(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 107003894 A
(43)申请公布日 2017.08.01

(21)申请号 201580063894.1

(22)申请日 2015.11.23

(30)优先权数据
14/581,772 2014.12.23 US

(85)PCT国际申请进入国家阶段日
2017.05.24

(86)PCT国际申请的申请数据
PCT/US2015/062053 2015.11.23

(87)PCT国际申请的公布数据
WO2016/105752 EN 2016.06.30

(71)申请人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 J.E.戈奇利希 G.A.波卡姆 S.胡

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 徐红燕 张涛

(51)Int.Cl.
G06F 9/46(2006.01)

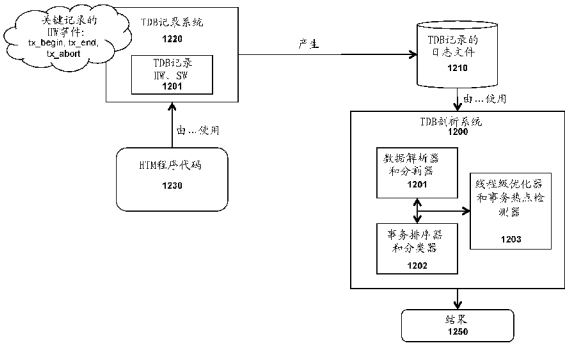
权利要求书2页 说明书16页 附图21页

(54)发明名称

用于硬件事务内存程序的剖析器的装置和方法

(57)摘要

描述了用于硬件事务内存(HTM)剖析器的装置和方法。例如,装置的一个实施例包括事务调试器(TDB)记录模块,用于记录与事务内存程序代码的执行相关的数据,该数据包括与事务内存程序代码中的事务事件和分支的执行相关的数据;以及剖析器,用于使用基于跟踪的重放技术分析记录的数据的部分以响应性地生成剖析数据,所述剖析数据包括可用于优化事务内存程序代码的事务级事件和函数级冲突数据。



CN 107003894 A

1.一种装置,包括:

事务调试器(TDB)记录模块,用于记录与事务内存程序代码的执行相关的数据,所述数据包括与所述事务内存程序代码中的事务事件和分支的执行相关的数据;以及

剖析器,用于使用基于跟踪的重放技术分析记录的数据的部分以响应性地生成剖析数据,所述剖析数据包括可用于优化所述事务内存程序代码的事务级事件和函数级冲突数据。

2.如权利要求1所述的装置,其中所述剖析器分析与事务事件和分支的执行相关的数据以生成具有函数级冲突数据的事务级事件。

3.如权利要求2所述的装置,其中所述剖析器还提取事务级事件并将所述事务级事件与来自所述事务内存程序代码的反汇编二进制信息互相关。

4.如权利要求3所述的装置,其中所述剖析器生成所述互相关的结果,所述结果包括识别出的在发生事务中止操作时执行的函数。

5.如权利要求4所述的装置,其中当针对事务检测到事务中止时,所述剖析器执行反向查找以识别对应的事务开始事件,并且一旦识别了所述事务开始事件,则捕获在所述事务的执行期间采取的所有可能分支的时间戳范围。

6.如权利要求5所述的装置,其中所述剖析器将来自周围时间戳范围内包含的分支的指令指针(IP)与来自所述事务内存程序代码的反汇编二进制信息进行比较以识别所述反汇编二进制信息中的对应IP。

7.如权利要求6所述的装置,其中所述剖析器附加地识别使用那些IP的一个或多个函数,由此允许用户看到应当分析哪一个或多个函数以实现性能优化。

8.如权利要求1所述的装置,还包括:

数据解析逻辑,用于提取要由所述剖析器分析的、与事务内存程序代码的执行相关的数据的部分。

9.如权利要求8所述的装置,其中所述数据解析逻辑提取由所述TDB记录模块记录的共享内存和事务事件分组信息。

10.如权利要求9所述的装置,其中所述事务事件分组信息包括指示与不同的事务事件相关联的时间的时间戳信息。

11.如权利要求10所述的装置,其中通用时间戳信息包括事务开始定时数据、事务结束定时数据和/或事务中止定时数据。

12.如权利要求9所述的装置,其中所述事务事件分组信息还包括分支跟踪分组信息,所述分支跟踪分组信息包括关于贯穿所述事务内存程序代码所采取的分支的信息。

13.一种方法,包括:

记录与事务内存程序代码的执行相关的数据,所述数据包括与所述事务内存程序代码中的事务事件和分支的执行相关的数据;以及

使用基于跟踪的重放技术分析记录的数据的部分以响应性地生成剖析数据,所述剖析数据包括可用于优化所述事务内存程序代码的事务级事件和函数级冲突数据。

14.如权利要求13所述的方法,其中分析还包括:

分析与所述事务事件和分支的执行相关的数据以生成具有函数级冲突数据的事务级事件。

15.如权利要求14所述的方法，其中所述剖析器还提取所述事务级事件并将所述事务级事件与来自所述事务内存程序代码的反汇编二进制信息互相关。

16.如权利要求15所述的方法，其中所述剖析器生成所述互相关的结果，所述结果包括识别出的在发生事务中止操作时执行的函数。

17.如权利要求16所述的方法，其中当针对事务检测到事务中止时，所述剖析器执行反向查找以识别对应的事务开始事件，并且一旦识别出所述事务开始事件，则捕获在所述事务的执行期间采取的所有可能分支的时间戳范围。

18.如权利要求17所述的方法，其中所述剖析器将来自周围时间戳范围内包含的分支的指令指针(IP)与来自所述事务内存程序代码的反汇编二进制信息进行比较以识别所述反汇编二进制信息中的对应IP。

19.如权利要求18所述的方法，其中所述剖析器附加地识别使用那些IP的一个或多个函数，由此允许用户看到应当分析哪一个或多个函数以实现性能优化。

20.如权利要求13所述的方法，还包括：

数据解析逻辑，用于提取要由所述剖析器分析的、与事务内存程序代码的执行相关的数据的部分。

21.如权利要求20所述的方法，其中所述数据解析逻辑提取由TDB记录模块记录的共享内存和事务事件分组信息。

22.如权利要求21所述的方法，其中所述事务事件分组信息包括指示与不同的事务事件相关联的时间的时间戳信息。

23.如权利要求22所述的方法，其中通用时间戳信息包括事务开始定时数据、事务结束定时数据和/或事务中止定时数据。

24.如权利要求21所述的方法，其中所述事务事件分组信息还包括分支跟踪分组信息，所述分支跟踪分组信息包括关于贯穿所述事务内存程序代码所采取的分支的信息。

用于硬件事务内存程序的剖析器的装置和方法

技术领域

[0001] 本发明的实施例总地涉及计算机系统领域。更具体地，本发明的实施例涉及用于硬件事务内存程序的剖析器(profiler)的装置和方法。

背景技术

[0002] 多核处理器和/或处理核内的多线程指令执行流水线使得软件程序员开发多线程软件程序。多线程软件自然是复杂的，因为同时执行的不同进程。然而，由于其执行方式的“不确定性”的方面，多线程软件此外还难以调试。具体地，即使程序从相同的输入状态开始，多线程软件程序也可能跨两个不同的运行时间而以不同方式执行。

[0003] 由于这些原因，“日志记录”被用于记录多线程软件程序的执行中的某些关键连接点。处理器目前设计有日志记录电路，所述日志记录电路观察处理器的软件执行并记录所述电路被设计用于检测的某些关键事件。如果软件程序崩溃，则分析日志记录以研究导致崩溃的程序执行。

[0004] 硬件事务内存(HTM)将很快被用作许多事务内存(TM)系统的基石。不幸的是，在HTM系统上执行日志记录和其它剖析技术提出了新的挑战，因为现有的仅软件剖析技术已经显示出仅对于仅软件TM(STM)系统是可接受的。因为硬件事务比软件事务执行快一个数量级或更多，所以在软件事务中引发大约10％开销的现有方法将对硬件事务引发100％或更多的开销。剖析团体通常同意，在剖析数据聚集时，超过10％的开销的任何事情都有可能引入探针效应，该效应将改变多线程程序的争用签名(contention signature)，这似乎表明这些现有技术对于HTM不可能够用。

附图说明

[0005] 图1A是图示了根据本发明的实施例的示例性有序流水线以及示例性寄存器重命名、无序发布/执行流水线二者的框图；

图1B是图示了根据本发明的实施例的在处理器中包括的有序架构核的示例性实施例和示例性寄存器重命名、无序发布/执行架构核二者的框图；

图2是根据本发明的实施例的具有集成的存储器控制器和图形的多核处理器和单核处理器的框图；

图3图示了根据本发明的一个实施例的系统的框图；

图4图示了根据本发明的实施例的第二系统的框图；

图5图示了根据本发明的实施例的第三系统的框图；

图6图示了根据本发明的实施例的片上系统(SoC)的框图；

图7图示了根据本发明的实施例的对比使用软件指令转换器将源指令集中的二进制指令转换为目标指令集中的二进制指令的框图；

图8图示了具有日志记录电路的示例性现有技术处理器；

图9图示了具有事务能力和日志记录电路的改进处理器；

图10a图示了可以由处理器执行的第一方法；

图10b图示了可以由处理器执行的第二方法；

图10c图示了可以由处理器执行的第三方法；

图10d图示了可以由处理器执行的第四方法；

图11图示了示例性分组结构；

图12图示了包括剖析系统的示例性架构；

图13图示了事务排序器和分类器模块的一个实施例；

图14图示了对于四个不同线程的线程度量的示例性集合；

图15A-C图示了源自本发明的实施例的优化结果的示例性集合；以及

图16图示了根据本发明的一个实施例的方法。

具体实施方式

[0006] 示例性处理器架构

图1A是图示了根据本发明的实施例的示例性有序取得、解码、引退流水线和示例性寄存器重命名、无序发布/执行流水线二者的框图。图1B是图示了根据本发明的实施例的在处理器中包括的有序取得、解码、引退核和示例性寄存器重命名、无序发布/执行架构核的框图。图1A-1B中的实线框图示了流水线 和核的有序部分，而虚线框的可选添加则图示了寄存器重命名、无序发布/执行流水线和核。

[0007] 在图1A中，处理器流水线100包括取得级102、长度解码级104、解码级106、分配级108、重命名级110、调度(也称为分派或发布)级112、寄存器读取/存储器读取级114、执行级116、写回/存储器写入级118、异常处理级122和提交级124。

[0008] 图1B示出了处理器核190，其包括耦合到执行引擎单元150的前端单元130，并且这二者都耦合到存储器单元170。核190可以是精简指令集计算(RISC)核、复杂指令集计算(CISC)核、超长指令字(VLIW)核或混合或替换的核类型。作为又一选项，核190可以是专用核，诸如例如网络或通信核、压缩引擎、协处理器核、通用计算图形处理单元(GPGPU)核、图形核等。

[0009] 前端单元130包括耦合到指令高速缓存单元134的分支预测单元132，指令高速缓存单元134耦合到指令转译后备缓冲器(TLB)136，指令转译后备缓冲器(TLB)136耦合到指令取得单元138，指令取得单元138耦合到解码单元140。解码单元140(或解码器)可以对指令进行解码，并且生成作为输出的一个或多个微操作、微代码入口点、微指令、其它指令或其它控制信号，其解码自、或以其它方式反映、或导出自原始指令。解码单元140可以使用各种不同的机制来实现。合适机制的示例包括但不限于查找表、硬件实现、可编程逻辑阵列(PLA)、微代码只读存储器(ROM)等。在一个实施例中，核190包括微代码ROM或存储用于某些宏指令的微代码(例如，在解码单元140中或以其它方式在前端单元130内)的其它介质。解码单元140耦合到执行引擎单元150中的重命名/分配器单元152。

[0010] 执行引擎单元150包括重命名/分配器单元152，重命名/分配器单元152耦合到引退单元154和一个或多个调度器单元156的集合。(一个或多个)调度器单元156表示任何数量的不同调度器，包括保留站、中心指令窗口等。(一个或多个)调度器单元156耦合到(一个或多个)物理寄存器文件单元158。(一个或多个)物理寄存器文件单元158中的每个表示一

个或多个物理寄存器文件，其中不同的物理寄存器文件存储一个或多个不同的数据类型，诸如标量整数、标量浮点、打包整数、打包浮点、向量整数、向量浮点、状态(例如，作为要执行的下一个指令的地址的指令指针)等。在一个实施例中，(一个或多个)物理寄存器文件单元158包括向量寄存器单元、写掩码寄存器单元和标量寄存器单元。这些寄存器单元可以提供架构向量寄存器、向量掩码寄存器和通用寄存器。(一个或多个)物理寄存器文件单元158被引退单元154重叠以图示可以在其中(例如，使用(一个或多个)重新排序缓冲器和(一个或多个)引退寄存文件；使用(一个或多个)未来的文件、(一个或多个)历史缓冲器和(一个或多个)引退寄存器文件；使用寄存器映射和寄存器池等)实现寄存器重命名和无序执行的各种方式。引退单元154和(一个或多个)物理寄存器文件单元158耦合到(一个或多个)执行集群160。(一个或多个)执行集群160包括一个或多个执行单元162的集合和一个或多个存储器访问单元164的集合。执行单元162可以并且在各种类型的数据(例如，标量浮点、打包整数、打包浮点、向量整数、向量浮点)上执行各种操作(例如，移位(shift)、加法、减法、乘法)。虽然一些实施例可以包括专用于特定功能或功能集合的数个执行单元，但是其它实施例可以仅包括一个执行单元或者全部执行全部功能的多个执行单元。(一个或多个)调度器单元156、(一个或多个)物理寄存器文件单元158和(一个或多个)执行集群160被示出为可能是复数的，因为某些实施例针对某些类型的数据/操作创建单独的流水线(例如，标量整数流水线、标量浮点/打包整数/打包浮点/向量整数/向量浮点流水线，和/或存储器访问流水线，其中的每个具有它们各自的调度器单元、(一个或多个)物理寄存器文件单元和/或执行集群--并且在单独的存储器访问流水线的情况下，某些实施例被实现为其中只有该流水线的执行集群具有(一个或多个)存储器访问单元164)。还应该理解，在使用单独的流水线的情况下，这些流水线中的一个或多个可以是无序发布/执行，并且其余的可以是有序的。

[0011] 存储器访问单元164的集合耦合到存储器单元170，存储器单元170包括数据TLB单元172，数据TLB单元172耦合到数据高速缓存单元174，数据高速缓存单元174耦合到第2级(L2)高速缓存单元176。在一个示例性实施例中，存储器访问单元164可以包括加载单元、存储地址单元和存储数据单元，其每一个单元均耦合到存储器单元170中的数据TLB单元172。指令高速缓存单元134进一步耦合到存储器单元170中的第2级(L2)高速缓存单元176。L2高速缓存单元176耦合到一个或多个其它级别的高速缓存并且最终耦合到主存储器。

[0012] 通过示例的方式，示例性寄存器重命名、无序发布/执行核架构可以实现流水线100如下：1)指令取得138执行取得和长度解码级102和104；2)解码单元140执行解码级106；3)重命名/分配器单元152执行分配级108和重命名级110；4)(一个或多个)调度器单元156执行调度级112；5)(一个或多个)物理寄存器文件单元158和存储单元170执行寄存器读取/存储读取级114；执行集群160执行执行级116；6)存储器单元170和(一个或多个)物理寄存器文件单元158执行写回/存储器写入级118；7)在异常处理级122中可能涉及各种单元；以及8)引退单元154和(一个或多个)物理寄存器文件单元158执行提交级124。

[0013] 核190可以支持一个或多个指令集(例如，x86指令集(具有已经向较新版本添加的一些扩展)；加利福尼亚州桑尼维尔的MIPS技术公司的MIPS指令集；加利福尼亚州桑尼维尔的ARM控股公司的ARM指令集(具有诸如NEON的可选附加扩展))，包括本文所述的(一个或多个)指令。在一个实施例中，核190包括支持打包数据指令集扩展(例如，AVX1、AVX2和/或一些形式的通用向量友好指令格式(U=0和/或U=1))，如下所述)的逻辑，从而允许使用打包数

