(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 110046024 A
(43)申请公布日 2019.07.23

(21)申请号 201811534767.7

(22)申请日 2018.12.14

(30)优先权数据
62/598,788 2017.12.14 US
15/896,590 2018.02.14 US

(71)申请人 三星电子株式会社
地址 韩国京畿道

(72)发明人 M.侯赛因扎德 T.王 杨征宇
T.埃文斯

(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 钱大勇

(51)Int.Cl.
G06F 9/455(2006.01)

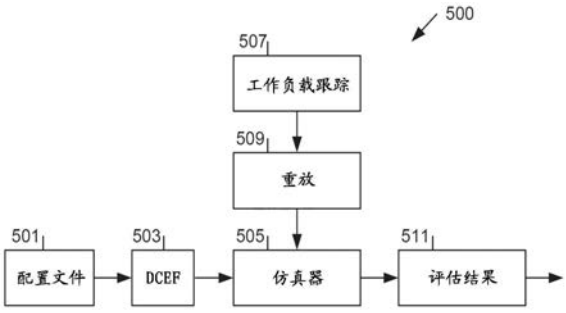
权利要求书2页 说明书11页 附图7页

(54)发明名称

用于数据中心存储评估框架仿真的方法

(57)摘要

提供了一种用于仿真数据中心的方法和一种非暂时性计算机可读记录介质,其上记录有用于执行仿真数据中心的方法的计算机程序。该方法包括在配置文件应用程序中存储要仿真的数据中心的至少一个硬件配置文件和至少一个功能描述文件;通过数据中心存储评估框架(DCEF)应用程序使用至少一个硬件配置文件和至少一个功能描述文件生成数据中心的仿真程序;和通过仿真器对DCEF应用程序生成的仿真程序执行基于流的仿真。



CN 110046024 A

1.一种仿真数据中心的方法,包括:

在配置文件应用程序中存储要仿真的数据中心的至少一个硬件配置文件和至少一个功能描述文件;

通过数据中心存储评估框架(DCEF)应用程序使用至少一个硬件配置文件和至少一个功能描述文件生成数据中心的仿真程序;和

通过仿真器对DCEF应用程序生成的仿真程序执行基于流的仿真。

2.如权利要求1所述的方法,还包括:

在DCEF应用程序中存储在设备池应用程序中构建数据中心的多个设备的模型;和

由DCEF应用程序中的仿真器程序生成器生成数据中心的仿真程序。

3.如权利要求2所述的方法,其中构建数据中心的多个设备包括中央处理单元(CPU)、包括双列直插式存储器模块(DIMM)/非易失性DIMM(NVDIMM)的存储器、硬盘驱动器(HDD)、固态驱动器(SSD)、网络接口控制器(NIC)、网络交换机、操作系统、工作负载应用程序、文件系统、驱动器、具有流和块的设备以及包括地址/数据(ADDR/DATA)总线、外围组件互连(PCI)总线、PCI Express(PCI-e)总线、串行高级技术附件(SATA)总线和智能驱动电子(IDE)总线的接口。

4.如权利要求1所述的方法,还包括由所述仿真器接收工作负载跟踪元数据和输出性能度量。

5.如权利要求4所述的方法,其中,所述性能度量包括输入/输出(I/O)性能、能量消耗、总拥有成本(TCO)、可靠性和可用性中的至少一个。

6.如权利要求5所述的方法,其中,所述I/O性能包括吞吐量、带宽和总执行时间。

7.如权利要求4所述的方法,还包括:由决策制定应用程序执行负载平衡和拓扑重组,以基于所述性能度量来改进所述数据中心的性能。

8.如权利要求1所述的方法,还包括:

将跟踪记录存储在仿真器中的跟踪文件应用程序中;

通过仿真器中的跟踪获取应用程序,从跟踪文件应用程序中逐个获取跟踪记录并生成相应的作业;

输入作业队列中的相应作业;和

通过作业分发应用程序启动流并从流的开始执行流。

9.如权利要求8所述的方法,还包括:

通过作业分发应用程序中的作业解码器解码作业以获得流标识符(ID)和元数据;

通过作业分发应用程序中的匹配器应用程序在设备池应用程序中查找相应的设备对象描述;和

初始化和操作由作业分发应用程序中的运行器应用程序从相应的设备对象描述构建的数据中心。

10.如权利要求1所述的方法,其中,所述数据中心包括多个主机,其中,所述主机是虚拟机(VM)和管理程序之一。

11.一种非暂时性计算机可读记录介质,其上记录有用于执行仿真数据中心的方法的计算机程序,该方法包括:

在配置文件应用程序中存储要仿真的数据中心的至少一个硬件配置文件和至少一个

功能描述文件;

通过数据中心存储评估框架(DCEF)应用程序使用至少一个硬件配置文件和至少一个功能描述文件生成数据中心的仿真程序;和

通过仿真器对DCEF应用程序生成的仿真程序执行基于流的仿真。

12.如权利要求11所述的非暂时性计算机可读记录介质,所述方法还包括:

在DCEF应用程序中存储在设备池应用程序中构建数据中心的多个设备的模型;和

由DCEF应用程序中的仿真器程序生成器生成数据中心的仿真程序。

13.如权利要求12所述的非暂时性计算机可读记录介质,其中构成所述数据中心的所述多个设备包括中央处理单元(CPU)、包括双列直插式存储器模块(DIMM)/非易失性DIMM(NVDIMM)的存储器、硬盘驱动器(HDD)、固态驱动器(SSD)、网络接口控制器(NIC)、网络交换机、操作系统、工作负载应用程序、文件系统、驱动器、具有流和块的设备以及包括地址/数据(ADDR/DATA)总线、外围组件互连(PCI)总线、PCI Express(PCI-e)总线、串行高级技术附件(SATA)总线和智能驱动电子(IDE)总线的接口。

14.如权利要求11所述的非暂时性计算机可读记录介质,所述方法还包括由所述仿真器接收工作负载跟踪元数据和输出性能度量。

15.如权利要求14所述的非暂时性计算机可读记录介质,其中所述性能度量包括输入/输出(I/O)性能、能量消耗、总体拥有成本(TCO)、可靠性和可用性中的至少一个。

16.如权利要求15所述的非暂时性计算机可读记录介质,其中所述I/O性能包括吞吐量、带宽和总执行时间。

17.如权利要求14所述的非暂时性计算机可读记录介质,所述方法还包括由决策制定应用程序执行负载平衡和拓扑重组,以基于所述性能度量来改进所述数据中心的性能。

18.如权利要求11所述的非暂时性计算机可读记录介质,所述方法还包括:

将跟踪记录存储在仿真器中的跟踪文件应用程序中;

通过仿真器中的跟踪获取应用程序,从跟踪文件应用程序中逐个获取跟踪记录并生成相应的作业;

输入作业队列中的相应作业;和

通过作业分发应用程序启动流并从流的开始执行流。

19.如权利要求18所述的非暂时性计算机可读记录介质,所述方法还包括:

通过作业分发应用程序中的作业解码器解码作业以获得流标识符(ID)和元数据;

