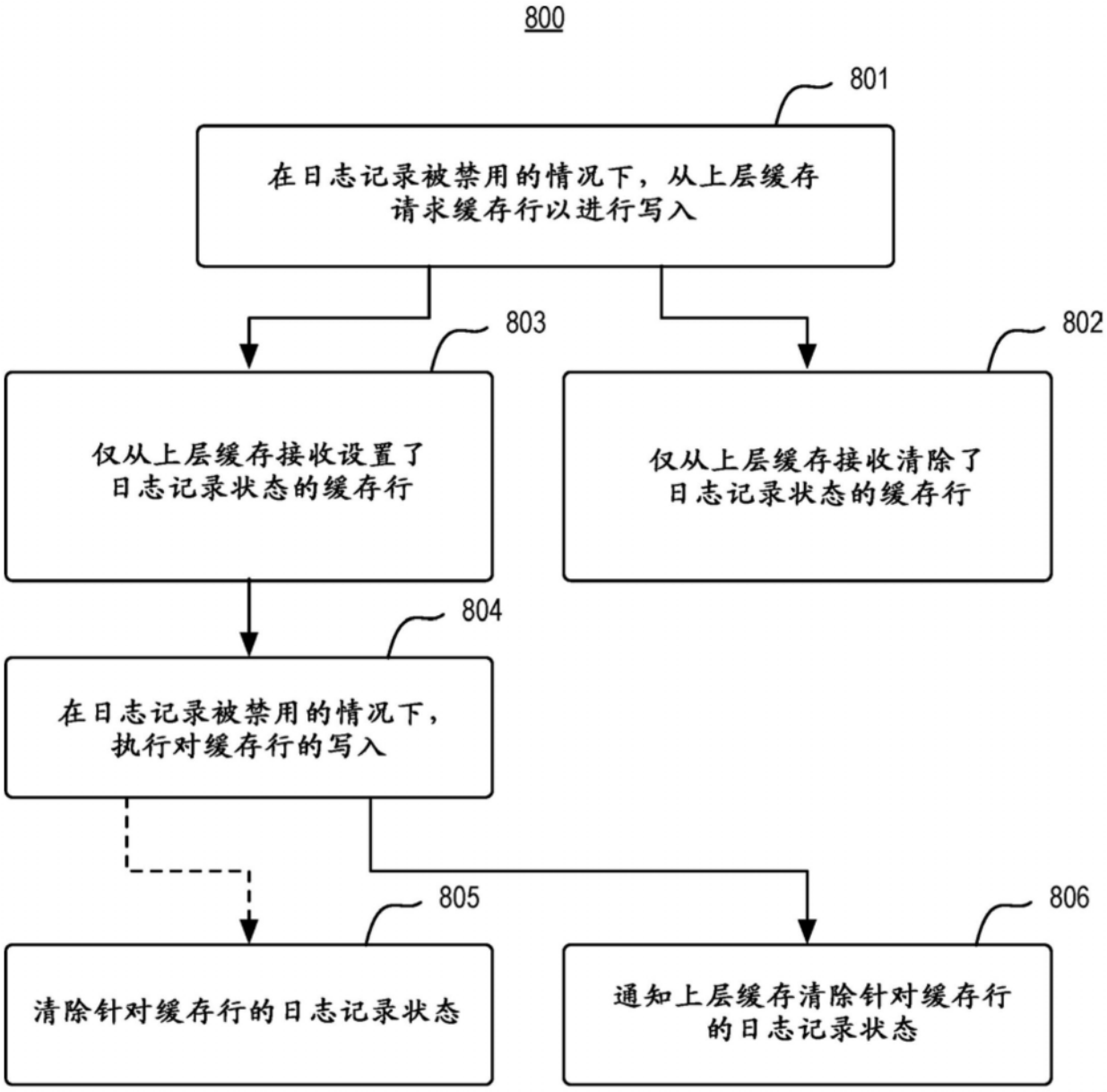

800
801
在日志记录被禁用的情况下，从上层缓存请求缓存行以进行写入
803
仅从上层缓存接收设置了日志记录状态的缓存行
802
仅从上层缓存接收清除了日志记录状态的缓存行
804
在日志记录被禁用的情况下，执行对缓存行的写入
805
清除针对缓存行的日志记录状态
806
通知上层缓存清除针对缓存行的日志记录状态