据来执行许多多媒体应用所使用的操作。

[0014] 应当理解，核可以支持多线程(执行操作或线程的两个或更多并行集合)，并且可以以各种方式这样做，各种方式包括时间分片多线程、同时多线程(其中单个物理核针对物理核是同时多线程的线程中的每个提供逻辑核)或其组合(例如，诸如在英特尔®超线程技术中的此后的时间分片取得和解码以及同时多线程)。

[0015] 虽然在无序执行的上下文中描述了寄存器重命名，但是应当理解，可以在有序的架构中使用寄存器重命名。虽然处理器的图示实施例还包括单独的指令和数据高速缓存单元134/174和共享L2高速缓存单元176，但是替换实施例可以具有用于指令和数据二者的诸如例如第1级(L1)内部高速缓存的单个内部高速缓存，或多级内部高速缓存。在一些实施例中，系统可以包括内部高速缓存和在核和/或处理器外部的外部高速缓存的组合。替换地，所有高速缓存可以在核和/或处理器外部。

[0016] 图2是根据本发明的实施例的处理器200的框图，处理器200可以具有多于一个核、可以具有集成存储器控制器并且可以具有集成图形。图2中的实线框示出了具有单个核202A、系统代理210、一个或多个总线控制器单元216的集合的处理器200，而虚线框的可选添加图示了具有多个核202A-N、在系统代理单元210中的一个或多个集成存储器控制器单元214的集合和专用逻辑208的替换处理器200。

[0017] 因此，处理器200的不同实现可以包括：1)CPU，其具有是集成图形和/或科学(吞吐量)逻辑的专用逻辑208(其可以包括一个或多个核)，以及是一个或多个通用核的核202A-N(例如，通用有序核、通用无序核、二者的组合)；2)协处理器，其具有是主要用于图形和/或科学(吞吐量)的大量专用核的核202A-N；和3)协处理器，其具有是大量通用有序核的核202A-N。因此，处理器200可以是通用处理器、协处理器或专用处理器，诸如例如网络或通信处理器、压缩引擎、图形处理器、GPGPU(通用图形处理单元)、高吞吐量多集成核(MIC)协处理器(包括30个或更多个核)、嵌入式处理器等。处理器可以在一个或多个芯片上实现。处理器200可以是一个或多个衬底的一部分和/或使用诸如例如BiCMOS、CMOS或NMOS之类的多种工艺技术中的任何技术在一个或多个衬底上实现。

[0018] 存储器层级包括核内的一个或多个级别的高速缓存、集合或一个或多个共享高速缓存单元206和耦合到集成存储器控制器单元214的集合的外部存储器(未示出)。共享高速缓存单元206的集合可以包括一个或多个中级高速缓存，诸如第2级(L2)、第3级(L3)、第4级(L4)或其它级别的高速缓存、最后级别高速缓存(LLC)和/或其组合。虽然在一个实施例中，基于环的互连单元212互连集成图形逻辑208、共享高速缓存单元206的集合和系统代理单元210/(一个或多个)集成存储器控制器单元214，但是替换实施例可以使用任何数量的用于对此类单元进行互连的公知技术。在一个实施例中，在一个或多个高速缓存单元206和核202-A-N之间保持一致性(coherency)。

[0019] 在一些实施例中，一个或多个核202A-N能够进行多线程。系统代理210包括协调和操作核202A-N的那些部件。系统代理单元210可以包括例如功率控制单元(PCU)和显示单元。PCU可以是或包括调整核202A-N和集成图形逻辑208的功率状态所需的逻辑和部件。显示单元用于驱动一个或多个外部连接的显示器。

[0020] 在架构指令集方面，核202A-N可以是同构或异构的；也就是说，核202A-N中的两个或更多核可能能够执行相同的指令集，而其它核可能能够只执行该指令集的子集或者不同

指令集。在一个实施例中，核202A-N是异构的并且包括下面描述的“小”核和“大”核二者。

[0021] 图3-6是示例性计算机架构的框图。本领域已知的用于膝上型计算机、台式计算机、手持PC、个人数字助理、工程工作站、服务器、网络设备、网络集线器、交换机、嵌入式处理器、数字信号处理器(DSP)、图形设备、视频游戏设备、机顶盒、微控制器、蜂窝电话、便携式媒体播放器、手持设备和各种其它电子设备的其它系统设计和配置也是合适的。通常，能够合并如本文所公开的处理器和/或其它执行逻辑的各种各样的系统或电子设备通常是合适的。

[0022] 现在参考图3，示出的是根据本发明的一个实施例的系统300的框图。系统300可以包括耦合到控制器集线器320的一个或多个处理器310、315。在一个实施例中，控制器集线器320包括图形存储器控制器集线器(GMCH)390和输入/输出集线器(IOH)350(其可以在单独的芯片上)；GMCH 390包括存储器和图形控制器，存储器340和协处理器345耦合到GMCH 390上；IOH 350将输入/输出(I/O)设备360耦合到GMCH 390。替换地，存储器和图形控制器之一或二者集成在处理器内(如本文所述)，存储器340和协处理器345直接耦合到处理器310，并且利用IOH 350与在单个芯片中的控制器集线器320耦合。

[0023] 图3中用虚线标示附加处理器315的可选性质。每个处理器310、315可以包括本文描述的一个或多个处理核并且可以是某版本的处理器200。

[0024] 存储器340可以是例如动态随机存取存储器(DRAM)、相变存储器(PCM)或二者的组合。对于至少一个实施例，控制器集线器320经由诸如前端总线(FSB)的多分支总线、诸如快速通道互连(QPI)之类的点对点接口或类似连接395来与(一个或多个)处理器310、315通信。

[0025] 在一个实施例中，协处理器345是专用处理器，诸如例如高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器等。在一个实施例中，控制器集线器320可以包括集成图形加速器。

[0026] 在包括架构、微架构、热、功耗特性等在内的品质度量的范围方面在物理资源310、315之间可以存在各种差异。

[0027] 在一个实施例中，处理器310执行控制一般类型的数据处理操作的指令。嵌入在指令内的可以是协处理器指令。处理器310将这些协处理器指令识别为应该由附加的协处理器345执行的类型。因此，处理器310将协处理器总线或其它互连上的这些协处理器指令(或表示协处理器指令的控制信号)发布到协处理器345。(一个或多个)协处理器345接受并执行接收到的协处理器指令。

[0028] 现在参考图4，示出的是根据本发明实施例的第一个更具体的示例性系统400的框图。如图4中所示，多处理器系统400是点对点互连系统，并且包括经由点对点互连450耦合的第一处理器470和第二处理器480。处理器470和480中的每一个可以是某版本的处理器200。在本发明的一个实施例中，处理器470和480分别是处理器310和315，而协处理器438是协处理器345。在另一个实施例中，处理器470和480分别是处理器310协处理器345。

[0029] 处理器470和480被示出分别包括集成存储器控制器(IMC)单元472和482。处理器470还包括作为其总线控制器单元的一部分的点对点(P-P)接口476和478；类似地，第二处理器480包括P-P接口486和488。处理器470、480可以使用P-P接口电路478、488经由点对点(P-P)接口450来交换信息。如图4中所示，IMC 472和482将处理器耦合到相应的存储器，即

存储器432和存储器434，它们可以是本地附接到相应的处理器的主存储器的部分。

[0030] 处理器470、480中的每个可以使用点对点接口电路476、494、486、498经由单独P-P接口452、454来与芯片集490交换信息。芯片集490可以可选地经由高性能接口439来与协处理器438交换信息。在一个实施例中，协处理器438是专用处理器，诸如例如高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器等。

[0031] 可以将共享高速缓存(未示出)包括在任一处理器中或在两个处理器外部、但是仍经由P-P互连与处理器连接，使得如果处理器被置于低功耗模式中则可以将一个或两个处理器的本地高速缓存信息存储在共享高速缓存中。

[0032] 芯片集490可以经由接口496耦合到第一总线416。在一个实施例中，第一总线416可以是外围部件互连(PCI)总线或诸如PCI Express总线或另一第三代I/O互连总线之类的总线，但是本发明的范围不限于此。

[0033] 如图4中所示，各种I/O设备414可以耦合到第一总线416连同总线桥418，总线桥418将第一总线416耦合到第二总线420。在一个实施例中，一个或多个附加处理器415，诸如协处理器、高吞吐量MIC处理器、GPGPU、加速器(诸如例如图形加速器或数字信号处理(DSP)单元)、现场可编程门阵列或任何其它处理器，耦合到第一总线416。在一个实施例中，第二总线420可以是低引脚数(LPC)总线。在一个实施例中，各种设备可以耦合到第二总线420，各种设备包括例如键盘和/或鼠标422、通信设备427和诸如盘驱动器或其它可以包括指令/代码和数据430的大容量存储设备的存储单元428。此外，音频I/O 424可以耦合到第二总线420。注意，其它架构是可能的。例如，代替图4的点对点架构，系统可以实现多分支总线或其它此类架构。

[0034] 现在参考图5，示出的是根据本发明的实施例的第二个更具体的示例性系统500的框图。图4和图5中的类似元素具有类似的参考标号，并且已经从图5省略了图4的某些方面，以避免模糊图5的其它方面。

[0035] 图5图示了处理器470、480可以分别包括集成存储器和I/O控制逻辑(“CL”)472和482。因此，CL 472、482包括集成存储器控制器单元并且包括I/O控制逻辑。图5不仅图示了耦合到CL 472、482的存储器432、434，而且图示了I/O设备514也耦合到控制逻辑472、482。旧有I/O设备515耦合到芯片集490。

[0036] 现在参考图6，示出的是根据本发明的实施例的SoC 600的框图。图2中的类似元素具有类似的参考标号。此外，虚线框是在更先进的SoC上的可选功能。在图6中，(一个或多个)互连单元602耦合到：应用处理器610，其包括一个或多个核202A-N的集合和(一个或多个)共享高速缓存单元206；系统代理单元210；(一个或多个)总线控制器单元216；(一个或多个)集成存储器控制器单元214；可以包括集成图形逻辑、图像处理器、音频处理器和视频处理器的集合或一个或多个协处理器620；静态随机存取存储器(SRAM)单元630；直接存储器存取(DMA)单元632；以及用于耦合到一个或多个外部显示器的显示单元640。在一个实施例中，(一个或多个)协处理器620包括专用处理器，诸如例如网络或通信处理器、压缩引擎、GPGPU、高吞吐量MIC处理器、嵌入式处理器等。

[0037] 本文公开的机制的实施例可以以硬件、软件、固件或此类实现方法的组合来实现。本发明的实施例可以被实现为在包括至少一个处理器、存储系统(包括易失性和非易失性存储器和/或存储元件)、至少一个输入设备和至少一个输出设备的可编程系统上执行的计

算机程序或程序代码。

[0038] 可以将诸如图4中所图示的代码430之类的程序代码应用于输入指令以执行本文所述的功能并生成输出信息。输出信息可以以已知的方式应用于一个或多个输出设备。为了本申请的目的，处理系统包括具有诸如例如数字信号处理器(DSP)、微控制器、专用集成电路(ASIC)或微处理器之类的处理器的任何系统。

[0039] 程序代码可以以高级程序或面向对象编程语言来实现，以与处理系统通信。如果需要，程序代码也可以以汇编或机器语言来实现。事实上，本文所述的机制在范围上不限于任何特定的编程语言。在任何情况下，语言可能是经编译或解释的语言。

[0040] 至少一个实施例的一个或多个方面可以通过存储在表示处理器内的各种逻辑的机器可读介质上的代表性指令来实现，当指令被机器读取时使得机器制造用于执行本文所描述的技术的逻辑。称为“IP核”的这种表示可以存储在有形的机器可读介质上，并提供给各种客户或制造工厂以加载到实际制造逻辑或处理器的制造机器中。

[0041] 这样的机器可读存储介质可以包括但不限于由机器或设备制造或形成的制品的非暂时性有形的布置，包括诸如硬盘、包括软盘、光盘、压缩盘只读存储器(CD-ROM)、压缩盘可重写(CD-RW)和磁光盘的任何其它类型的盘、半导体器件，诸如只读存储器(ROM)、诸如动态随机存取存储器(DRAM)、静态随机存取存储器(SRAM)之类的随机存取存储器(RAM)、可擦除可编程只读存储器(EPROM)、闪存、电可擦除可编程只读存储器(EEPROM)、相变存储器(PCM)，磁卡或光卡，或适于存储电子指令的任何其它类型的介质在内的存储介质。

[0042] 因此，本发明的实施例还包括非暂时性有形的机器可读介质，其包含指令或包含设计数据，诸如定义本文所描述的结构、电路、装置、处理器和/或系统特征的硬件描述语言(HDL)。这样的实施例也可以称为程序产品。

[0043] 在一些情况下，可以使用指令转换器将来自源指令集的指令转换为目标指令集。例如，指令转换器可以将指令转译(例如，使用静态二进制转译，包括动态编译的动态二进制转译)、转变(morph)、仿真或以其它方式转换为要由核处理的一个或多个其它指令。指令转换器可以以软件、硬件、固件或其组合来实现。指令转换器可以在处理器上、在处理器外、或部分在处理器上并且部分在处理器外。

[0044] 图7是根据本发明的实施例的对比使用软件指令转换器将源指令集中的二进制指令转换为目标指令集中的二进制指令的框图。在图示的实施例中，指令转换器是软件指令转换器，但是替换地，指令转换器可以以软件、固件、硬件或其各种组合来实现。图7示出了可以使用x86编译器704编译高级语言702的程序，以生成可由具有至少一个x86指令集核的处理器716本地执行的x86二进制代码706。具有至少一个x86指令集核的处理器716表示任何如下处理器：所述任何处理器可以通过兼容地执行或以其它方式处理(1)英特尔x86指令集核的指令集的大部分或者(2)目标是在具有至少一个x86指令集内核的英特尔处理器上运行的应用或其它软件的目标代码版本以便实现与具有至少一个x86指令集核的英特尔处理器基本相同的结果，来执行与具有至少一个x86指令集核的英特尔处理器基本上相同的功能。x86编译器704表示可操作以生成可以在具有或没有附加联动(linkage)处理的情况下在具有至少一个x86指令集核的处理器716上执行的x86二进制代码706(例如，目标代码)的编译器。

[0045] 类似地，图7示出了可以使用替换的指令集编译器708来编译高级语言702的程序，

以生成可由不具有至少一个x86指令集核的处理器714(例如,具有执行加利福尼亚州桑尼维尔的MIPS科技公司的MIPS指令集和/或执行加利福尼亚州桑尼维尔的ARM控股公司的ARM指令集的核的处理器)在本地执行的替换指令集二进制代码710。指令转换器712用于将x86二进制代码706转换为可由不具有x86指令集核的处理器714在本地执行的代码。该转换后的代码不太可能与替换指令集二进制代码710相同,因为能够实现此的指令转换器很难制造;然而,转换后的代码将完成一般操作,并由来自替换指令集的指令组成。因此,指令转换器712表示通过仿真、模拟或任何其它过程来允许不具有x86指令集处理器或核的处理器或其它电子设备执行x86二进制代码706的软件、固件、硬件或其组合。

[0046] 具有事务能力和日志记录电路以报告事务操作的处理器

图8示出了在半导体芯片上实现的现有技术处理器800,其具有日志记录电路801_1至801_N以用于跟踪多线程程序流的特定方面并从处理器800外部记录此类方面,使得可以在后面研究并理解程序的执行。在实现中,处理器日志记录电路801_1至801_N的每个实例被设计为将其本地处理核的每个线程看作在“区块(chunk)”中执行,其中某些特定查找的事件终止区块。通过将每个线程的区块的序列存储在诸如系统存储器103存储器的存储位置中,可以透彻地分析也许跨所有核805-1到805_X执行线程的更大多线程程序的多线程执行。