通过作业分发应用程序中的匹配器应用程序在设备池应用程序中查找相应的设备对象描述;和

初始化和操作由作业分发应用程序中的运行器应用程序从相应的设备对象描述构建的数据中心。

20.如权利要求11所述的非暂时性计算机可读记录介质,其中所述数据中心包括多个主机,其中主机是虚拟机(VM)和管理程序之一。

用于数据中心存储评估框架仿真的方法

[0001] 相关申请的交叉引用

[0002] 本专利申请要求2017年12月14日在美国专利商标局提交的序列号为62/598,788的美国临时专利申请的优先权,其全部内容通过引用整体并入本文。

技术领域

[0003] 本公开一般涉及用于仿真的方法和装置,更具体地,涉及用于数据中心存储评估框架(DCEF)仿真的方法和装置。

背景技术

[0004] 组织具有运行数百个虚拟机的数十个物理主机的数据中心是昂贵且耗时的。当公司升级现有数据中心以便安装新设备(如新处理器,新兴内存技术或高端存储设备)或采用新的管理算法(如资源分配)时,这一点更为明显。但是,升级整个系统不仅成本高昂,而且还可能损害数据中心正在进行的任务并产生更多费用,而结果可能不值得付出努力。

[0005] 因此,需要一种评估和估计由数据中心存储系统的硬件和软件改变带来的性能变化(例如,改进或降级)的装置和方法,而无需物理地改变硬件和软件。

发明内容

[0006] 根据一个实施例,提供了一种仿真数据中心的方法。该方法包括在配置文件应用程序中存储要仿真的数据中心的至少一个硬件配置文件和至少一个功能描述文件;通过数据中心存储评估框架(DCEF)应用程序使用至少一个硬件配置文件和至少一个功能描述文件生成数据中心的仿真程序;和通过仿真器对DCEF应用程序生成的仿真程序执行基于流的仿真。

[0007] 提供了一种非暂时性计算机可读记录介质,其上记录有用于执行仿真数据中心的方法的计算机程序,该方法包括:在配置文件应用程序中存储要仿真的数据中心的至少一个硬件配置文件和至少一个功能描述文件;通过数据中心存储评估框架(DCEF)应用程序使用至少一个硬件配置文件和至少一个功能描述文件生成数据中心的仿真程序;和通过仿真器对DCEF应用程序生成的仿真程序执行基于流的仿真。

附图说明

[0008] 通过以下结合附图的详细描述,本公开的某些实施例的以上和其他方面,特征和优点将更加明显,其中:

[0009] 图1A是根据一个实施例的基于流的仿真的图示;

[0010] 图1B是根据一个实施例的事件驱动仿真的图示;

[0011] 图2是根据一个实施例的数据中心架构的框图;

[0012] 图3是根据一个实施例的图2的数据中心的框图;

[0013] 图4是根据一个实施例的图3的主机的框图;

[0014] 图5是根据一个实施例的DCEF工作流的框图；

[0015] 图6是根据一个实施例的图5的DCEF的框图；

[0016] 图7是根据一个实施例的仿真执行流的方法的流程图；

[0017] 图8是根据一个实施例的作业分配和执行的方法的流程图；

[0018] 图9是根据一个实施例的图8的作业分发应用程序的流程图；和

[0019] 图10是根据一个实施例的图6和图8的设备池应用的框图。

具体实施方式

[0020] 在下文中，参考附图详细描述本公开的实施例。应该注意的是，相同的元件将由相同的附图标记表示，尽管它们在不同的附图中示出。在以下描述中，仅提供诸如详细配置和组件的具体细节以帮助全面理解本公开的实施例。因此，对于本领域技术人员来说显而易见的是，在不脱离本公开的范围的情况下，可以对这里描述的实施例进行各种改变和修改。另外，为了清楚和简明，省略了对公知功能和结构的描述。以下描述的术语是考虑到本公开中的功能而定义的术语，并且可以根据用户、用户的意图或习惯而不同。因此，术语的定义应基于整个说明书中的内容来确定。

[0021] 本公开可以具有各种修改和各种实施例，其中下面参考附图详细描述实施例。然而，应该理解的是，本公开不限于这些实施例，而是包括在本公开的范围内的所有修改、等同物和替代物。

[0022] 尽管包括诸如第一、第二等序数的术语可用于描述各种元件，但结构元件不受这些术语的限制。这些术语仅用于区分一个元素与另一个元素。例如，在不脱离本公开的范围的情况下，第一结构元件可以被称为第二结构元件。类似地，第二结构元件也可以称为第一结构元件。如这里所使用的，术语“和/或”包括一个或多个相关项的任何和所有组合。

[0023] 本文使用的术语仅用于描述本公开的各种实施例，但不旨在限制本公开。除非上下文另有明确说明，否则单数形式旨在包括复数形式。在本公开中，应该理解，术语“包括”或“具有”表示特征、数量、步骤、操作、结构元件、部件或其组合的存在，并且不排除添加一个或多个其他特征、数字、步骤、操作、结构元件、部件或其组合的存在或可能性。

[0024] 除非另外定义，否则本文使用的所有术语具有与本公开所属领域的技术人员理解的含义相同的含义。诸如在通常使用的字典中定义的那些术语将被解释为具有与相关领域中的上下文含义相同的含义，并且除非在本公开中明确定义，否则不应被解释为具有理想或过度正式的含义。

[0025] 本公开涉及评估和估计由数据中心存储系统的硬件和软件改变引起的性能改变(例如，改进或降级)，而无需物理地改变硬件和软件。因此，数据中心管理员可以容易地解决包括异构存储介质(例如，固态驱动器(SSD)和硬盘驱动器(HDD))和复杂网络拓扑的数据中心中的存储资源管理和优化问题。

[0026] 根据一个实施例，公司可以在单个机器上通过基于流的仿真来评估数据中心的新的或建议的安装，而无需花费资金来物理地安装数据中心。

[0027] 根据一个实施例，本公开是基于模块的、高度封装的、可插拔的和可缩放的。

[0028] 根据一个实施例，本公开使用自动编程来基于用户硬件配置描述文件生成仿真器程序。

[0029] 根据一个实施例，本公开可以进行多种类型的性能评估，其中生成的仿真器读取工作负载样本(例如，工作负载跟踪元数据)，使用基于流的方法仿真性能，并输出许多性能度量，例如作为输入/输出(I/O)性能、能耗、总体拥有成本(TCO)、可靠性审计和可用性。

[0030] 根据一个实施例，本公开包括决策制定助手，其中决策可以基于特定工作负载模式下的不同硬件配置的性能评估结果由系统管理员或由系统自动进行以进行负载平衡(包括虚拟机磁盘(VMDK)迁移)和重新组织数据中心的拓扑以提高性能。

[0031] 图1A是基于流的仿真的图示。图1B是根据一个实施例的事件驱动仿真的图示。

[0032] 参见图1A和1B示，当前的计算机系统仿真器可以分为两大类：时序仿真器，其可以是循环驱动的或事件驱动的；以及功能仿真器。循环驱动的仿真器通常用于仿真CPU或电子系统等低级微体系结构，主要用于时序/功率测量，以非常慢的执行速度(1-1000KI PS)提供极其精确的结果。