图8

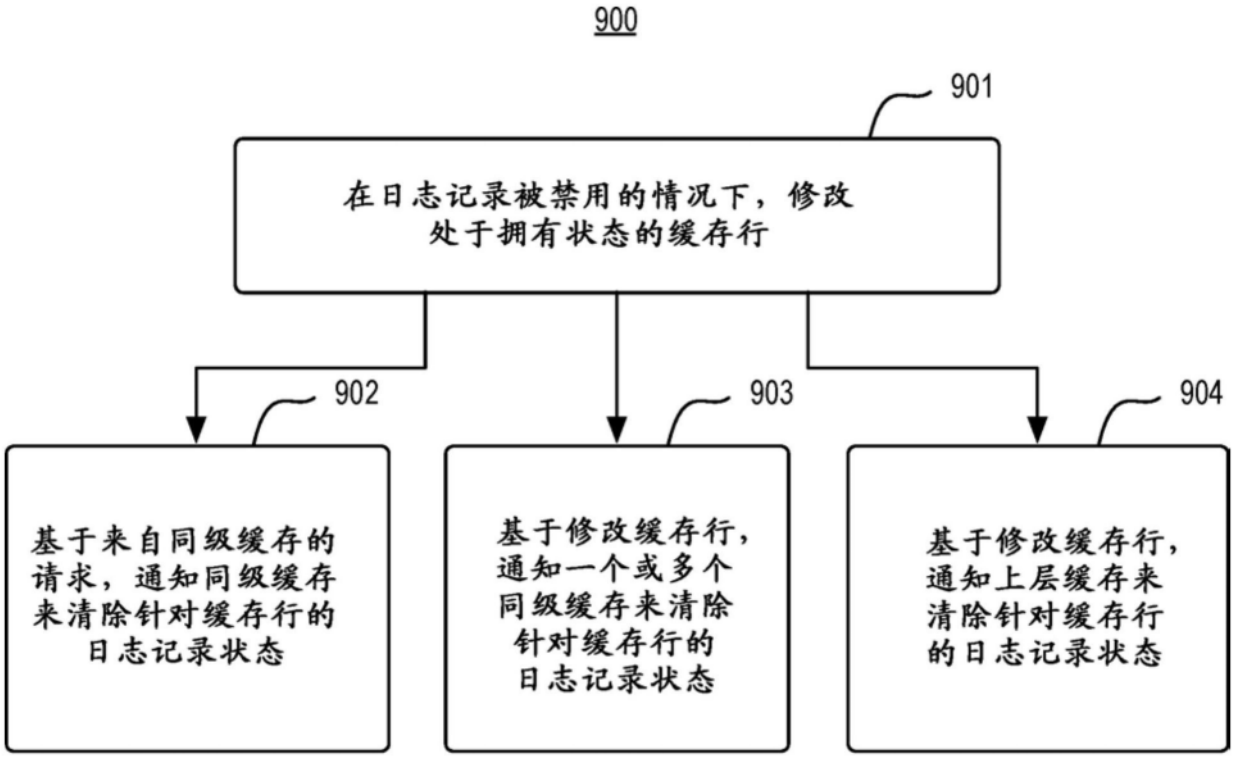

900
901
在日志记录被禁用的情况下，修改处于拥有状态的缓存行
902
基于来自同级缓存的请求，通知同级缓存来清除针对缓存行的日志记录状态
903
基于修改缓存行，通知一个或多个同级缓存来清除针对缓存行的日志记录状态
904
基于修改缓存行，通知上层缓存来清除针对缓存行的日志记录状态

图9