[0047] 日志记录电路的每个实例被分配有系统存储器803的特定区域,其中存储其各自的区块。由具体核执行的每个硬件线程被分配有分配给日志记录电路的系统存储器区域内的它自己的相应空间。这里,如本领域中已知的,单个指令执行流水线可以同时执行多个硬件线程(例如,8个硬件线程)。此外,每个处理核可以包含不止一个指令执行流水线(例如,图8示出了每个核具有两个指令执行流水线806)。

[0048] 硬件线程被理解为是在指令执行流水线内活动地执行的线程。指令执行流水线通常被设计为同时执行最大/有限数量的硬件线程,其中所述最大量/极限是由流水线的硬件设计设置的。软件线程被理解为是单个的程序代码指令流。处理器支持的软件线程数可以大大超过硬件线程数。当线程的状态/上下文信息被切换到指令执行流水线中时,软件线程被识别为也为硬件线程。当软件线程的状态/上下文被切换出指令执行流水线时,软件线程失去其硬件线程状态。在一个实施例中,每个硬件线程具有日志记录电路的一个实例(为简单起见,图8仅示出每个核一个日志记录电路)。

[0049] 在实现中,日志记录电路实例(例如,实例801_1)被设计为在以下条件中的任何下终止线程的区块:1)存储器竞争条件;2)线程从活动状态切换到休眠状态;3)转译后备缓冲器(TLB)无效;4)线程被变换到它被配置用于的特权级别之外(例如,响应于中断或异常,从“用户”特权级别到“内核”特权级别的线程变换);5)线程尝试访问不可高速缓存的存储器区域。这里,上述事件中的任何都对多线程程序执行的不确定性方式有所贡献。

[0050] 图8还示出了用于特定线程的区块的分组的现有技术结构的插图(inset)820。如在插图820所观察到的,分组包括:1)分组格式识别符(FMT);2)区块的终止原因(CTR);3)差分时间戳(此分组与前一区块的分组之间的时间)(TSD);4)由区块在线程的前一区块的终止和该区块的终止之间执行的指令数(CS);5)区块的未完成写入数(即,引退但尚未全局可见(提交)的存储操作的数量)(RSW);6)最旧的尚未引退的宏指令的引退加载/存储操作的数量(NTB)。在实施例中,核ID和线程ID由软件层添加,软件层可以基于区块被存储在系统

存储器803中的哪里来确定二者。

[0051] 这里，每个日志记录电路实例801_1至801_N被耦合到处理器的它们各自的处理核805_1至805_N中的“勾连器(hook)”804_1至804_N(例如，在执行各软件线程的相应指令流的指令执行流水线806_1至806_N附近)，勾连器804_1至804_N被设计为检测查找的区块终止事件。在执行特定线程期间，各勾连器检测线程的区块终止事件并将事件报告给日志记录电路801。作为响应，日志记录电路801制定与插图820的结构一致的分组，并使分组被写入到外部存储器803。

[0052] 每个核内的这些勾连器中的一个勾连器被耦合到存储器竞争检测电路807_1至807_N。如图8中观察的那样，每个处理核存在与该核的L1高速缓存808_1至808_N邻近耦合的一个存储器竞争检测电路。每个存储器竞争电路807_1至807_N被设计为检测其相关联的L1高速缓存处的存储器竞争。

[0053] 当两个不同的软件进程(例如，两个不同的线程)尝试访问相同的存储器位置并且那些存储器访问中的至少一个是写入时，发生存储器竞争。这里每个线程都记住当前区块的所有存储器访问(地址)。当检测到与当前区块记住的地址之一的冲突(无论此访问已过去多长时间)时，该区块被终止，并创建新的区块。

[0054] 值得注意的是，在同一个核上的两个不同的线程尝试访问相同的存储器位置时或在两个不同核上的两个不同的线程尝试访问同一个存储器位置时，可能引起竞争。在后一种情况下，第一个核将探听(snoop)第二个核的L1高速缓存。这里，互连网络809用于传送此类探听。

[0055] 每个存储器竞争检测电路807_1至807_N跟踪近期的读取操作和近期的写入操作(称为“读取集”和“写入集”)，并将它们与传入的读取请求和传入的写入请求进行比较。存储器竞争电路将在任何时候检测到指向相同存储器地址的同时的“写入后读取”(RAW)、“写入后写入”(WAW)或“读取后写入”(WAR)操作时检测存储器竞争条件。在各种实施例中，冲突地址的身份可以可选地被包括在针对存储器竞争被记录的区块中(取决于是否需要更大或更小的区块)。

[0056] 图9示出了对图8的现有技术处理器的改进。这里，在实施例中，图9的改进处理器假定上面关于图8描述的所有功能加上下面立即描述的附加改进。

[0057] 如图9中观察到的那样，处理器的各个核905_1至905_N包括执行“事务”的附加能力。事务的执行对应于远远超出传统推测性执行边界的推测性代码的执行。传统的推测性执行(诸如分支预测)允许指令执行流水线在确认所采取的方向是正确的程序流路径之前沿着程序分支执行程序。这里，执行指令的结果被包含在流水线内部(例如，在重排序缓冲器中)，而未被外部写入到处理器的架构状态(寄存器)。照此，推测性执行的指令的结果不是全局可见的。如果预测分支不正确，则流水线被清除(flush)，并且程序流从不正确地预测的分支重新启动。如果预测是正确的，则指令结果被提交到流水线外部的架构状态，以供其它进程全局查看。然而，一般来说，可以包含的数据改变的量是具有有限大小的，并且因此推测性执行的代码的量是有限的。

[0058] 相比之下，支持事务的处理核允许远远超出上面讨论的这类推测性执行的推测性执行(尽管图9的核也可被设计为包括分支预测)。当前的硬件事务内存在L1或L2高速缓存中存储推测性写入。如果事务被中止(abort)，则高速缓存数据被丢弃并且存储器改变决不

会被推送到主存储器，所以其它核永远不会看到它们。在超线程(即，SMT)的情况下，上下文切换自动中止活动的事务，因此在同一个核上执行的另一软件事务不能从另一软件线程中看到飞行中的事务。在事务的情况下，进程能够“好像”它们已经在共享数据项上放置锁定一样进行执行。在基本方法中，进程(例如，软件线程)获取对整个数据库(例如，整个共享存储器，诸如LLC高速缓存和/或系统存储器903或其中保持进程数据的区域)的锁定。进程执行一些逻辑，并且在逻辑完成时确定需要对数据项中的一个或多个做出改变。然后进程将数据项“提交”到数据库，并释放数据库上的锁定，从而允许其它进程访问数据项。

[0059] 在实现中，处理器的执行流水线906_1至906_N具有增强的功能单元以支持允许软件线程相信它已经如上所述地锁定了数据库的指令(例如，XACQUIRE和XRELEASE)。也就是说，XACQUIRE指令在被执行时宣布推测性执行的开始以及数据库上的锁定的获取。XRELEASE指令在被执行时宣布推测性执行的结束以及数据库上的锁定的释放。重要的是，在实现中，处理器900的底层硬件要更多地动作来让软件线程在它实际上尚未在技术上锁定整个数据库、而是使得处理器内的冲突检测硬件921寻找并施行针对相同数据项的竞争线程之间的串行操作时相信它已经在数据库上放置了锁定。

[0060] 在这里，应该清楚的是，如果存在想要使用相同数据库的另一个并行线程，则允许第一软件线程锁定整个数据库可能损害性能。第二线程将会别无选择而只能等到第一线程将其数据提交到数据库并释放锁定为止。在效果上，实际锁定整个数据库将导致使用相同数据库的两个并发线程串行执行而不是并行执行。

[0061] 照此，XACQUIRE指令具有“打开”处理器内的冲突检测硬件921的效果，该冲突检测硬件理解数据库(例如，系统存储器或其特定部分)应该“表现为被锁定”。这意味着冲突检测硬件921将允许另一个进程访问数据库，只要该访问不与由执行XACQUIRE指令的进程进行的访问相竞争并相信它已经获得锁定(这里，竞争访问被理解为意味着相同的存储器地址)。如果检测到竞争访问，线程将被“中止”，这导致事务的状态清除并且程序返回到XACQUIRE指令以重新启动针对该事务的另一尝试。这里，冲突检测电路921检测何时另一进程尝试了访问与执行了XACQUIRE并正在代码的推测区域内执行的事务相同的存储器位置。

[0062] 在另一实现中，处理器还支持允许更高级事务语义学的附加指令(例如，XBEGIN，XEND和XABORT)。XBEGIN和XEND分别与XACQUIRE和XRELEASE实质上作用相同。在这里，XBEGIN宣布推测性执行开始(打开冲突检测电路921)，并且XEND宣布推测性执行的结束(关闭冲突检测电路921)。操作如上面讨论的那样进行，除了事务中止在执行被中止的线程的核的控制寄存器空间922(例如，由一个或多个寄存器电路实现的EAX模型特定寄存器空间)中留下错误代码之外，所述错误代码提供关于中止的更多细节(例如，由ABORT指令导致的中止，事务可能在重试时成功，冲突导致中止，内部缓冲器溢出，命中调试断点，在嵌套事务期间发生中止)。

[0063] 留在寄存器空间922中的信息可以用于将中止之后的程序流引导到除了事务的自动重试中之外的(情况)。另外，处理器可以支持明确地中止事务的指令(例如，XABORT)。XABORT指令使程序员能够定义除了明确设计到处理器硬件中的那些事务中止条件之外的其它事务中止条件。在XABORT的情况下，EAX寄存器将包含由XABORT指令提供的信息(例如，描述引起其执行的事件)。

[0064] 提供事务支持的处理器增加了调试多线程程序代码的复杂性。照此，图9的改进处

理器900包括对日志记录电路901的附加增强，这些附加增强被设计为识别事务的存在并且基于它们来描绘区块。更具体地说，在图9中观察到的核内的附加勾连器930被设计成：1)检测表明事务的推测性代码的执行开始的指令(例如，XACQUIRE或XBEGIN)的执行，并将事件报告给日志记录电路901；以及2)检测表明事务的推测性代码的执行结束的指令(例如，XRELEASE或XEND)的执行，并将事件报告给日志记录电路901。响应于这些事件中的任一个，日志记录电路901将终止区块，创建描述区块终止的分组，并向外将分组报告给系统存储器903(例如，经由存储器控制器909)。

[0065] 另外，新的勾连器930将报告被中止的事务的存在。作为响应，日志记录电路901将终止区块，创建描述区块终止的分组，并向外将分组写入系统存储器903。值得注意的是，在这种方法中，用于日志记录目的的中止检测从处理核905内实际检测到用于中止事务的冲突的冲突检测电路921离去，而不是在存储器竞争检测电路907上。下面更详细地讨论冲突检测电路921和存储器竞争检测电路907之间的关系。在处理器包括包含描述中止的附加信息的寄存器空间922(例如，上述EAX寄存器空间)的实现中，附加勾连器930还被设计为将包含在寄存器空间922中的信息报告给日志记录电路901。在支持明确地终止事务的指令(例如，XABORT)的处理器中，还将创建并向外报告事务中止分组(例如，用EAX寄存器内容(如果可用的话))。

[0066] 图10a示出了由图9的处理器执行的第一方法。如图10a中观察到的那样，执行标记推测性执行的事务代码的执行开始的指令1001。在实施例中，该指令不被视为“已执行”直至它被引退。响应于指令的执行，信号被引导到日志记录电路1002。响应于该信号，日志记录电路创建指示因为事务已经启动所以区块正在被终止的区块终止分组1003。区块终止分组被向外报告1004(例如，通过写入至外部系统存储器中)。

[0067] 图10b示出了由图9的处理器执行的第二方法。如图10b中观察到的那样，执行标记推测性执行的事务代码的执行结束的指令(例如，在事务已经成功地提交其数据改变之后)1011。在实施例中，指令不被视为“已执行”直至它被引退。响应于指令的执行，信号被引导到日志记录电路1012。响应于该信号，日志记录电路创建指示因为事务已经结束所以区块正在被终止的区块终止分组1013。区块终止分组被向外报告1014(例如，通过写入至外部系统存储器中)。

[0068] 图10c示出了由图9的处理器执行的第三方法。如图10c中观察到的那样，硬件(例如，冲突检测电路921)检测到另一线程尝试了到由事务访问的存储器位置的访问1021。发送指示冲突的存在的信号(例如，从冲突检测电路922到正在执行该事务的处理核流水线)1022。响应于该信号，事务被中止，并且信息被写入到控制寄存器空间中1023。响应于事务的中止，信号被发送到日志记录电路1024。响应于发送到日志记录电路的信号，日志记录电路访问寄存器空间并创建指示因为事务已被中止所以区块正被终止的区块终止分组，并且该区块终止分组包括来自寄存器的信息1025。区块终止分组被向外报告1026(例如，通过写入至外部系统存储器中)。

[0069] 图10d示出了由图9的处理器执行的第四方法。如图10d中观察到的那样，执行明确地中止事务的指令，并将信息写入到控制寄存器中1031。响应于事务的中止，信号被发送到日志记录电路1032。响应于发送到日志记录电路的信号，日志记录电路访问寄存器并创建指示因为事务已被中止所以区块正被终止的区块终止分组，并且该区块终止分组包括来自

寄存器的信息1033。区块终止分组被向外报告1034(例如,通过写入至外部系统存储器中)。

[0070] 图11示出了当写出事务相关的区块终止分组时由图9的日志记录电路901写出的分组结构1100的实施例。如图11中观察到的那样,该分组结构大体上维持与图8的现有技术分组820相同的结构。然而,这里,与现有技术的分组820不同的是,分组的区块终止原因(CTR)字段可以包含指示以下附加事件中的任何的信息:1)事务开始;2)事务结束;和3)事务中止。在另外的实现中,CTR字段可以附加地指示处理器是否支持明确的中止指令(例如,XABORT)。

[0071] 除图8的分组结构之外的附加改进在于:最旧的但尚未引退的宏指令信息的引退的加载/存储操作的数量(图8的分组820中的NTB)被替换为"事务状态字"(TSW),其提供描述针对事务的区块终止的附加信息(例如,针对其中CTR是事务终止的情况)。

[0072] 在实现中,在事务中止的情况下,TSW包含(例如EAX)控制寄存器的内容,或者在事务开始或事务结束的情况下,包含"事务嵌套计数器"寄存器(未绘出)的内容。在事务中止的情况下,在实施例中,EAX控制寄存器的内容指示:1)中止是否来自XABORT指令;2)事务是否可能在重试中成功;3)中止是否来自冲突;4)中止是否来自溢出;5)中止是否来自调试断点;6)中止的事务是否被嵌套。对于嵌套事务,处理器被设计为支持事务内的一系列事务(例如,第一事务可以启动另一事务等等)。在其保留寄存器空间内的事务嵌套计数器值基本上保持对当前事务属于哪个内部事务(如果有的话)的跟踪。

[0073] 在实现中,可以启用存储器竞争检测电路907(图8的现有技术日志记录技术的一部分),同时还启用用于事务中止检测的特殊勾连器930和日志记录。这可以通过用允许新的事务勾连器930和存储器竞争勾连器二者同时被启用并活动的模式来设计图9的处理器900来实现。这允许例如记录在事务内可能潜在地发生的全部冲突(例如,因为存储器竞争勾连器使得日志记录电路901在事务的执行期间向外报告任何检测到的冲突(值得注意的是,存储器竞争条件和冲突是类似的事件))。此附加信息在以下实现中可能特别有用,其中提供的与中止相应的控制寄存器信息(例如,上述EAX寄存器)没有详细说明具有导致事务中止的冲突的特定存储器地址(在其它替换实现中,可以修改冲突检测电路922以报告此信息以便进入到寄存器空间中)。