[0033] 事件驱动的仿真器通常用于仿真高级计算机系统，其中描述和精确处理系统中的每个可能事件。事件驱动的仿真器比循环驱动的仿真器快得多。循环驱动的仿真器和事件驱动的仿真器都可以仿真并发，但是它们的严格实现不允许用户轻松仿真任意计算机系统。因此，循环驱动的仿真器和事件驱动的仿真器本身都缺乏灵活性，这可能需要很长的开发时间。另一方面，虽然功能仿真器可能仅需要短的开发时间，但功能仿真器本身不提供关于物理特性的任何信息，因为功能仿真器仅用于验证功能。

[0034] 许多时序仿真器使用事件驱动和后向模型，并且存储器元件具有历史而不是单个值。基于流的仿真利用了关键的洞察力，即在序列化时可以更好地理解复杂算法。因此，与其中事件的调用时间复杂(例如，取决于许多因素)的事件驱动方法相反，在基于流的技术中，事件驱动方法由于串行执行而是可预测的。

[0035] 基于流的仿真的目标是让架构师通过仅仿真功能来测量性能、能量消耗和功能验证。在基于流的仿真器中，仿真下的每个存储元件的内容都与时间相关联。从概念上讲，时间是可以增加或减少的全局浮点数。由于系统的状态也是时间绑定的，因此可以在调用链之后回滚一系列事件，这被称为闪回。

[0036] 在本公开中，采用基于流的仿真方法。也就是说，流被定义为连续发生的事件序列，其通过调用例程来触发，并在程序计数器从该例程返回时结束。例程是一段代码，用于描述系统中元素的功能，其被称为块。除了任务功能之外，每个块还具有延迟/功率分量，用于时序/能量评估。块可以接收作业并处理该作业。此外，块可以选择性地生成其他作业并将生成的作业发送到作业队列，类似于事件驱动的仿真，但是处于更高级别。

[0037] 基于流的技术依赖于调用栈以类似于硬件中的缓冲，使得程序员不需要非常频繁地使用作业队列。如图1A和1B所示，在事件驱动的仿真方法中，所有块调度事件以相互调用，而在基于流的仿真方法中，块可以直接相互调用。但是，仍然可以使用作业队列来调用块，块通常用于调用自身的块，这称为环回。此外，基于流的仿真可以通过将仿真系统的各个部分压缩成单个块来实现快速开发，具体取决于抽象级别。

[0038] 图2是根据一个实施例的数据中心201、203、205和207的架构200的框图。

[0039] 参见图2，架构200可以包括第一数据中心201、第二数据中心203、第三数据中心205、第四数据中心207和全球网络/因特网209。第一数据中心201、第二数据中心203、第三数据中心205、第四数据中心207各自连接到全球网络/因特网209。尽管图2中示出了四个数

据中心201、203、205和207，但是本公开不限于此。架构200可以包括任何数量的数据中心(例如，本地网络)。数据中心201、203、205和207与全球网络/因特网209之间的连接可以是无线连接、有线连接或其任何组合。

[0040] 数据中心201、203、205和207中的每一个可以具有多个主机(例如，物理主机或服务器)，如下面参考图3更详细描述的。虚拟机(VM)和管理程序(软件)可以在主机上运行。多个VM可以在管理程序上运行。

[0041] 图3是根据一个实施例的图2的数据中心201、203、205和207的框图。

[0042] 参见图3，每个数据中心201、203、205和207可以包括第一主机301、第二主机303、第三主机305、第四主机307和网络交换机309。第一主机机器301、第二主机303、第三主机305和第四主机307连接到网络交换机309。尽管图3中示出了四个主机301、303、305和307，但是本公开不限于此。数据中心201、203、205和207可包括任何数量的主机。主机和网络交换机309之间的连接可以是无线连接、有线连接或其任何组合。

[0043] 在每个主机301、303、305和307内，可以包括不同的硬件设备，例如中央处理单元(CPU)、存储器(双列直插式存储器模块(DI MM)/非易失性DI MM(NVDI MM))、HDD、SSD、网络接口控制器(NI C)和接口(例如，地址/数据(ADDR/DATA)总线、外围组件互连(PCI)总线、PCI Express(PCI-e)总线、串行高级技术附件(SATA)总线和智能驱动电子(I DE)总线)。

[0044] 图4是根据一个实施例的图3的主机301、303、305和307的框图。

[0045] 参见图4，主机301、303、305和307中的每一个可以包括ADDR/DATA总线401、PCI总线403、PCI-e总线405、SATA总线407、I DE总线409、CPU 411、DI MM/NVDI MM 413、HDD 415、SSD 417和NI C 419。CPU 411连接到ADDR/DATA总线401、PCI总线403、PCI-e总线405、SATA总线407，以及DI MM/NVDI MM 413连接到ADDR/DATA总线401。HDD 415连接到SATA总线407和I DE总线409。SSD 417连接到PCI-e总线405、SATA总线407和I DE总线409。NI C 419连接到PCI总线403和PCI-e总线405。虽然图4中示出了用于主机301、303、305和307的特定配置，但是本公开不限于此，并且其他配置是可能的。

[0046] 图5是根据一个实施例的DCEF工作流应用程序500的框图。

[0047] 参见图5，DCEF工作流应用程序500可以包括配置文件(配置文件)应用程序501、DCEF应用程序503、仿真器505、工作负载跟踪应用程序507、重放应用程序509和评估结果应用程序511。配置文件应用程序501可以向DCEF应用503提供不同的硬件和软件配置文件。即，本公开允许在多个抽象级别的不同类型的描述(例如，硬件描述和软件/功能描述)。DCEF应用程序503使用至少一个硬件配置文件和至少一个功能描述文件生成仿真器，并将其提供给仿真器505。DCEF应用程序503从设备池构建组件(包括软件)的模型，并将它们组装以建立一个仿真器。然后，工作负载跟踪应用程序507向重放应用程序509提供工作负载跟踪，并且重放应用程序509将工作负载跟踪提供给要重放的仿真器505以生成评估结果。仿真器505将评估结果提供给评估结果应用程序511。

[0048] 在一个实施例中，提供一种非暂时性计算机可读记录介质，其上记录有用于执行图5所示的仿真数据中心的方法的计算机程序。

[0049] 图6是根据一个实施例的图5的DCEF应用程序503的框图。

[0050] 参见图6，DCEF应用程序503可以包括设备池应用程序601和流仿真器(流仿真)应用程序603。设备池应用程序601向流仿真器应用程序603提供输出，其中流仿真器应用程序

603从图5中的配置文件应用程序501接收配置文件，使用至少一个硬件配置文件和至少一个功能描述文件生成仿真程序，并将仿真程序输出到仿真器505。流仿真器应用程序603使用配置文件来确定来自设备池应用程序601的哪些设备包含在仿真程序中。

[0051] 图7是根据一个实施例的仿真执行流程的方法的流程图。

[0052] 参见图7，在701处，选择跟踪记录并将启动器作业放入作业队列中。

[0053] 在702处，确定作业队列是否为空。如果作业队列不为空，则方法进行到703。如果作业队列为空，则方法进行到711。

[0054] 在703处，从作业队列中选择作业并将其提供给仿真器(例如，基于流的仿真器)以在作业上运行仿真。例如，跟踪文件中的跟踪记录被逐一提供给基于流的仿真器，其中仿真器产生相应的作业并将每个作业提供给下面将更详细描述的仿真器的作业分发应用程序(例如，图5中的仿真器505)。在一个实施例中，作业分发应用程序可以是一个作业分发应用程序。

[0055] 在705处，作业分发应用程序启动作业的流程。也就是说，对于提供给作业分发应用程序的作业队列中的每个作业，作业分发应用程序尝试启动流。

[0056] 在707处，由作业分发应用程序启动的流从其开始被执行。也就是说，对于启动的流，流程返回到其开始执行。

[0057] 在709处，确定是否需要新的跟踪。如果不需要新的跟踪，则该方法返回到702。如果需要新的跟踪，则方法进行到711。

[0058] 在711处，确定是否到达跟踪文件的末尾。如果未到达跟踪文件的末尾，则该方法返回到701。如果到达跟踪文件的末尾，则方法进行到713。

[0059] 在713处，提供评估报告(例如，输出，打印)。也就是说，当跟踪文件中的所有作业都完成时，提供评估报告。

[0060] 图8是根据一个实施例的用于作业分配和执行的方法800的流程图。作业分配和执行在仿真器(例如图5中的仿真器505)中进行。

[0061] 参见图8，方法800包括跟踪文件应用程序801、跟踪获取应用程序803、作业队列应用程序805、作业分发应用程序807和设备池应用程序809。跟踪文件应用程序801中的跟踪记录可以存储在仿真器中，并且跟踪文件应用程序801向跟踪获取应用程序803提供输出。跟踪获取应用程序803可以从跟踪文件应用程序一个接一个地获取跟踪记录并产生相应的作业。跟踪获取应用程序803向作业队列应用程序805和作业分发应用程序807提供输出。可以在作业队列应用程序805中输入相应的作业。作业队列应用程序805和设备池应用程序809各自提供输出到作业分发应用程序807。作业分配应用程序807可以启动流并从流的开始执行流。

[0062] 方法800执行由DCEF生成的仿真器。

[0063] 图9是根据一个实施例的图8的作业分发应用程序807的流程图。

[0064] 参见图9，作业分发应用程序807包括作业解码器901，运行器应用程序903和匹配器应用程序905。作业解码器901接收来自图8的作业队列应用程序805的输入，并且向运行器应用程序903和匹配器应用程序905提供输出，用于向运行器应用程序903提供输入和向匹配器应用程序905提供流ID。运行器应用程序903接收来自图8的跟踪获取应用程序803的输入，并且匹配器应用程序905从图8中的设备池应用程序809接收输入。

[0065] 作业分发应用程序807从作业队列应用程序805接收作业，并在作业解码器901中对作业进行解码，以获得作业的流标识符(ID)和元数据。然后，流ID被发送到匹配器应用程序905，并且匹配器应用程序905使用流ID在设备池应用809中查找相应的设备对象描述。设备池应用程序809包含指向包括硬件和软件组件的所有对象描述的指针，并具有数据库中系统的所有组件的信息。

[0066] 匹配器应用程序905在设备池应用程序809中查找对应的设备对象描述。从设备对象描述构建设备模型，并且用于构建的设备的运行器(例如，运行器应用程序903)初始化并操作该构建的设备模型。另外，如果需要，可以将一个或多个新作业分派到作业队列。如果硬件设备包括无限循环以使其保持活动，则作业队列应用程序805可以不为空。

[0067] 图10是根据一个实施例的图6和8的设备池应用程序的框图。

[0068] 参见图10，设备池应用程序601、809至少包括具有流和块的设备1001。在一个实施例中，设备池可以包括CPU 1003、DI MM/NVDI MM 1005、存储器总线1007、IDE1009、网络(NET)设备1011、SATA 1013、PCI 1015、PCI-e1017、SSD 1019、HDD 1021、NIC 1023、网络交换机1025、工作负载(WL)设备1027、操作系统(OS)1029、文件系统1031和驱动器应用程序1033中的至少一个。然而，设备池应用程序601、809不限于这些组件，并且可以包括其他组件。

[0069] 用于本公开的硬件描述和编程语言(HDPL)可以定义如下：

$$n \in \mathbb{R}$$

$$v \in Identifier$$

$$\oplus \in Operator$$

[0070]

$$C \in Class$$

$$P \in PrimitiveType$$

$$q \in TypeQualifier$$

$$p \in Parameter$$

$$\eta \in Specification$$

$$m \in Module ::= \text{module } m \text{ extends } m'\{\vec{d}\}$$

$$d \in Descriptor ::= \vec{p}|\vec{\eta}|\vec{\rho}|\vec{f}|\vec{c}|\vec{\alpha}|\vec{s}$$

$$\rho \in PortDe\,f ::= \text{input}\,T\,\rho[e]|\text{output}\,T\,\rho[e]|\text{inout}\,T\,\rho[e]$$

$$f \in Field ::= f \text{ from } e \text{ to } e$$

$$c \in Connection ::= \rho_1 \leftarrow \rho_2|\rho_1 \rightarrow \rho_2|\rho_1 \leftrightarrow \rho_2|\rho_1 = \rho_2|$$

$$\alpha \in Architecture ::= r|\psi$$

$$s \in Slot ::= \text{slot}\,s\,(\overrightarrow{Tv}):T\vec{\gamma}\{\vec{\omega}\}$$

$$e \in Exp ::= n|p|v|v[e]|e \oplus e|(e)|s(\vec{e})|e@f|null$$

[0071]

$$r \in StoreElement ::= \text{storage}\,r\,[e]\{r \mid T\,v\}$$

$$\psi \in Pipline ::= \text{pipeline}\,\psi\,(\overrightarrow{Tv})\,\{\vec{r}\}$$

$$\tau \in Stage ::= \text{stage}\,\tau\,(\overrightarrow{Tv}) \text{ when } e$$

$$T \in Type ::= C \mid q\,P$$

$$\gamma \in PhysChar ::= \sim\gamma(e)$$

$$\omega \in Statement ::= (\mathrm{C} + + Statement)$$

$$|\text{serial}\,\{\vec{\omega}\}$$

$$|\text{concurrent}\,\{\vec{\omega}\}$$

$$|\text{fire}\,\{\psi\}$$

[0072] 在HDPL中，合并了硬件描述语言和编程语言。利用HDPL，用户可以使用编程语言来描述硬件设备。HDPL的示例可以是用于模块描述的C++的扩展。语法如上所述。这样的HDPL使用户能够通过C++代码内的管道定义新硬件模块。其他编程语言实现是专门设想的。

[0073] HDPL语法模板如下：

```
module <name> extends <base-module>
{
parameters:
    <name> = <value>;
```