[0074] 此外,区块终止分组的TSW信息可以包括与中止有关的、关于存储器竞争检测电路907是否检测到任何冲突的信息。如果未检测到,则暗示中止事务的冲突检测电路921实际上经历了"假阳性"冲突。在实现中,假阳性在冲突检测电路921处是可能的,因为高速缓存(诸如L1高速缓存)使用散列化电路来确定高速缓存的数据项要被存储在哪里以及通常多个不同的存储器地址可以散列化成相同的高速缓存存储位置的事实。在另外的实现中,存储器竞争检测电路907也能够出于类似的原因而产生假阳性——但是与在事务冲突检测电路921驻留的高速缓存电路中相比,在存储器竞争检测电路中存储器地址的散列化和存储可以是不同的(例如,布隆过滤器用于保持读取集和写入集并且存储器地址被散列化成特定的布隆过滤器位置)。照此,在这种情况下,如果存储器竞争检测电路报告任何冲突,则不能完全依赖它们用于检测事务中止。

[0075] 在另一实施例中,事务相关的区块终止分组的CTR信息指示事务是否由于迟锁获取(late lock acquire,LLA)被终止。迟锁获取是特殊情况,其允许事务提交其数据,即使事务尚未完成。通常,当事务需要被"暂停"(例如响应于异常或不安全指令)时,LLA被施加

以使得其状态能够被外部保存。在事务的状态被外部保存后，事务恢复正常操作。在这种情况下，处理核内的勾连器再次向外向日志记录电路901报告任何LLA事件的发生，日志记录电路901向外报告与LLA及其事务终止有关的区块终止事件。

[0076] 日志记录电路901可以以任何数量的方式实现。在第一极端处，日志记录电路901可以完全在专用的定制逻辑电路中实现。在另一极端处，日志记录电路901可以被实现为执行程序代码以实行其各种功能的微控制器或其它形式的程序代码执行电路(例如，固件)。这两个极端之间的其它混合也是可能的。

[0077] 由于由上述讨论所教导的任何逻辑过程可以用控制器、微控制器或类似组件来执行，这样的处理可以用诸如机器可执行指令之类的程序代码来实现，所述机器可执行指令使得执行这些指令的机器实行某些功能。

[0078] 相信由上述讨论所教导的过程也可以以各种面向对象或非面向对象的计算机编程语言在源级程序代码中描述。可以使用制品来存储程序代码。存储程序代码的制品可以被实施为(但不限于)：一个或多个存储器(例如，一个或多个闪存、(静态，动态或其它)随机存取存储器)，光盘，CD-ROM，DVD ROM，EPROM，EEPROM，磁卡或光卡，或适合于存储电子指令的其它类型的机器可读介质。程序代码还可以通过在传播介质中实施的数据信号(例如，经由通信链路(例如，网络连接))的方式从远程计算机(例如，服务器)下载到请求计算机(例如，客户机)。

[0079] 用于硬件事务内存程序的剖析器的装置和方法

下面描述的本发明的附加实施例包括使用基于跟踪的重放来执行通过使用上面描述的硬件扩展(例如，图8-11和相关联的文本)记录的硬件事务内存(HTM)程序的线程级和函数级剖析分析的软件方法。具体地，本发明的这些实施例包括提供使用上面描述的硬件扩展记录的事务内存(TM)程序的线程级剖析数据的软件剖析技术。

[0080] 基于执行的重放(其中以其被记录的精确方式重执行和重放程序)可以提供使用上述HTM扩展所记录的TM程序的重要和详细的分析。然而，它也可能是过于缓慢的，使得在许多现实世界情况下，重放的执行时间比非重放执行慢1到2个数量级(10x-100x)。在程序在扩展的时间段(诸如数个小时)内执行的现实世界情况下，完成这样的经记录的系统的完整的基于执行的重放可能花费数天、数周甚至数月。照此，这种开销对于许多现实世界的应用是不可接受的。

[0081] 本文描述了基于跟踪的重放机制，其通过处理和分析所有经记录的执行数据来外推给定TM程序的性能瓶颈。因为基于跟踪的重放实际上不会重新执行程序，因此这样的重放可与执行原始程序一样快。事实上，在一些情况下，基于跟踪的重放可能比执行原始程序所花费的对应时间显著地快得多。

[0082] 因为下面描述的技术使用基于跟踪的重放，因此可能无法提供可从基于执行的重放中收集的精确的事务冲突数据。然而，这样精确的冲突检测剖析并不总是需要的。在仅需要具有函数级精度的线程级剖析数据的情况下(这将可能是常见的“第一遍”剖析实践)，本发明的这些实施例可以显著减少系统生成用于TM程序的剖析数据所花费的时间(例如，相比于基于执行的重放在剖析分析中的一到两个数量级的改进)。

[0083] 所描述的方法是相比于诸如Gottschlich等人的PACT 2012“Visualizing Transactional Memory”以及Zyulkyarov等人的PACT 2010“Discovering and

Understanding Performance Bottlenecks in Transactional Applications"中描述的已知解决方案的改进。首先，使用上述硬件扩展，这些实施例遭受可忽略的剖析数据收集开销，这使得它们成为针对硬件事务的可行方法。没有现有系统可以提供可接受的剖析数据收集开销。其次，本文描述的技术与已知的最佳的基于执行的重放技术相比快了数个数量级地提取剖析数据，这使得它们在记录的执行持续数分钟至数小时至数天的环境中是实用的。这种方法的折衷是所描述的实施例在事务冲突方面降低了精度。然而，正如我们的实验数据所示，这种事务冲突数据并不总是需要的。在诸如这些的情况下，所描述的实施例为程序员提供了用于事务程序的剖析器分析的快速周转时间。

[0084] 图12图示了根据本发明的一个实施例使用的系统组件，包括事务调试器(TDB)记录系统1220，其包括TDB记录硬件和/或软件1201(例如，操作系统)以响应于(例如，使用上述硬件扩展)执行事务内存程序代码1230而记录TDB数据。收集的TDB数据然后被存储在由TDB剖析器系统1200使用的一组TDB记录的日志文件1210中。如图所示，TDB剖析系统1200的一个实施例包括用于通过使用基于跟踪的重放技术执行剖析的三个组件：(1)数据解析器和分割器1201；(2)事务排序器和分类器1202；和(3)线程级优化器和事务热点检测器1203。

[0085] 1．数据解析器和分割器

在一个实施例中，数据解析器和分割器1201处理TDB记录的日志文件1210。尽管大部分功能是用任何数据解析器所预期的功能，但是一些方面根据TDB系统的需求进行了独特的定制：

多线程：解析/生成共享内存和事务事件分组信息。上文关于图11描述了这样的分组的一个示例，并且其可以包含与所执行的指令相关的各种元素，诸如通用时间戳信息(例如，用于跨线程的保证同步)，其可以包括事务开始、事务结束和事务中止定时数据。事件分组信息还可以包括上文讨论的事务状态字(TSW)。

[0086] *单线程*：分支跟踪分组信息。这些分组包含关于贯穿程序所采取的分支的信息。此信息可以针对反汇编的二进制来使用以识别指令指针位置，以及同样地当发生事务事件时正在执行的精确函数(例如，当事务冲突发生时正在执行什么函数)。

[0087] 因为从记录的执行中处理的数据可能异常大，因此系统的一个实施例使用(在解析器和分割器模块1201内的)数据分割器来最小化由每个分组所引发的开销。例如，分支跟踪分组包含比共享内存或事务分组显著地少得多的数据量。照此，一个实施例以更压缩的方式定义这些小分组。

[0088] 此压缩的副作用是它需要用于剖析分析的执行次序的分离，因为分组在它们的软件表示中不是统一的。然而，如下面关于事务排序器和分类器所讨论的，可以提供多个接口来访问记录的执行流，其中一个接口允许从同一流中拉取所有事件，这简化了当数据位于跨分支事件和事务事件流时对剖析信息的处理。

[0089] 2．事务排序器和分类器

系统的一个实施例使用多线程事务排序器和分类器1202，其提供用于访问事务数据、分支数据或二者的多个流(API)。图13中示出了事务排序器和分类器1202的一个实施例的细节，其包括用于识别具有函数级冲突数据的事务事件1302的事务事件流1303和分支跟踪流1301。

[0090] 具体地，图13图示了如何使用执行分支跟踪信息和事务事件信息二者的组合，事

务级事件可被提取并被互相关至反汇编的二进制信息，这允许事务中止至少被关联至在事务被中止时正执行的当前函数。

[0091] 在一个实施例中，这以如下方式完成。当事务中止发生时，剖析系统执行反向查找以找到对应的“事务开始”事件。一旦被识别，(从分支跟踪流1301)捕获在事务的执行期间所采取的所有可能分支的时间戳范围。来自包含在周围时间戳范围内的分支的指令指针(IP)中的每一个都针对被执行程序的反汇编二进制来使用。一旦在反汇编二进制中找到对应IP，就识别使用这些IP的函数(或多个函数)，这允许程序员看到哪个函数(或哪些函数)导致事务冲突，以及因此哪个函数(或哪些函数)应该被分析以实现性能优化。

[0092] 3．线程级优化器和事务热点检测器

除了图14(其图示了具有线程ID 1-4的四个线程的度量)中所示的为每个线程提供基本度量之外，系统的一个实施例通过分析图14中所示的提取的高级别数据来外推哪些线程可能引发最大的性能改进。此热点检测对于将程序员指向一般兴趣区域是有用的。

[0093] 当使用图14中所示的高级别数据并然后使用来自图13的更低级别数据时，可以识别具有函数级精度的线程级事务性能瓶颈，所有都不需要进行基于执行的重放。

[0094] 这已经通过包括在四核RTM机器上使用四个线程的三个微基准测试的优化结果而实验性地证实了。图15A图示了具有HTM和SGL的链表的结果，图15B图示了仅具有HTM的链表的结果，并且图15C图示了仅具有HTM的散列表的结果，每个图表绘出原始性能(1)(没有本文描述的本发明的实施例)对照考虑到结果的剖析数据之后的优化性能(2)(经由本发明的实施例实现)。结果，可以看出本发明的实施例相比现有实现取得了显著的时间改进。

[0095] 图16中图示了根据本发明的一个实施例的方法。该方法可以在上述架构的上下文内实现，但不限于任何特定的架构。

[0096] 在1601处，记录与事务内存程序代码的执行相关的数据。如同提到的那样，在一个实施例中，数据包括与事务内存程序代码中的事务事件和分支的执行相关的数据。数据可以被存储在例如日志/跟踪文件或数据库中。

[0097] 在1602处，使用基于跟踪的重放技术分析记录的数据的部分，并且在1603处，基于该分析，生成包括函数级冲突的指示的剖析数据。例如，如上文讨论的，剖析器可以从记录的数据中提取事务级事件，并将事务级事件与来自事务内存程序代码的反汇编二进制信息互相关。然后，剖析器可以生成互相关的结果，包括识别出的当发生事务中止操作时执行的函数。

[0098] 最后，在1604处，可以使用函数级冲突数据来优化事务内存程序代码。例如，一旦事务中止操作已经与特定函数相关联，则可以修改该函数和/或该函数周围的程序代码，使得中止操作不再发生或更少频次地发生。

[0099] 本发明的实施例可以包括上面已经描述的各种步骤。这些步骤可以体现在机器可执行指令中，所述机器可执行指令可以用于使通用或专用处理器执行步骤。替换地，这些步骤可以由包含用于执行步骤的硬连线逻辑的特定硬件部件或由编程的计算机部件和定制硬件部件的任何组合来执行。

[0100] 如本文所述，指令可以指代被配置为执行某些操作或具有存储在体现在非暂时性计算机可读介质中的存储器中的预定功能或软件指令的硬件的特定配置，诸如专用集成电路(ASIC)。因此，图中所示的技术可以使用在一个或多个电子设备(例如，终端站、网络元件

等)上存储和执行的代码和数据来实现。这样的电子设备使用诸如非暂时性计算机机器可读存储介质(例如,磁盘;光盘;随机存取存储器;只读存储器;闪存设备;相变存储器)和暂时性计算机机器可读通信介质(例如,电、光、声或其它形式的传播信号——诸如载波、红外信号、数字信号等)之类的计算机机器可读介质来存储和(内部地和/或通过网络与其它电子设备)传送代码和数据。

[0101] 此外,这样的电子设备通常包括耦合到一个或多个其它部件的一个或多个处理器的集合,所述一个或多个其它部件诸如是一个或多个存储设备(非暂时性机器可读存储介质)、用户输入/输出设备(例如键盘、触摸屏和/或显示器)和网络连接。处理器集合和其它部件的耦合通常通过一个或多个总线和桥(也称为总线控制器)。携带网络流量的存储设备和信号分别表示一个或多个机器可读存储介质和机器可读通信介质。因此,给定电子设备的存储设备通常存储用于在该电子设备的一个或多个处理器的集合上执行的代码和/或数据。当然,可以使用软件、固件和/或硬件的不同组合来实现本发明的实施例的一个或多个部分。贯穿该具体实施方式,出于解释的目的,阐述了许多具体细节以便提供对本发明的透彻理解。然而,对于本领域技术人员清楚的是,可以在没有这些具体细节中的一些的情况下实践本发明。在某些情况下,为了避免模糊本发明的主题,未以详细的细节描述公知的结构和功能。因此,本发明的范围和精神应根据以下权利要求来判断。

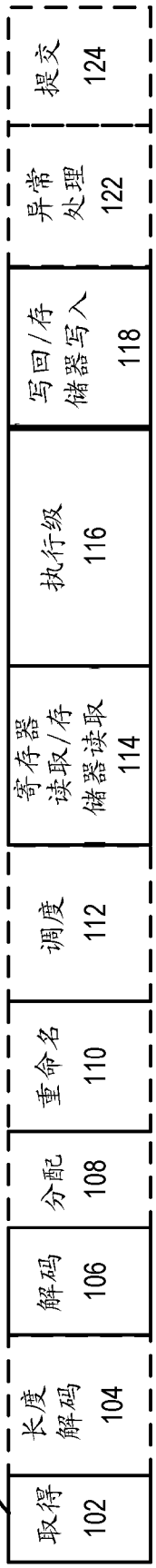

流水线100
取得 102
长度解码 104
解码 106
分配 108
重命名 110
调度 112
寄存器读取/存储器读取 114
执行级 116
写回/存储器写入 118
异常处理 122
提交 124