[0074]

```
specifications:
    <name> = <value>;

channels:
    input <type> <port1>[<size>], <ports2>[<size>], …;
```

```
output <type> <port1>[<size>], <ports2>[<size>], …;
inout <type> <port1>[<size>], <ports2>[<size>], …;

architecture:
    storage <name>[<size>]
    {
        …
     }

    pipeline <name>[<size>](…)
    {
        stage <name>(…) when <perform cond>;
     }

    submodule <name>(…) extends, base-module>
     {…}
```

[0075]

```
    hist <type> <name>[<size>];    // variables with history
    <type> <name>[<size>];         // regular variables

topology:
    <input/inout> ← <output/inout>;
    <output/inout> → <input/inout>;
    <inout/inout> ←→ <inout>;
    <input> = <input> ← <output/inout>;

fields:
    <name> from <value> to <value>;

implementation:
    block <name>(…); <return type>
    ~latency(…) ~active_power(…) ~passive_power(…)
    {
        …
    }
```

```
block <input/inout port>(int index, <type> data)
~latency(…) ~active_power(…) ~passive_power(…)
{
    …
}

block <pipeline stage>(…)
~latency(…) ~active_power(…) ~passive_power(…)
{
    fire <pipeline>[<index>](…);
    concurrent {
    …
    }
    serial {
    …
    }
}
```

[0077] 以上描述了用于描述通用模块的模板代码。描述可以针对硬件和软件模块二者。基于流的仿真方法和设计被用作转换器(transpiler),其在HDPL中获取代码并生成其等效的C++代码以实现所描述系统的基于流的仿真。用户只需关心每个块的功能和等待时间/功率分量(可能是估计)。整体时序测量由转换器进行。基于流的方法用于使从HDPL到C++的转换更容易,并使循环精确仿真更快。然而,本公开不限于仅使用基于流的方法。也可以使用事件驱动和循环驱动的方法。在用户仅对以更快的方式调查功能感兴趣的情况下,可以仅考虑功能仿真。

[0078] 模块描述可以包括如下的多个范围指定符(scope specifier):

[0079] 范围指定符“parameter”用于参数化模块。在整个模块描述中,参数可以在任何地方使用。参数类似于C++中的模板类,但更专注于HDPL。例如,参数可以改变模块的功能或改变成员阵列的维度。

[0080] 范围指定符“specification”用于重新配置基本模块,其中可以使用继承器模块的该范围指定符来修改基本模块的参数。

[0081] 范围指定符“channel”用于将通道实现为由一组访问函数和缓冲器组成的C++类。模块可以具有三种类型的通道以提供各种电导率:输入、输出和入出。

[0082] 范围指定符“input”用于指定仅可以连接到另一模块的输出通道的连接。可以在模块描述范围内收听、读取和刷新输入通道。

[0083] 范围指定符“out put”是可以连接到另一模块的输入通道的输出端口。输出端口可以连接到多个输入通道。

[0084] 范围指定符“inout”是可以从两侧读取和写入的全双工通道。入出可以仅连接到另一个模块的输入、输出和入出通道。

[0085] 范围指定符“architecture”包含模块的体系结构声明，其可以包括对象、存储、流水线、子模块和变量的多个实例。

[0086] 范围指定符“storage”描述了设备的存储空间的逻辑结构，其可以形成诸如hdd.sector.page.line的访问树。存储结构类似于C中的常规结构，但是存储结构实现了复杂的类，其使用分层或截止算法存储在哈希表内修改的历史。

[0087] 范围指定符“pipeline”是模块功能的主要描述。模块在一个或多个描述的管道中执行其任务。管道包括用于引用管道的名称、用于启用超级管道的维度以及一些可能的输入。为了描述管道，其阶段应列在管道主体内。每个阶段是函数调用，下面参考模块的“实现”范围对其进行描述。此外，管道可以是有条件的，这表示仅当存在特定条件时才可以执行流水线阶段。

[0088] 范围指定符“submodule”类似于定义新模块，但子模块仅限于在子模块的父模块内使用。

[0089] 范围指定符“variable”类似于C++类结构。可以在模块内实例化一些变量、数组、结构(或构造)、类和模块。变量可以由“hist”指定符声明，该指定符通过为变量分配修改历史来使其时间绑定。可以使用getter和setter方法访问这些变量。请注意，hist也可以在存储结构中使用。

[0090] 范围指定符“topology”用于描述子模块的连通性。

[0091] 范围指定符“field”描述地址字段。稍后可以以addr@field的格式使用字段，该字段返回地址的特定位范围。

[0092] 范围指定符“implementation”描述了所有函数主体。此外，管道阶段可以用在体系结构范围中。以块的形式描述功能，其包括名称、可能的输入向量、可能的输出类型，诸如等待时间和功率的物理特性，以及函数主体。主体是C++代码，其中包含一些用于时序仿真的附加关键字:fire，wait，serial和concurrent。

[0093] 范围指定符“fire”将作业放入仿真器作业队列中以启动管道。

[0094] 范围指定符“wait”用于调用另一个块并使呼叫者忙。

[0095] 范围指定符“serial”是使得其内部的所有内容以串行或顺序方式运行的代码块。

[0096] 范围指定符“concurrent”是使得其内部的所有内容同时运行的代码块。

[0097] 本公开的应用程序基于需要而变化。总的来说，仿真器在性能、能量和可靠性方面生成完整的报告。

[0098] 术语“性能”表示对整个系统的性能的估计以及运行数据中心的给定工作负载的准确结果的报告。性能还可以包括吞吐量、带宽和总执行时间。

[0099] 术语“吞吐量”基于用户的要求报告系统的每个部分的每秒输入/输出(I/O)操作的数量(IOPS)。

[0100] 术语“带宽”是在仿真完成之后包括存储和网络设备的设备的精确带宽的报告。

[0101] 术语“总执行时间”是每个工作负荷的总执行时间的估计，这可能有助于估计成本。

[0102] 术语“能量”是在完成每个物理设备的能量消耗的仿真时的完整报告，以及整个数

据中心的估计。

[0103] 术语“可靠性”表示仿真器可以接收每个设备的可靠性模型并估计每个设备的寿命、故障率、丢弃率、平均故障时间、恢复、修复、平均故障间隔时间和平均值下降时间。

[0104] 基于输出报告，可以计算总拥有成本、可用性以及若干其他测量和估计。

[0105] 尽管已经在本公开的详细描述中描述了本公开的某些实施例，但是在不脱离本公开的范围的情况下，可以以各种形式修改本公开。因此，本公开的范围不仅仅基于所描述的实施例来确定，而是基于所附权利要求及其等同物来确定。