图 1A

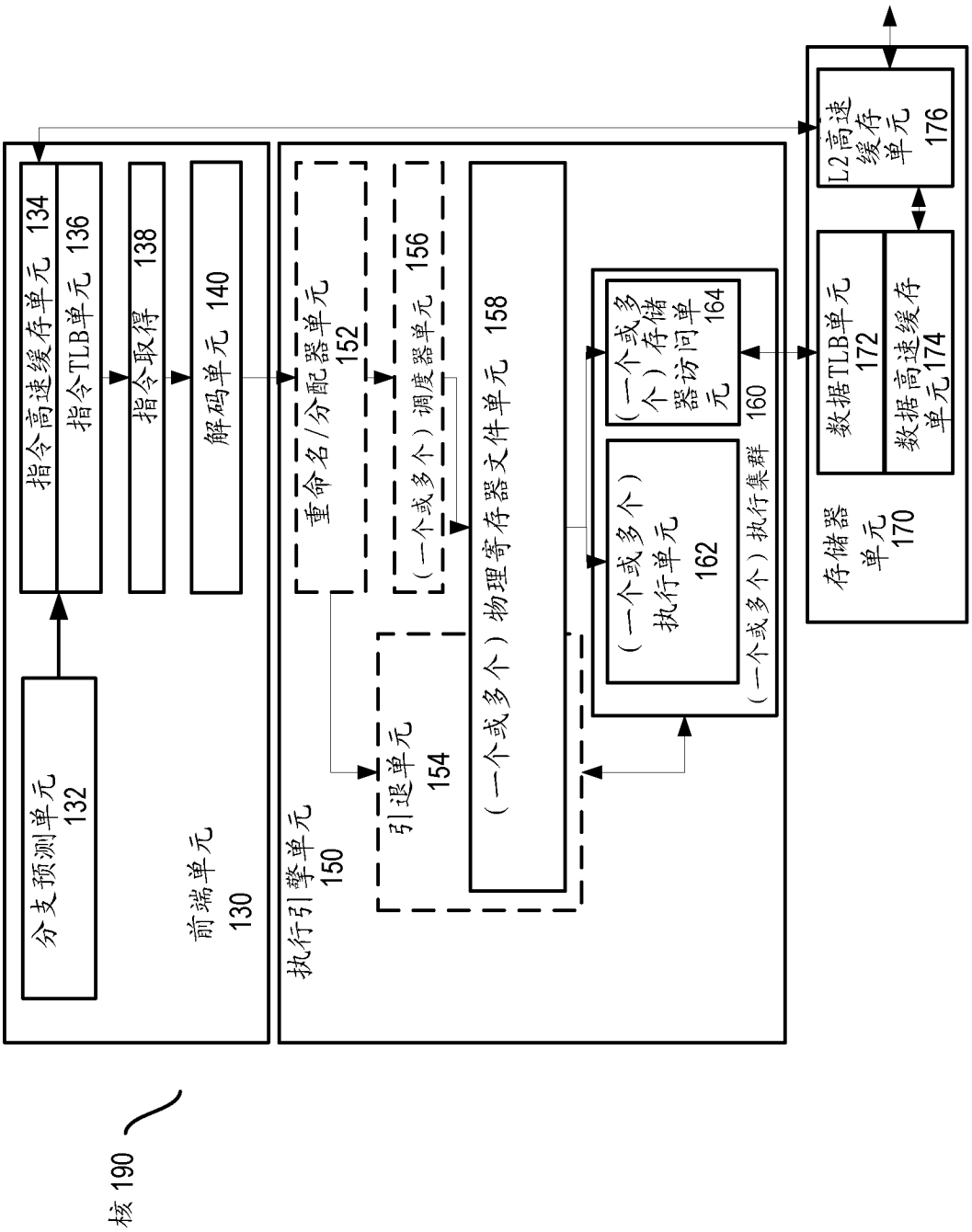

核 190
前端单元 130
分支预测单元 132
指令高速缓存单元 134
指令TLB单元 136
指令取得 138
解码单元 140
执行引擎单元 150
重命名/分配器单元 152
引退单元 154
(一个或多个)调度器单元 156
(一个或多个)物理寄存器文件单元 158
(一个或多个)执行集群 160
(一个或多个)执行单元 162
(一个或多个)存储器访问单元 164
存储器单元 170
数据TLB单元 172
数据高速缓存单元174
L2高速缓存单元 176

图 1B

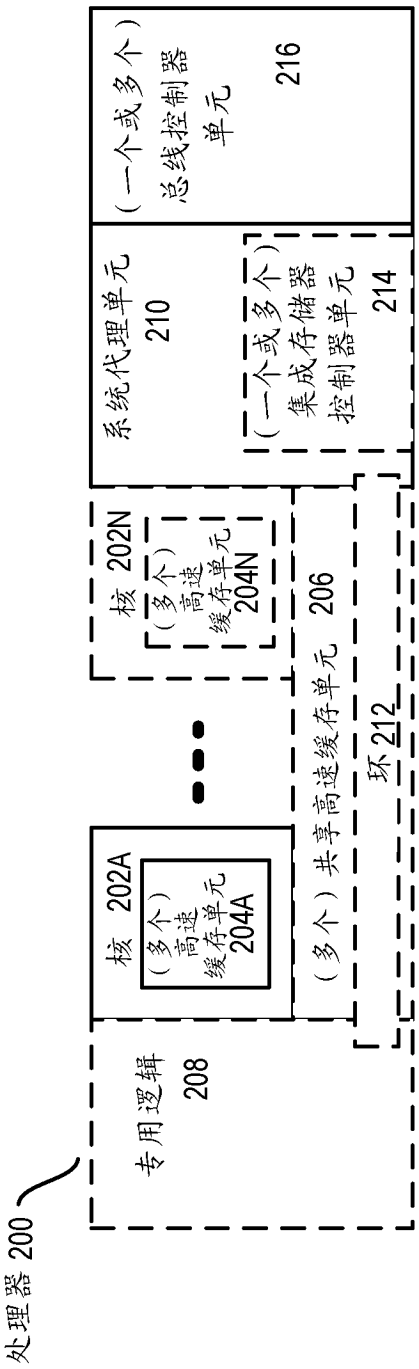

处理器 200
专用逻辑 208
核 202A
（多个）高速缓存单元 204A
核 202N
（多个）高速缓存单元 204N
（多个）共享高速缓存单元 206
环 212
系统代理单元 210
（一个或多个）集成存储器控制器单元 214
（一个或多个）总线控制器单元 216

图 2

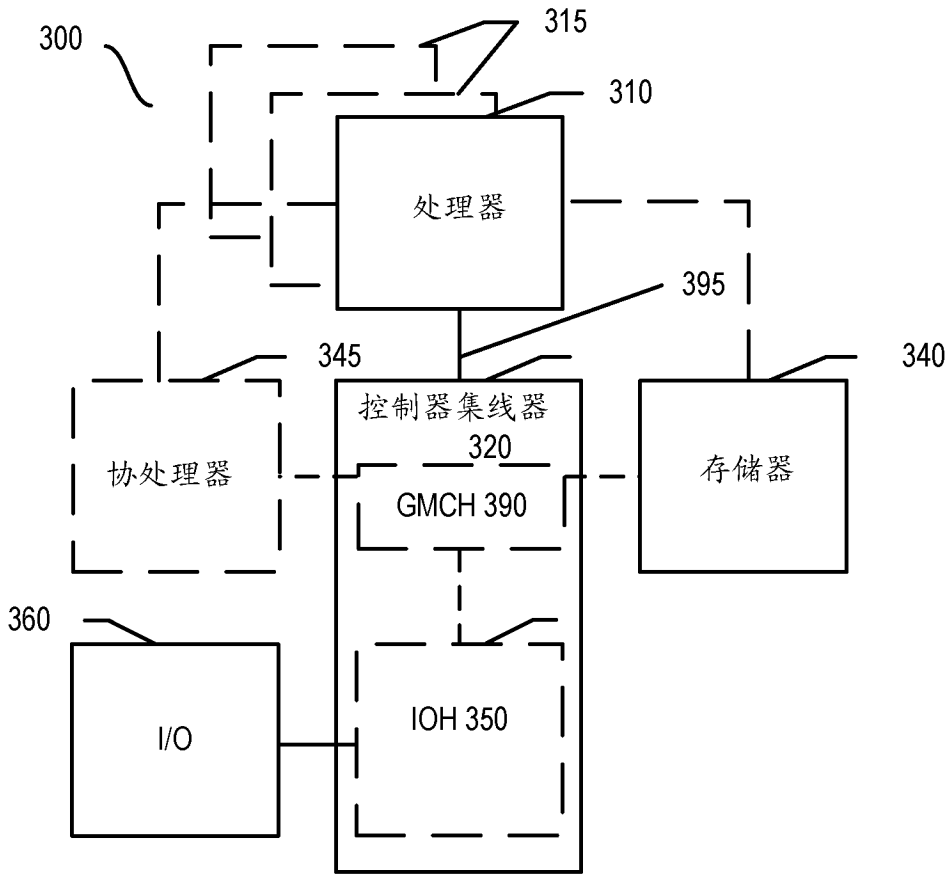

300
315
310
处理器
395
345
340
控制器集线器
320
协处理器
GMCH 390
存储器
360
I/O
IOH 350

图 3

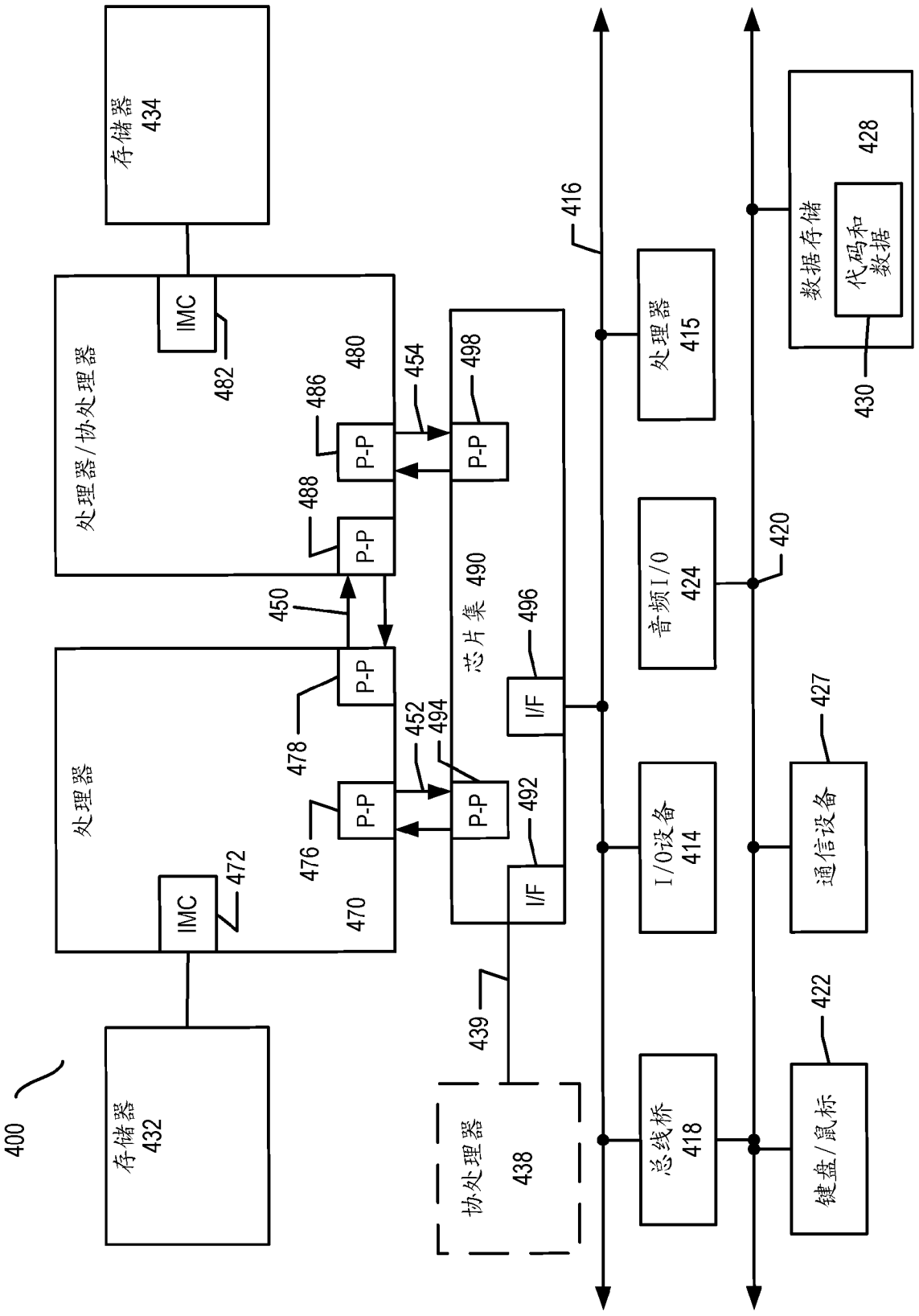

400
存储器 432
存储器 434
处理器 470
处理器/协处理器 480
IMC 472
IMC 482
P-P 476
P-P 478
P-P 452
P-P 494
P-P 486
P-P 488
P-P 454
P-P 498
450
芯片集 490
I/F 492
I/F 496
协处理器 438
439
处理器 415
416
I/O设备 414
音频I/O 424
420
通信设备 427
总线桥 418
键盘/鼠标 422
数据存储 428
代码和数据 430

图 4

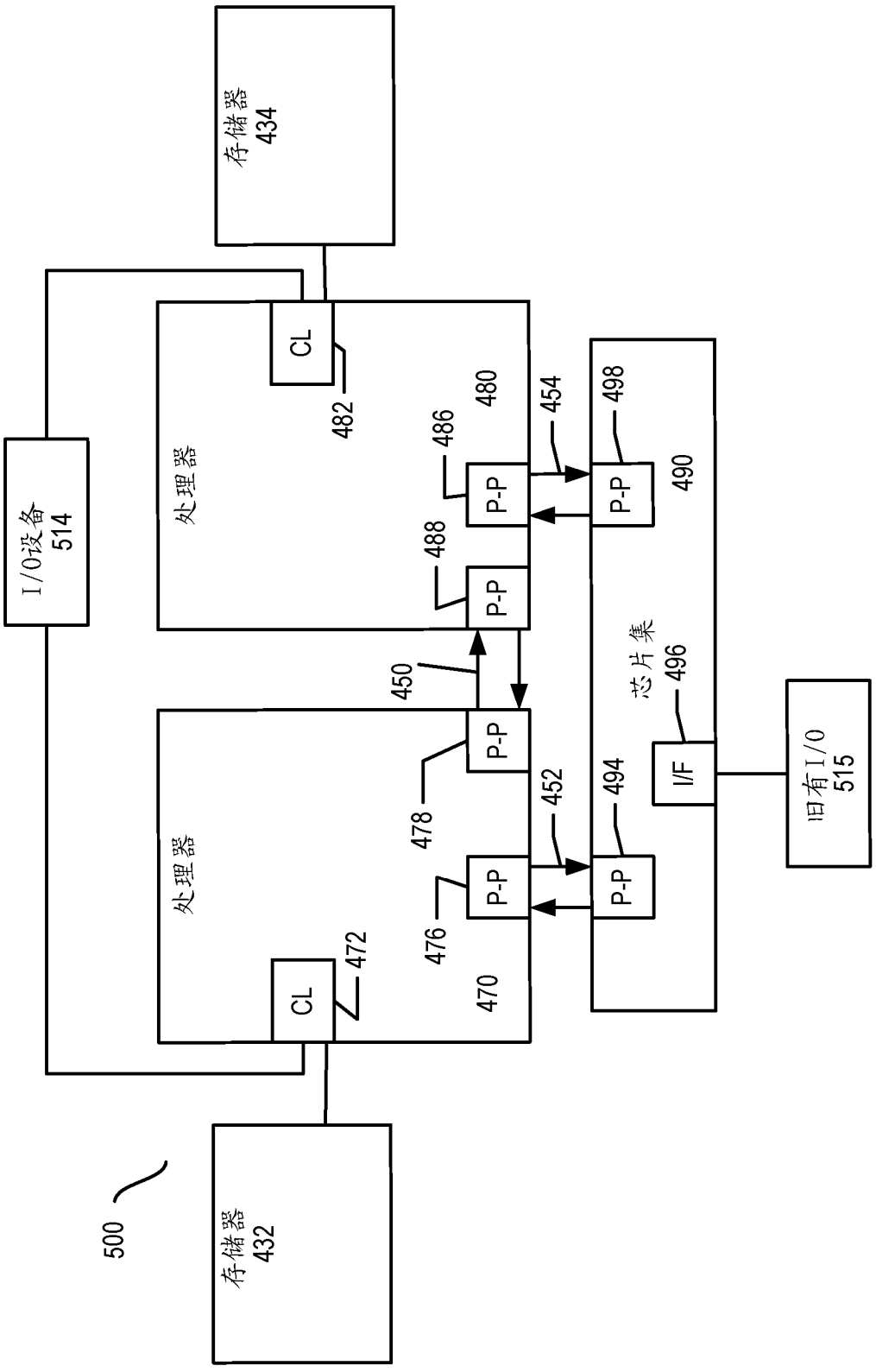

存储器
434
CL
482
处理器
480
486
P-P
488
P-P
454
498
P-P
490
芯片集
I/O设备
514
450
P-P
478
处理器
452
494
P-P
476
P-P
496
I/F
旧有I/O
515
CL
472
470
500
存储器
432