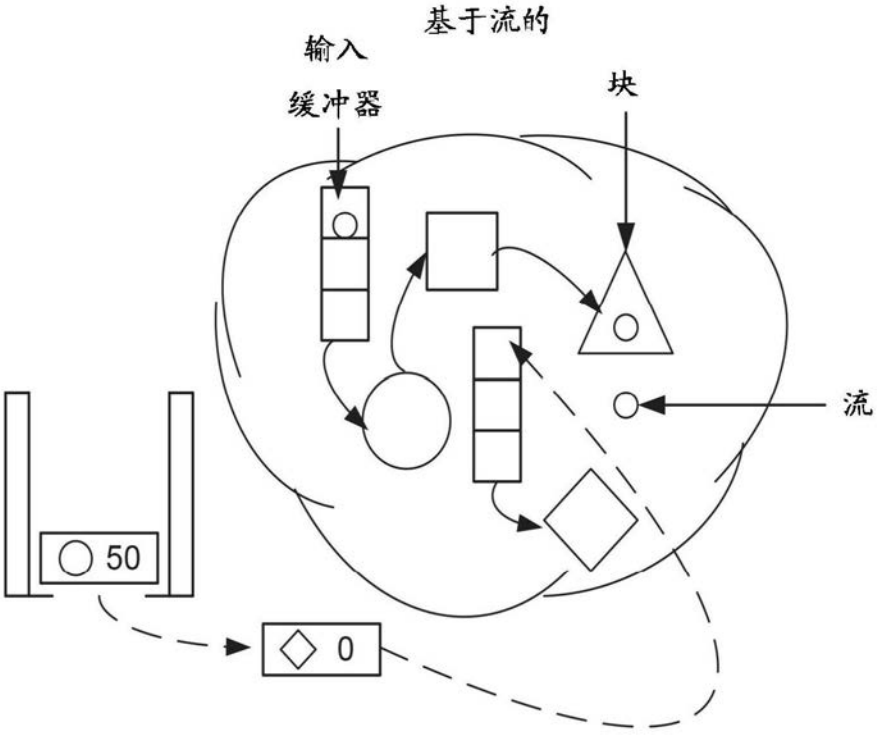


图1A

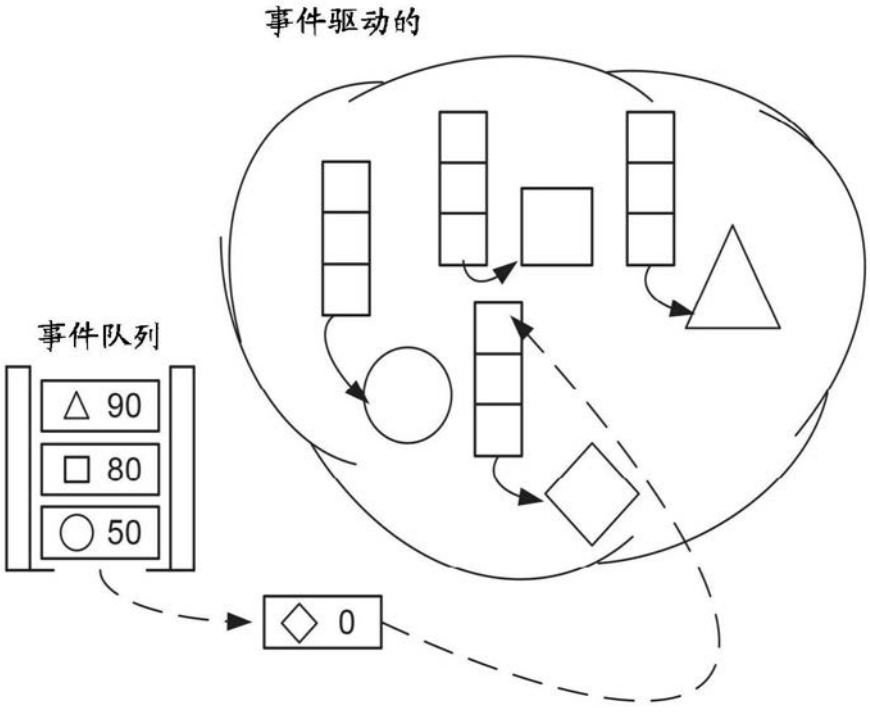


图1B

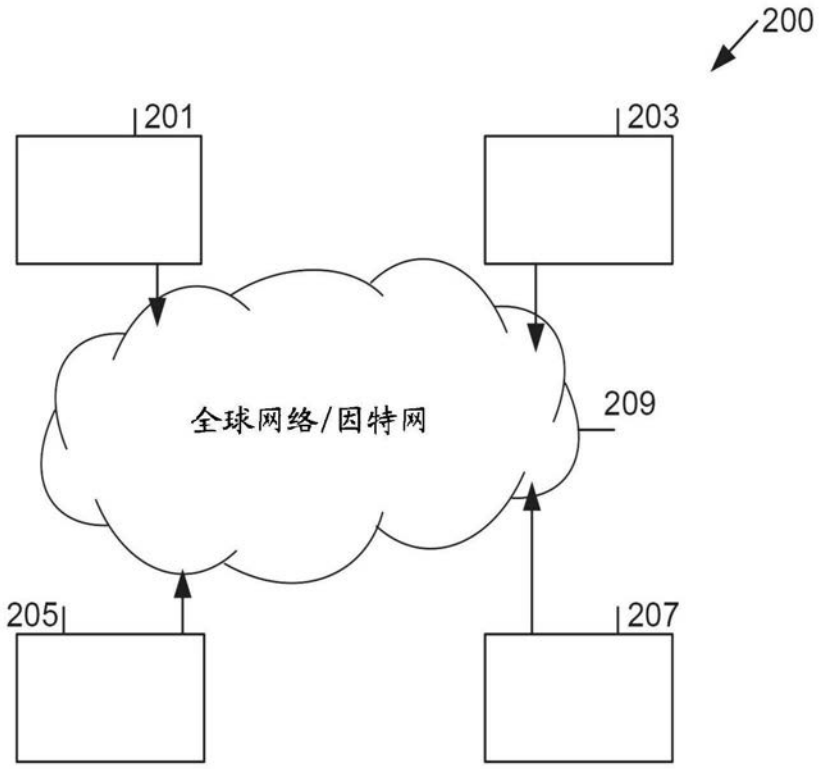

200
201
203
全球网络/因特网
209
205
207

图2

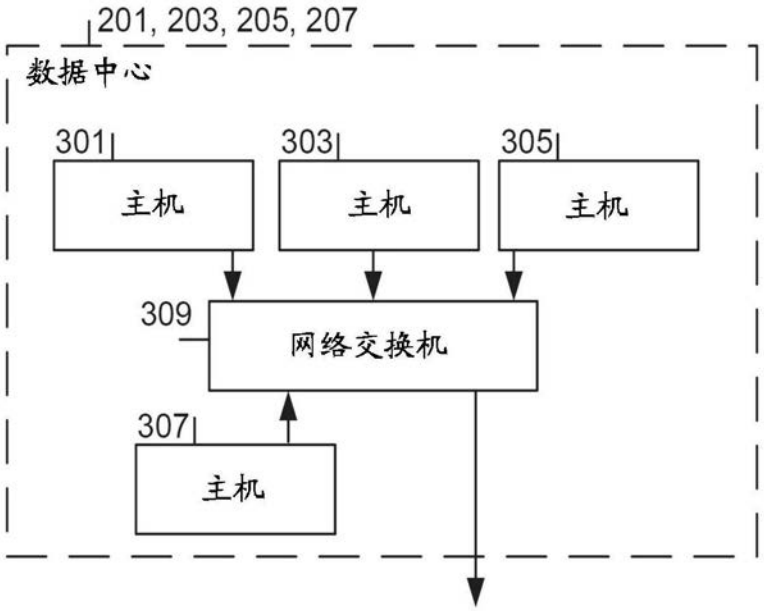

201, 203, 205, 207
数据中心
301
主机
303
主机
305
主机
309
网络交换机
307
主机

图3

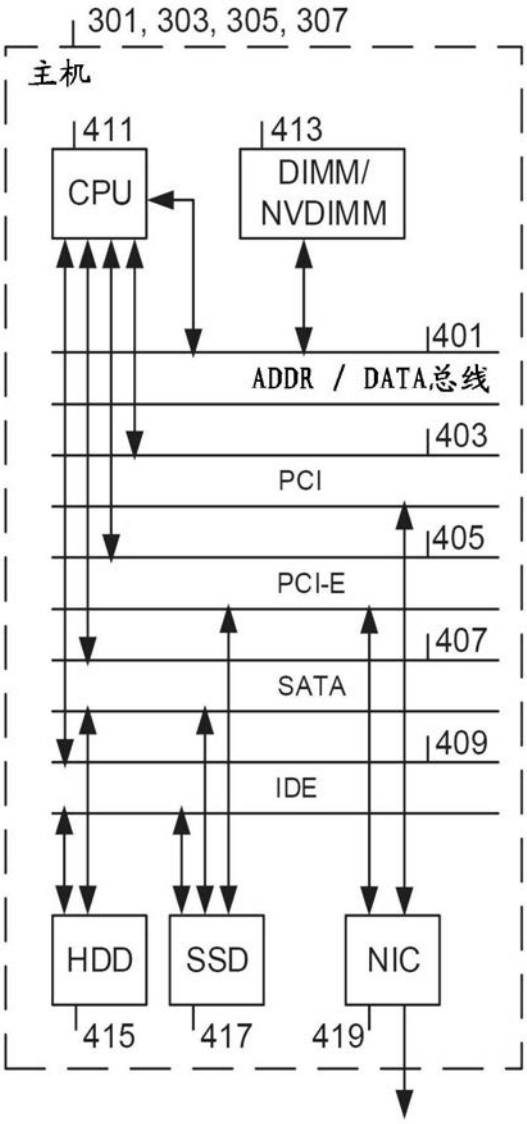

301, 303, 305, 307
主机
411
CPU
413
DIMM/
NVDIMM
401
ADDR / DATA总线
403
PCI
405
PCI-E
407
SATA
409
IDE
HDD
SSD
NIC
415
417
419

图4

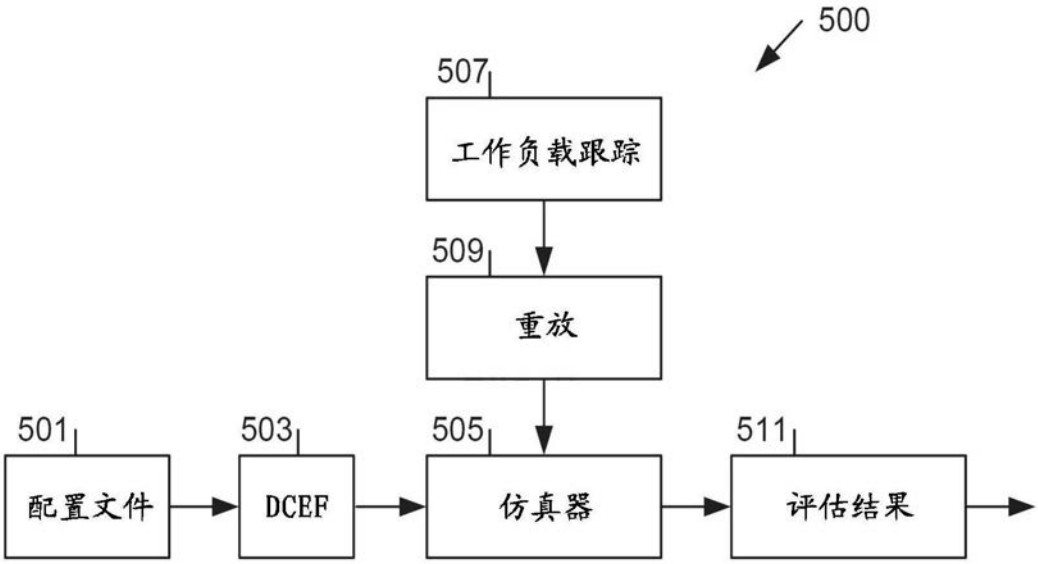

500
507
工作负载跟踪
509
重放
501
配置文件
503
DCEF
505
仿真器
511
评估结果

图5

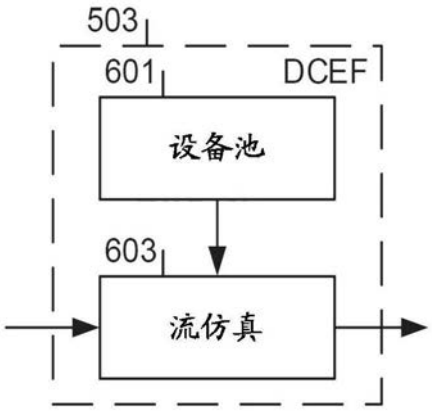

503
601
DCEF
设备池
603
流仿真

图6

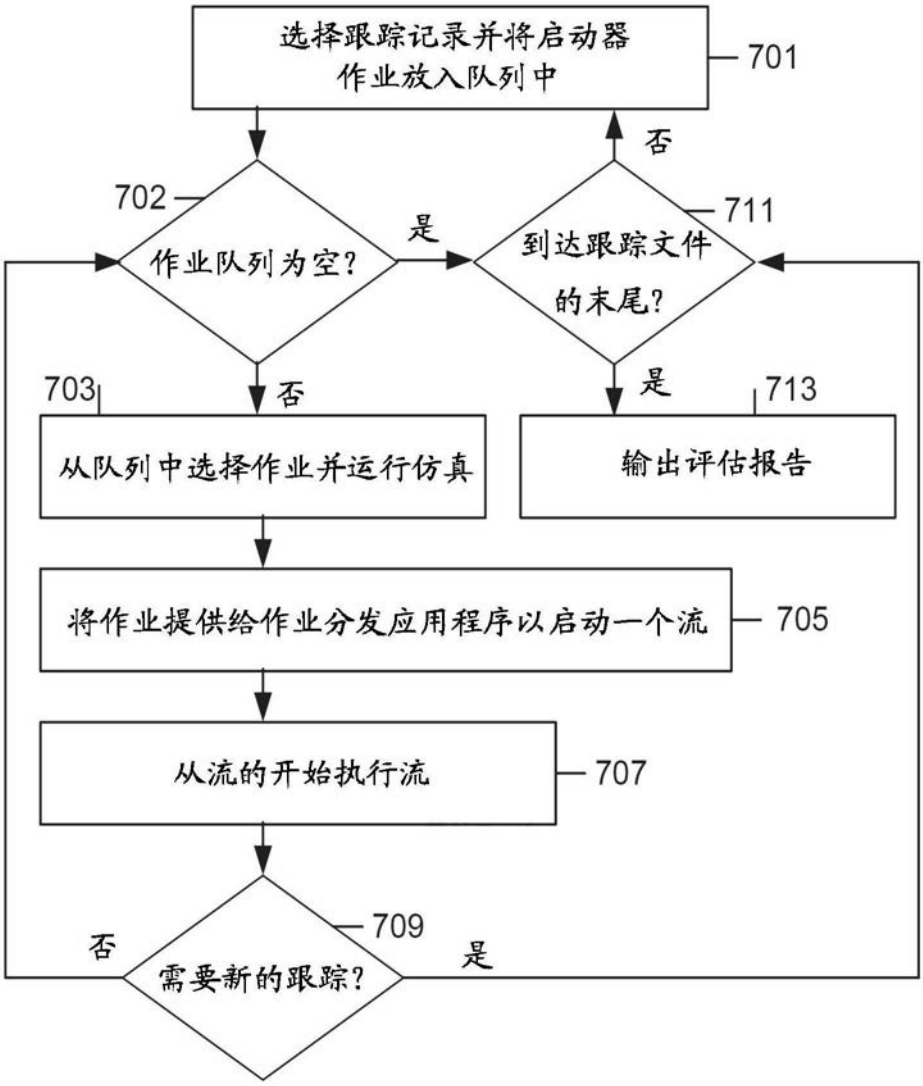

选择跟踪记录并将启动器作业放入队列中
701
否
702
作业队列为空?
是
711
到达跟踪文件的末尾?
703
否
是
713
从队列中选择作业并运行仿真
输出评估报告
将作业提供给作业分发应用程序以启动一个流
705
从流的开始执行流
707
709
否
需要新的跟踪?
是

图7