图 5

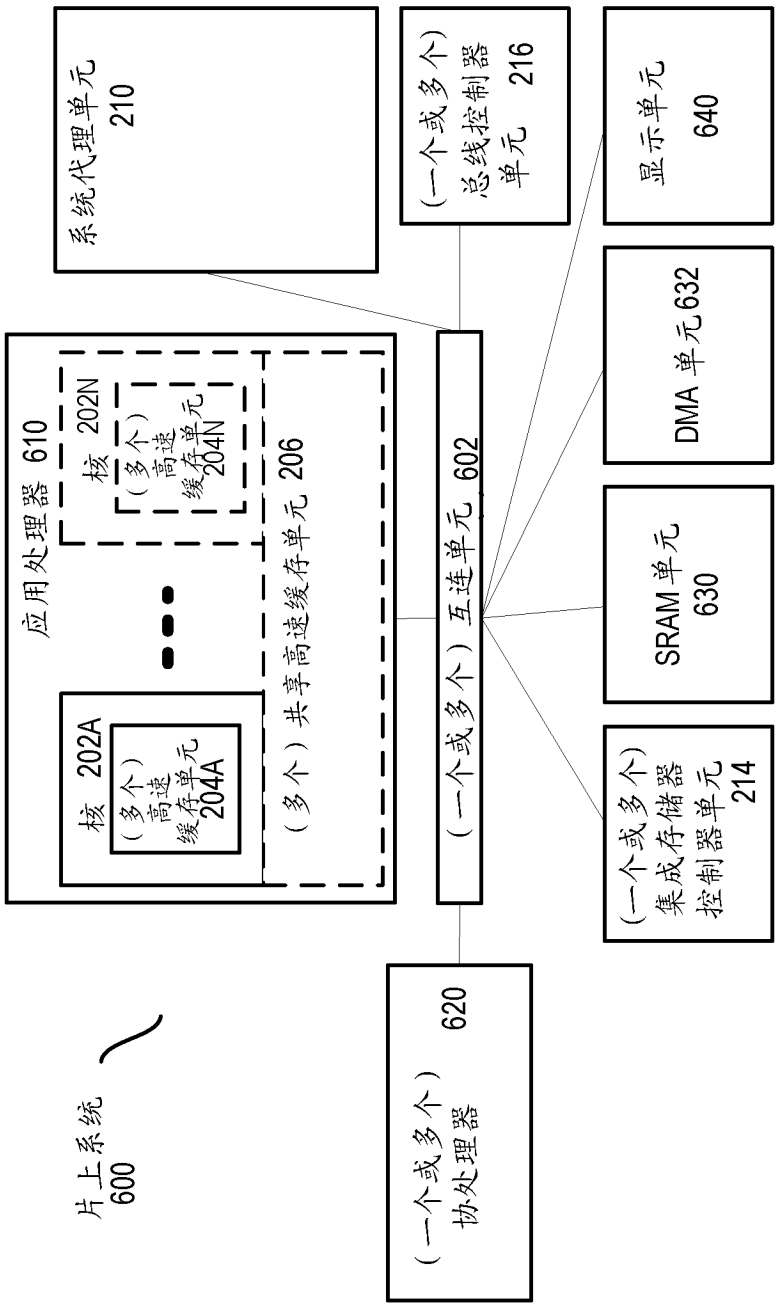

片上系统
600
系统代理单元
210
(一个或多个)
总线控制器
单元 216
显示单元
640
应用处理器 610
核 202A
(多个)
高速
缓存单元
204A
核 202N
(多个)
高速
缓存单元
204N
(多个)共享高速缓存单元 206
(一个或多个)互连单元 602
DMA 单元 632
SRAM 单元
630
(一个或多个)
集成存储器
控制器单元
214
(一个或多个)
协处理器
620

图 6

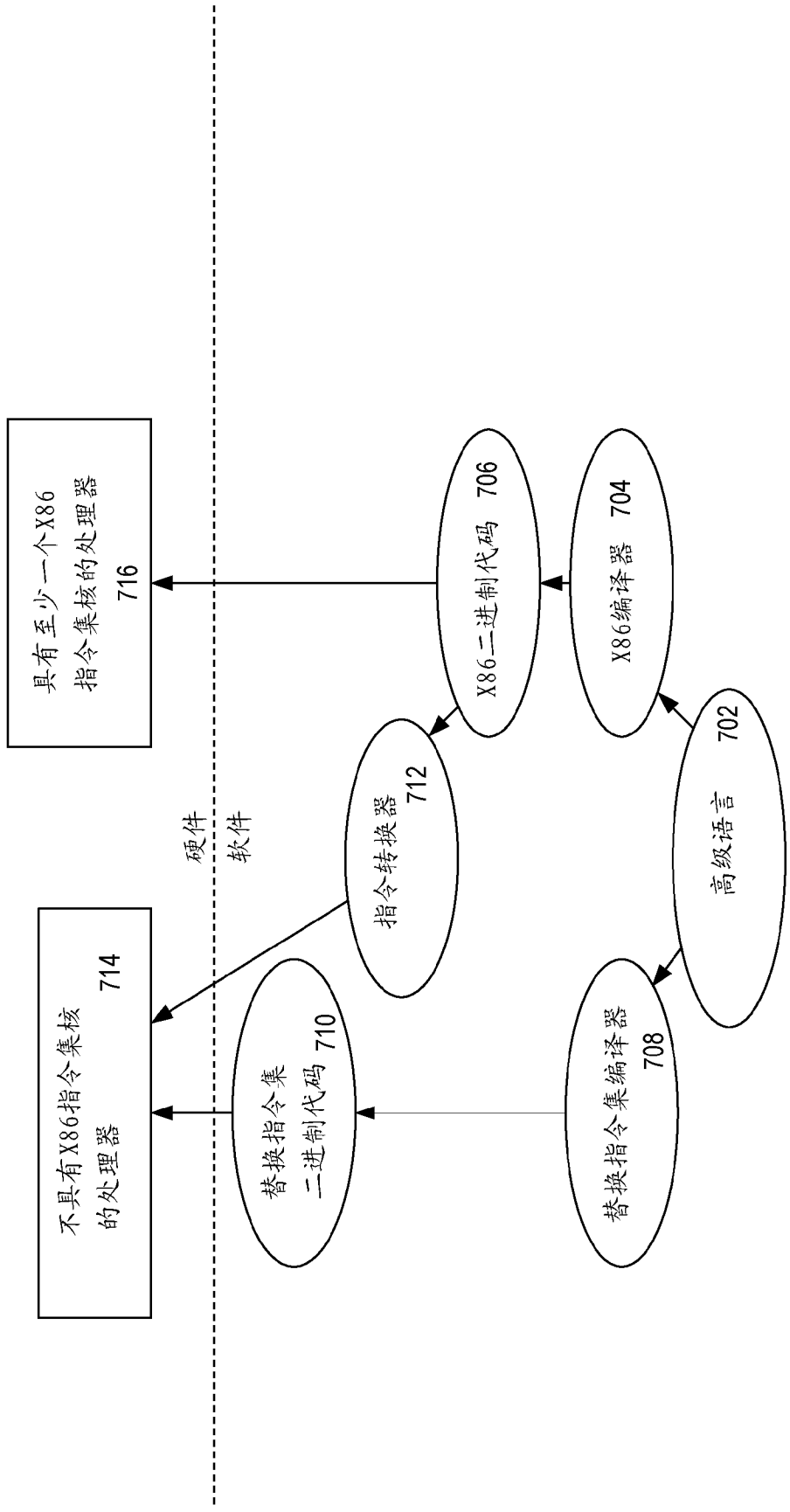

具有至少一个X86
指令集核的处理器
716
X86二进制代码 706
X86编译器 704
指令转换器
712
高级语言 702
硬件
软件
不具有X86指令集核
的处理器 714
替换指令集
二进制代码 710
替换指令集编译器
708

图 7

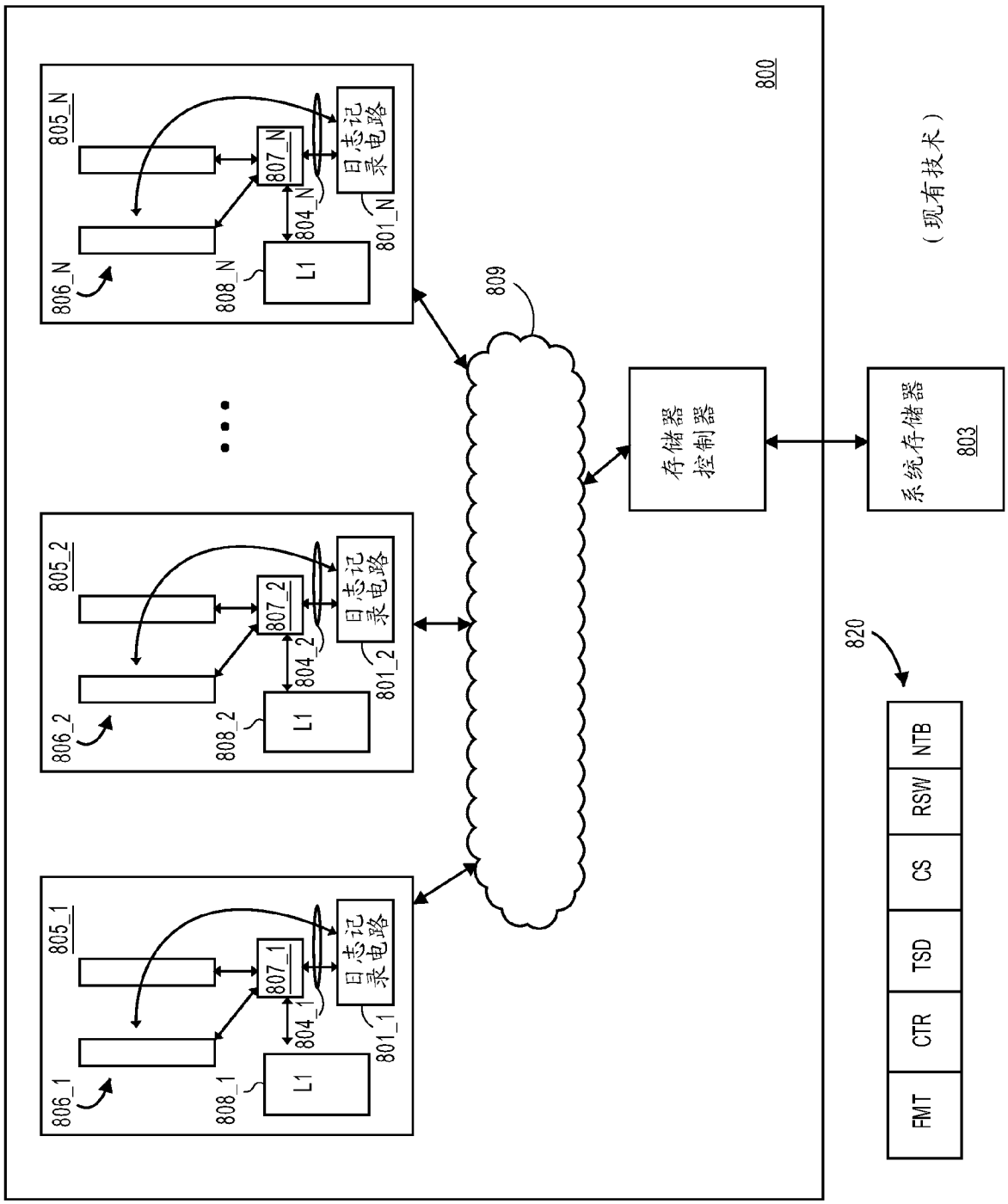

800
805_N
807_N
804_N
801_N
806_N
808_N
L1
日志记录电路
809
805_2
807_2
804_2
801_2
806_2
808_2
805_1
807_1
804_1
801_1
806_1
808_1
存储器控制器
系统存储器 803
820
FMT
CTR
TSD
CS
RSW
NTB
（现有技术）

图 8

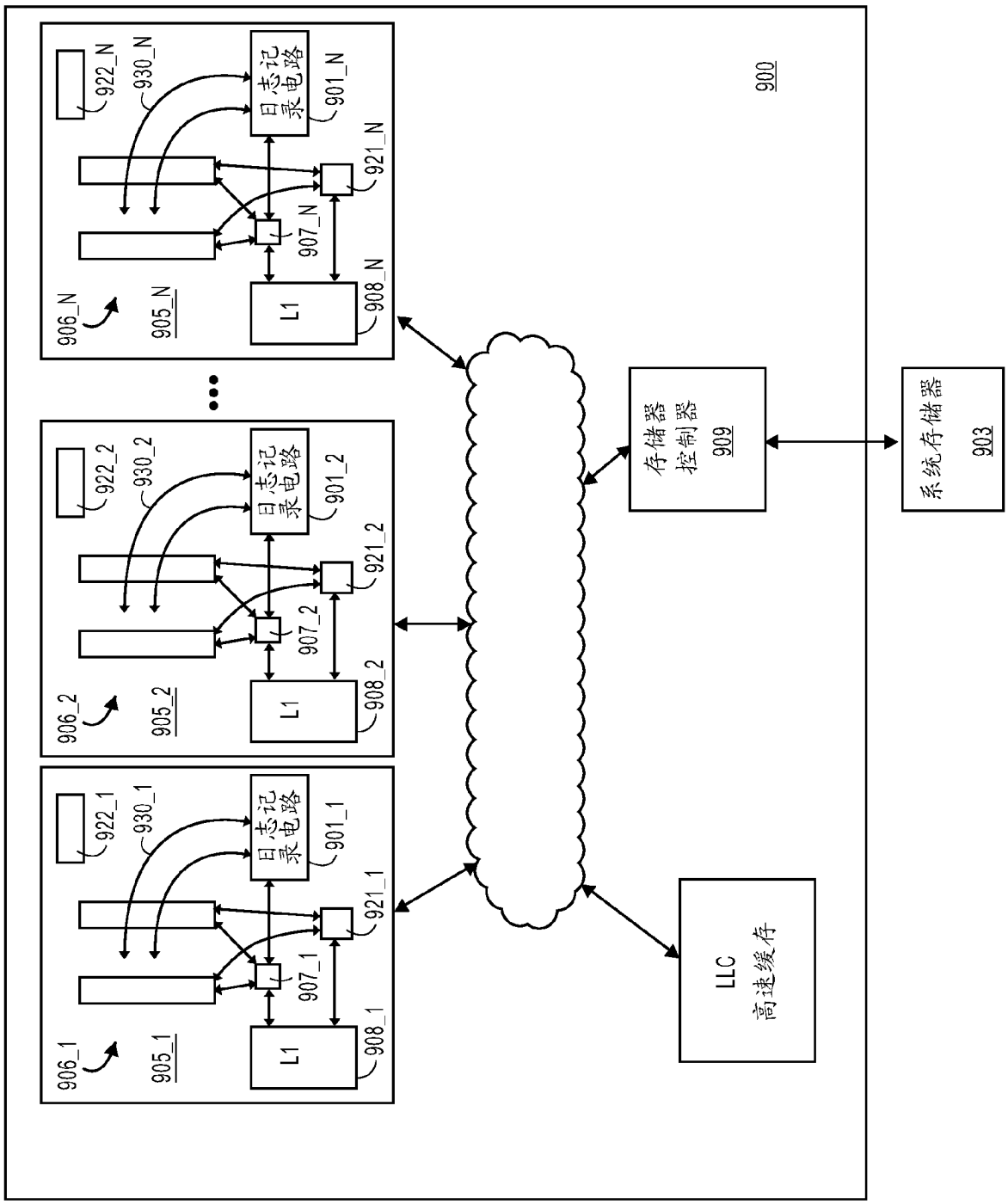

900
906_N
905_N
922_N
930_N
日志记录电路
901_N
921_N
907_N
908_N
L1
906_2
905_2
922_2
930_2
日志记录电路
901_2
921_2
907_2
908_2
L1
906_1
905_1
922_1
930_1
日志记录电路
901_1
921_1
907_1
908_1
L1
存储器控制器
909
系统存储器
903
LLC
高速缓存