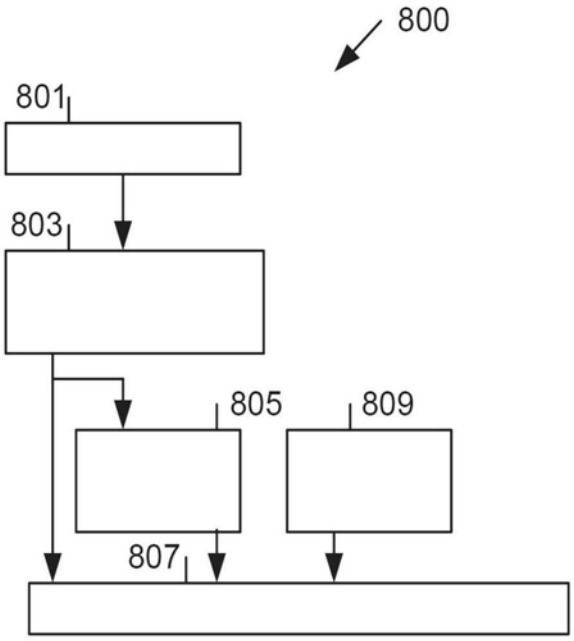

800
801
803
805
809
807

图8

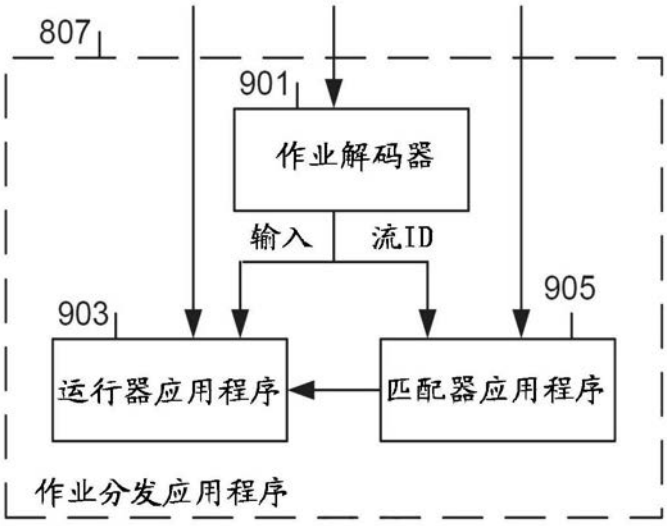

807
901
作业解码器
输入
流ID
905
903
运行器应用程序
匹配器应用程序
作业分发应用程序

图9

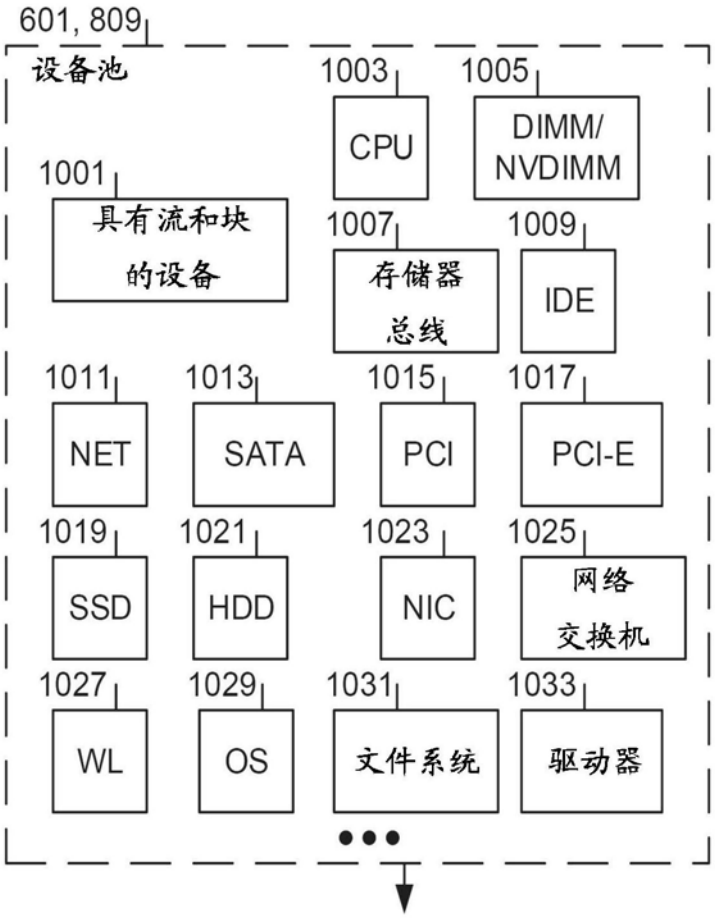

601, 809
设备池
1003
CPU
1005
DIMM/
NVDIMM
1001
具有流和块
的设备
1007
存储器
总线
1009
IDE
1011
NET
1013
SATA
1015
PCI
1017
PCI-E
1019
SSD
1021
HDD
1023
NIC
1025
网络
交换机
1027
WL
1029
OS
1031
文件系统
1033
驱动器

图10