图 9

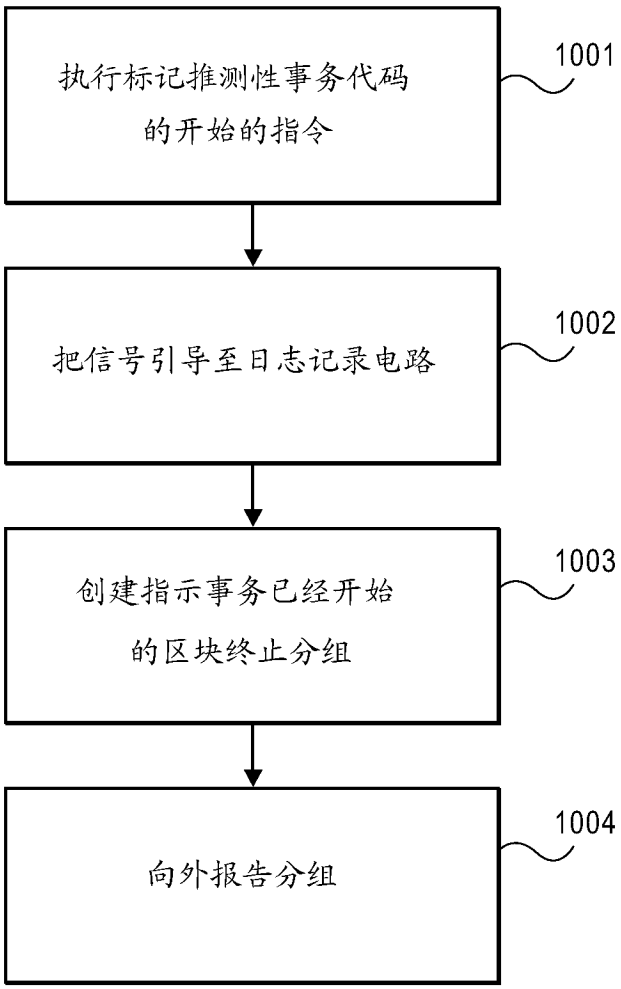

执行标记推测性事务代码的开始的指令
1001
把信号引导至日志记录电路
1002
创建指示事务已经开始的区块终止分组
1003
向外报告分组
1004

图 10A

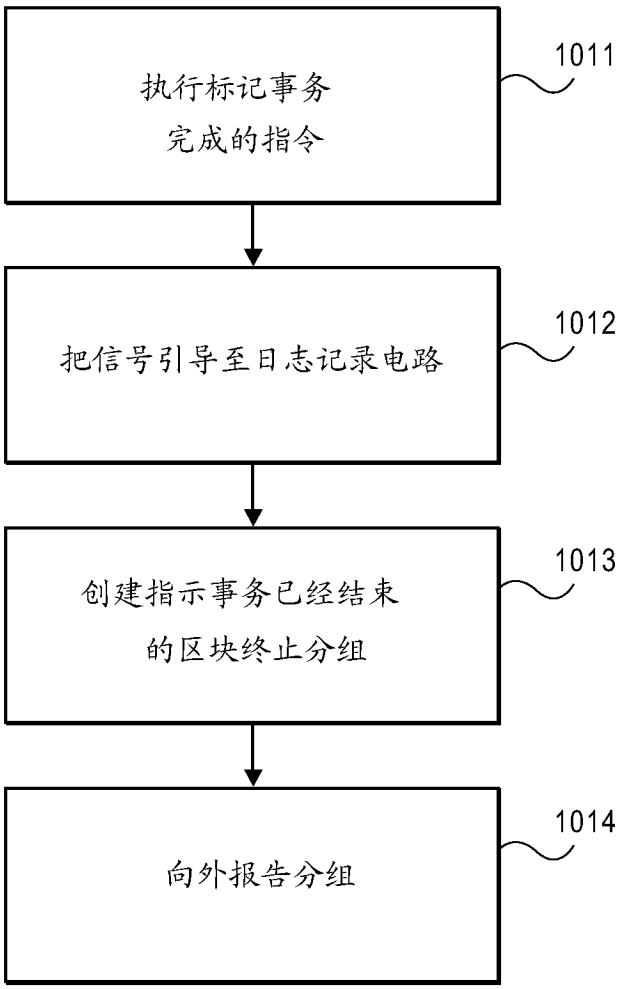

执行标记事务
完成的指令
1011
把信号引导至日志记录电路
1012
创建指示事务已经结束
的区块终止分组
1013
向外报告分组
1014

图 10B

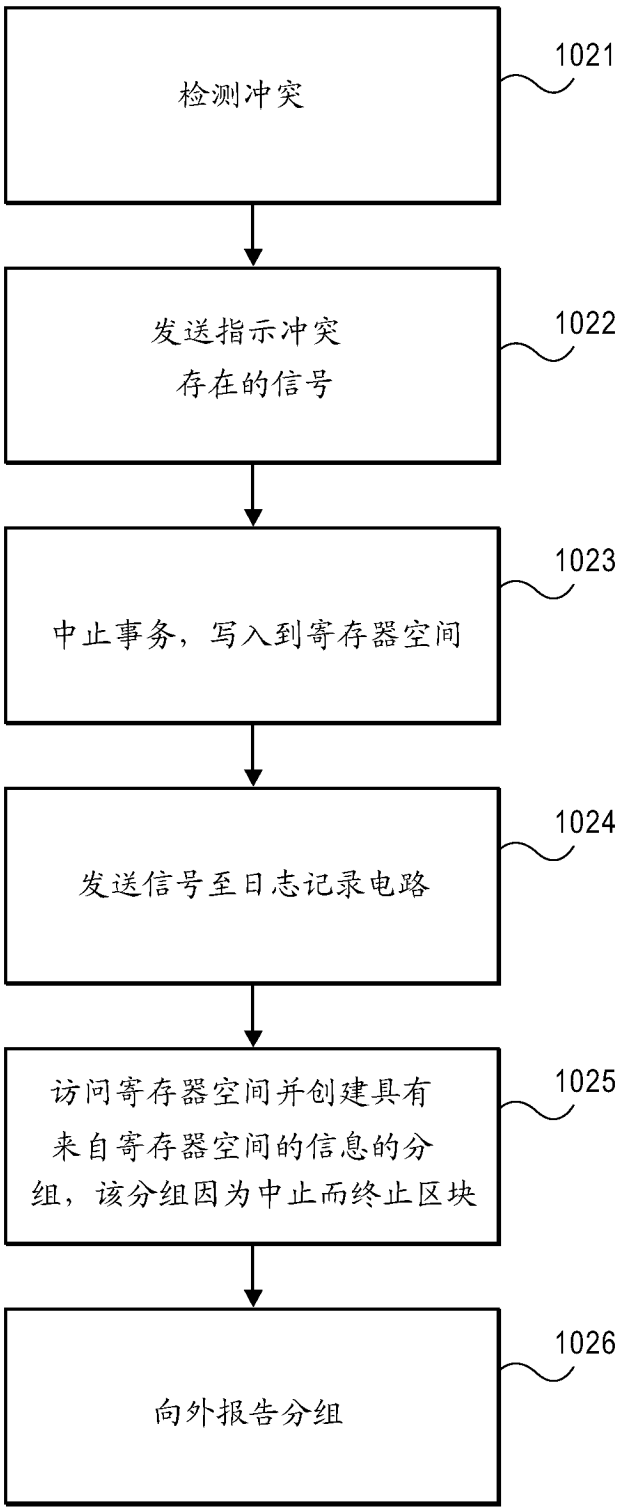

检测冲突
1021
发送指示冲突
存在的信号
1022
中止事务，写入到寄存器空间
1023
发送信号至日志记录电路
1024
访问寄存器空间并创建具有
来自寄存器空间的信息的分
组，该分组因为中止而终止区块
1025
向外报告分组
1026

图 10C

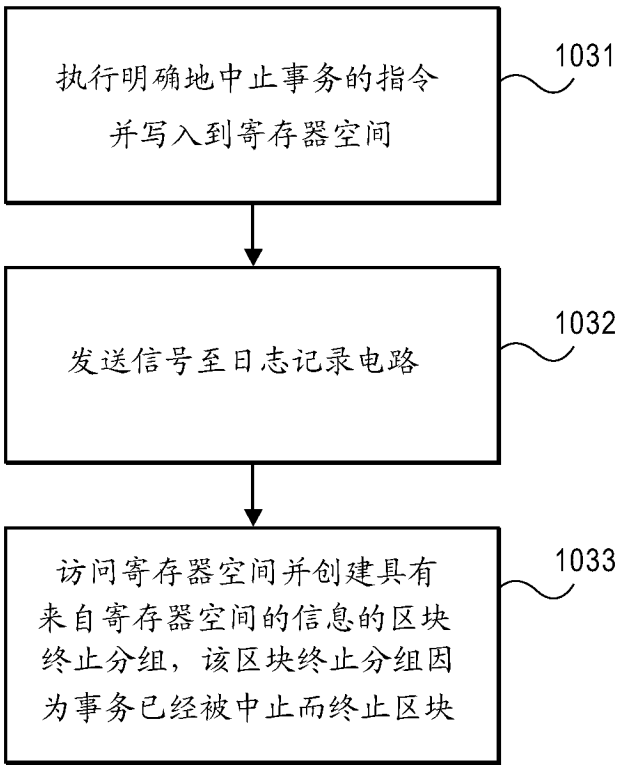


图 10D

1100

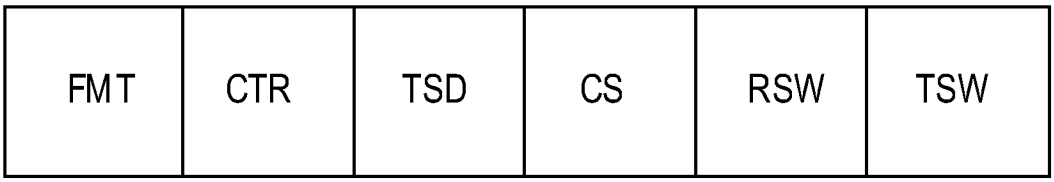


图 11

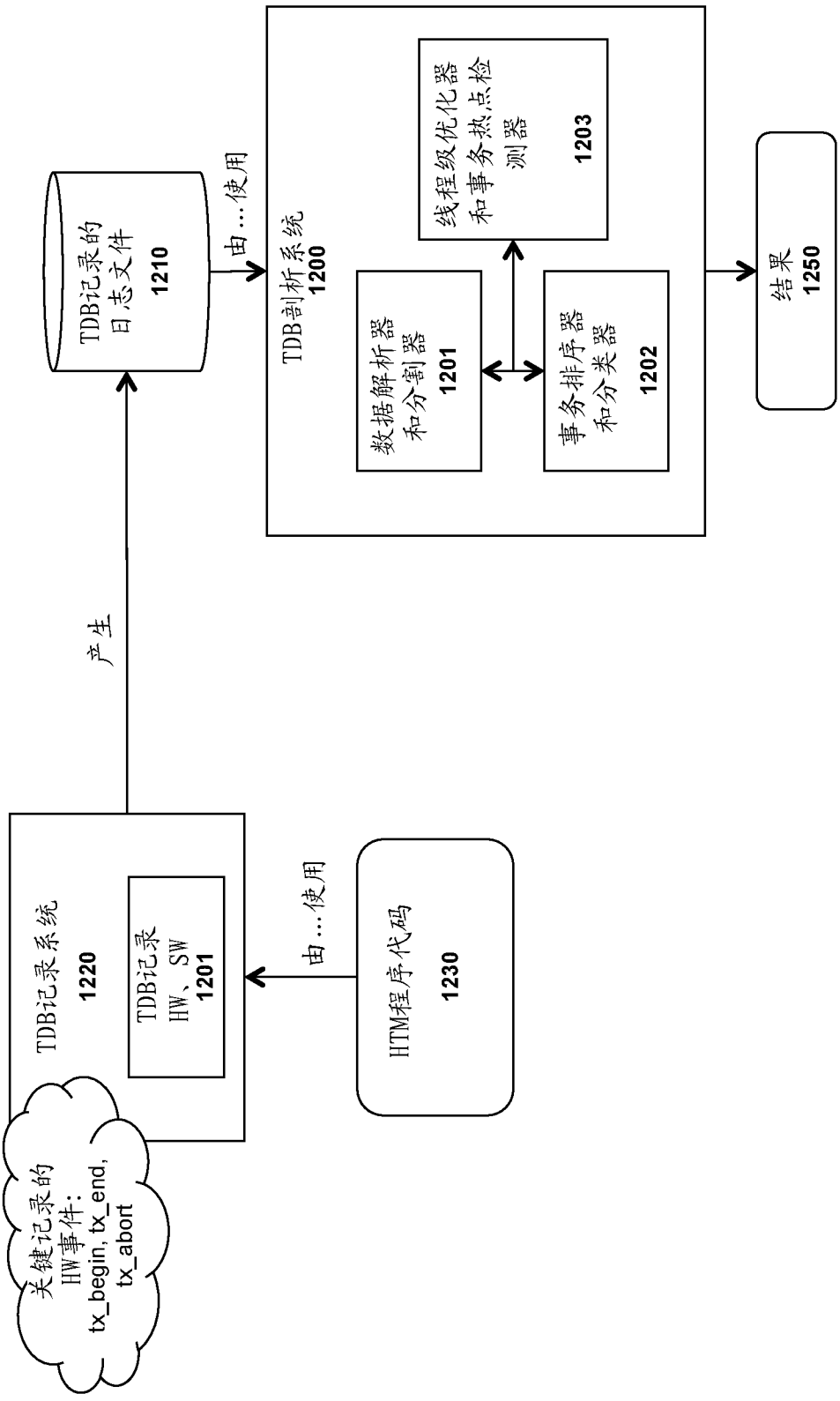

TDB记录的
日志文件
1210
由...使用
TDB剖析系统
1200
线程级优化器
和事务热点检
测器
1203
数据解析器
和分割器
1201
事务排序器
和分类器
1202
结果
1250
产生
TDB记录系统
1220
TDB记录
HW、SW
1201
由...使用
HTM程序代码
1230
关键记录的
HW事件:
tx_begin, tx_end,
tx_abort

图 12

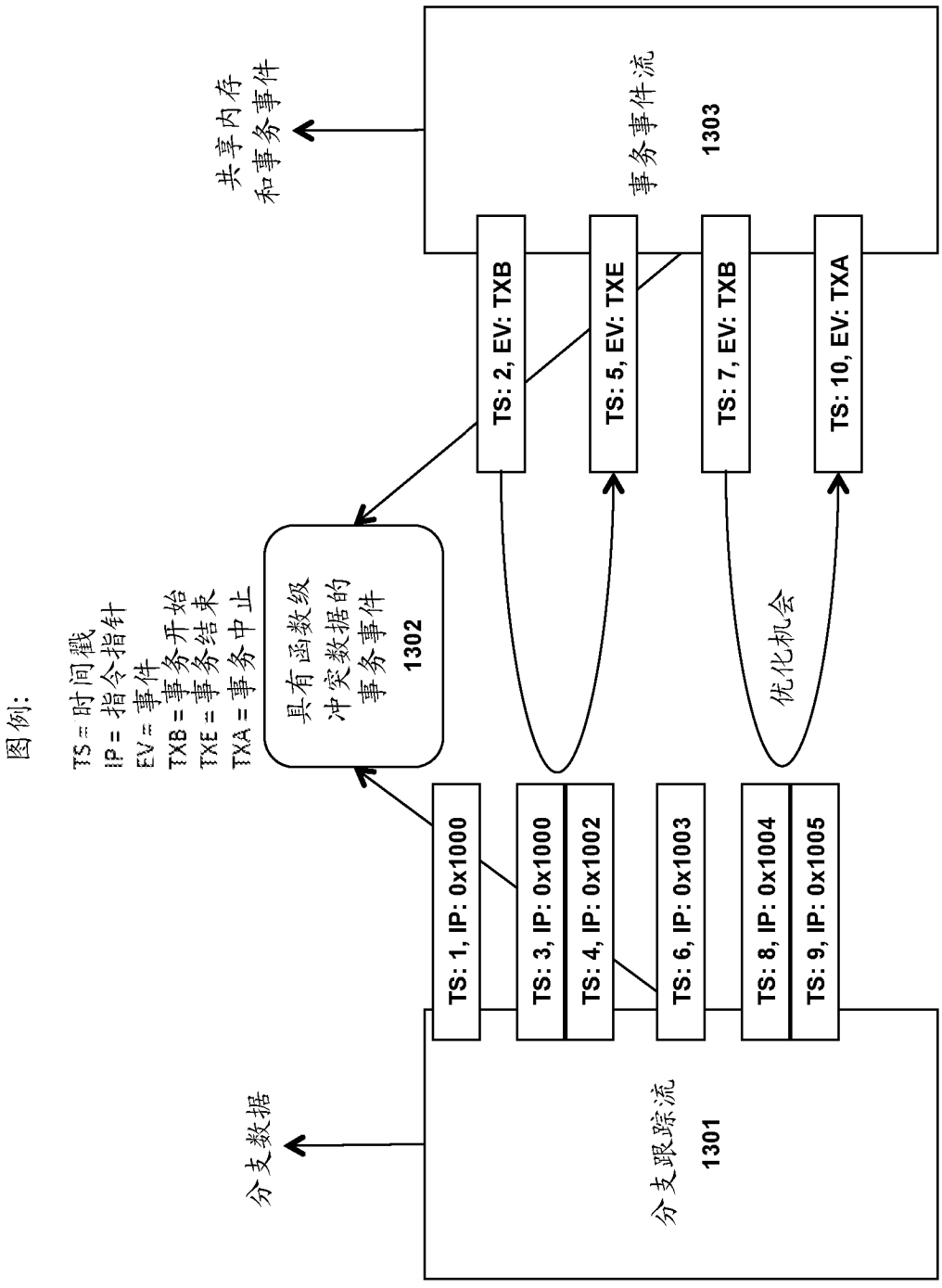

图例:
TS = 时间戳
IP = 指令指针
EV = 事件
TXB = 事务开始
TXE = 事务结束
TXA = 事务中止
具有函数级
冲突数据的
事务事件
1302
共享内存
和事务事件
事务事件流
1303
TS: 2, EV: TXB
TS: 5, EV: TXE
TS: 7, EV: TXB
TS: 10, EV: TXA
优化机会
TS: 1, IP: 0x1000
TS: 3, IP: 0x1000
TS: 4, IP: 0x1002
TS: 6, IP: 0x1003
TS: 8, IP: 0x1004
TS: 9, IP: 0x1005
分支数据
分支跟踪流
1301

图 13

```
Thread id: 1
Commits: 1000, average # of instructions per commit: 175
Total # of instructions committed: 175560
Aborts: 9, average # of instructions per abort: 52
Total # of instructions aborted: 472

Start time: 78,841,926
End time:
Total time: 32,159,910

   Aborts by threads:

      Thread 4 -- aborts: 4, instr: 340

Thread id: 2
Commits: 1000, average # of instructions per commit: 142
Total # of instructions committed: 142412
Aborts: 5, average # of instructions per abort: 56
Total # of instructions aborted: 280

Start time: 60,066,054
End time: 84,153,364
Total time: 24,087,310

   Aborts by threads:

      Thread 1 -- aborts: 1, instr: 0
```

图 14

```
Thread id: 3
Commits: 1000, average # of instructions per commit: 485
Total # of instructions committed: 485996
Aborts: 20, average # of instructions per abort: 115
Total # of instructions aborted: 2315

Start time: 78,587,202
End time:
Total time: 63,756,410

   Aborts by threads:


Thread id: 4
Commits: 1000, average # of instructions per commit: 412
Total # of instructions committed: 412520
Aborts: 26, average # of instructions per abort: 111
Total # of instructions aborted: 2891

Start time: 89,972,645
End time:
Total time: 62,787,944

   Aborts by threads:

      Thread 1 -- aborts: 8, instr: 872
      Thread 3 -- aborts: 6, instr: 570
```

图 14(续)

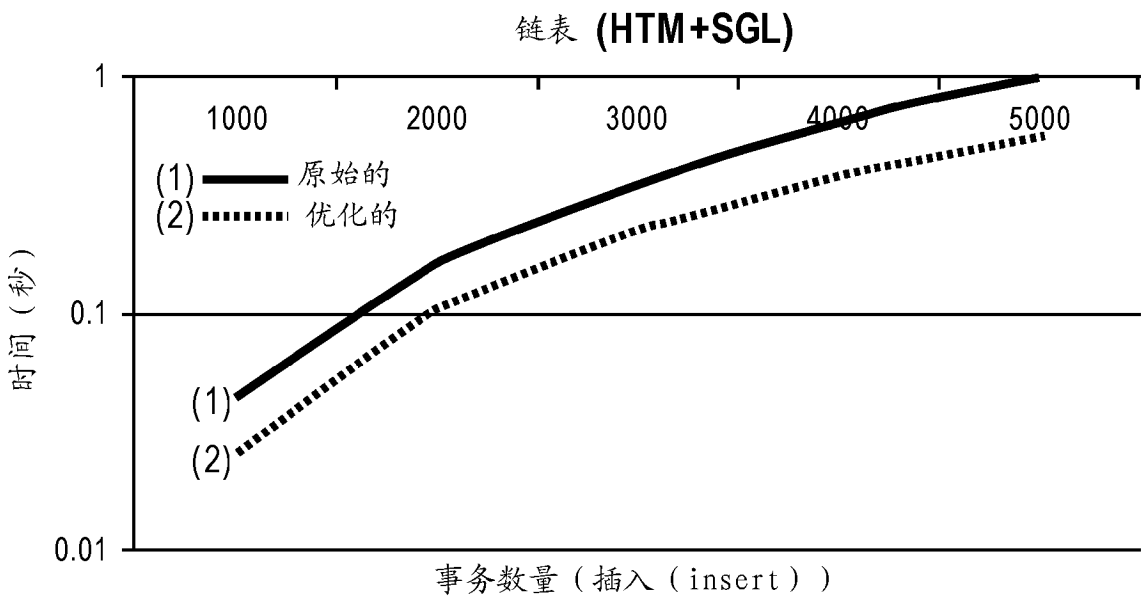

链表 (HTM+SGL)
1
0.1
0.01
1000
2000
3000
4000
5000
(1) 原始的
(2) 优化的
时间（秒）
事务数量（插入（insert））

图 15A

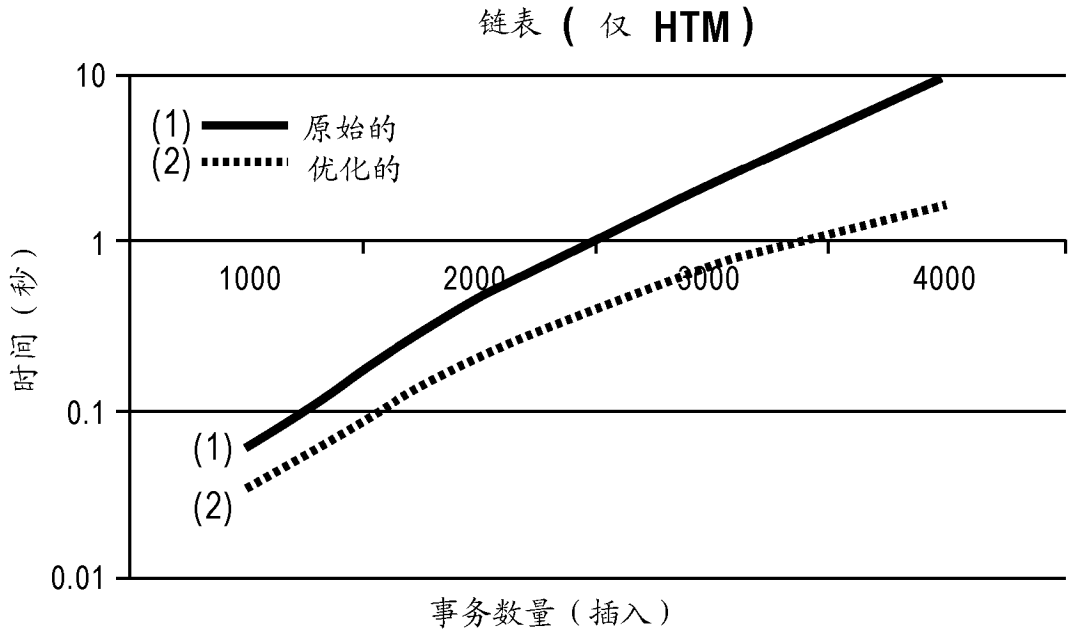

链表（仅 HTM）
10
1
0.1
0.01
1000
2000
3000
4000
(1) 原始的
(2) 优化的
时间（秒）
事务数量（插入）

图 15B

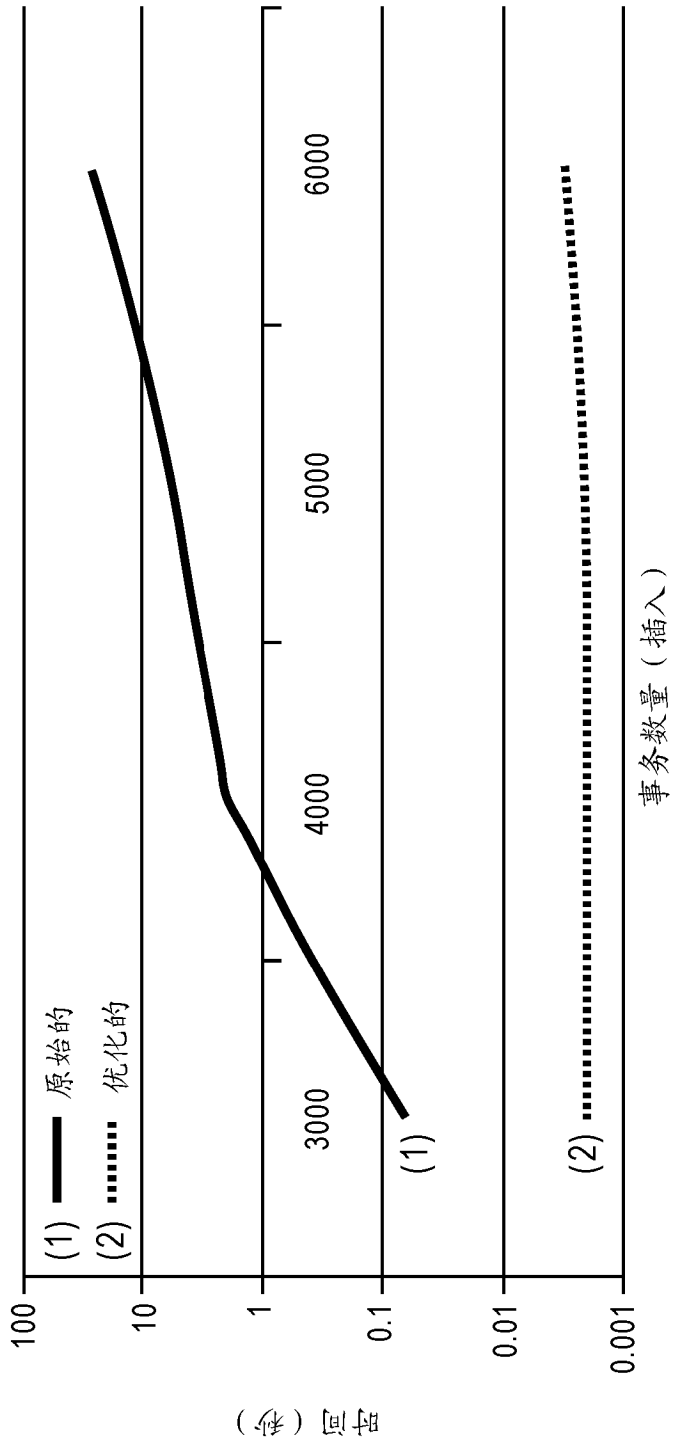

散列表（仅HTM）
(1) 原始的
(2) 优化的
100
10
1
0.1
0.01
0.001
3000
4000
5000
6000
(1)
(2)
事务数量（插入）
时间（秒）

图 15C

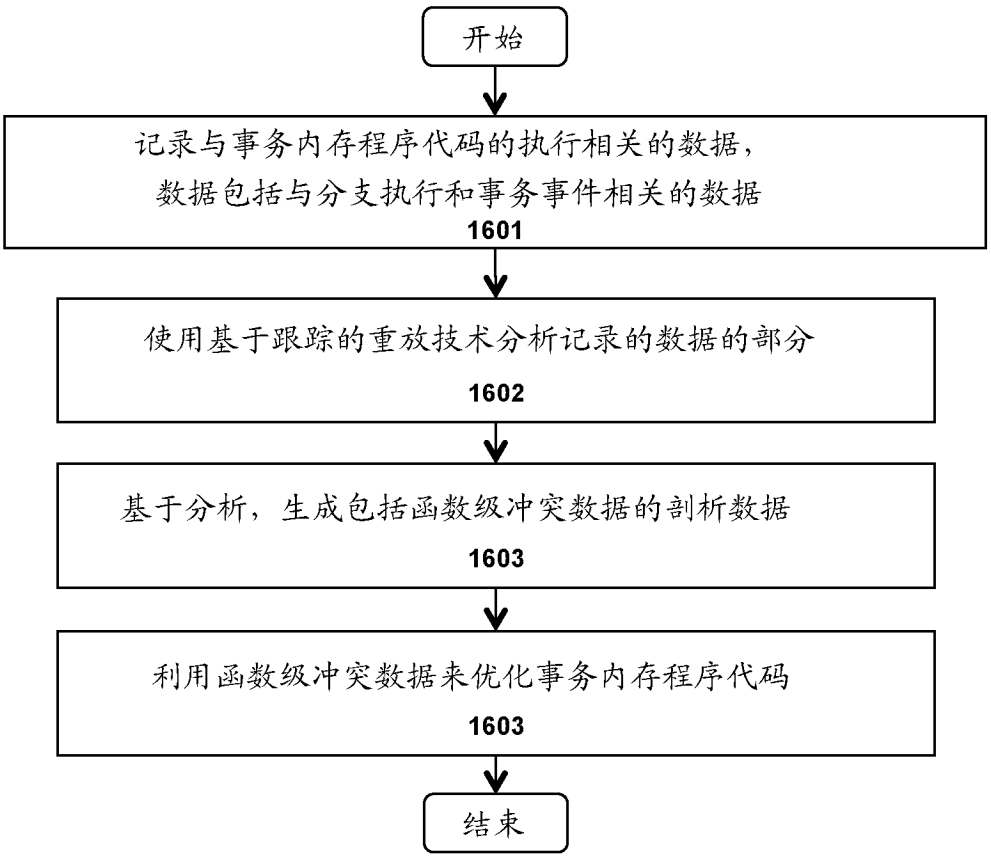

开始
记录与事务内存程序代码的执行相关的数据，
数据包括与分支执行和事务事件相关的数据
1601
使用基于跟踪的重放技术分析记录的数据的部分
1602
基于分析，生成包括函数级冲突数据的剖析数据
1603
利用函数级冲突数据来优化事务内存程序代码
1603
结束

图 16