(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 102110012 A
(43)申请公布日 2011.06.29

(21)申请号 201010618074.3

(22)申请日 2010.12.22

(30)优先权数据
12/645,568 2009.12.23 US

(71)申请人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 C·J·科迈克 N·杜卡 M·伯罗斯
S·A·特金

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 姬利永

(51)Int.Cl.
G06F 9/455(2006.01)
G06F 9/46(2006.01)

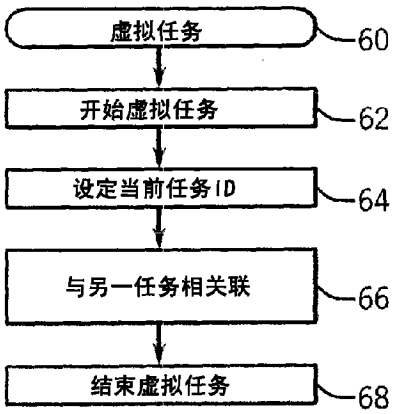
权利要求书 2 页 说明书 5 页 附图 4 页

(54)发明名称

用于使用虚拟任务对多核处理器进行性能分析的关系建模

(57)摘要

本发明提供一种使用虚拟任务对多核处理器进行性能分析的关系建模方法。关系模型可被用于为多核处理器中的多个线程的每一个编码原语。原语可包括任务和参数，诸如缓冲器。类似于设定渲染目标的隐式创建任务可通过使那些隐式创建任务与实际编码任务相关联来可视化。



CN 102110012 A

1. 一种方法，包括：

接收在多核处理器中隐式创建的任务的标识；以及

使所述隐式创建的任务与实际编码任务相关联。

2. 如权利要求 1 所述的方法，其特征在于，包括提供实际编码和隐式创建的任务之间的关系的可视化。

3. 如权利要求 1 所述的方法，其特征在于，包括指示所述任务之间的关系。

4. 如权利要求 3 所述的方法，其特征在于，指示所述任务之间的关系包括指示所述隐式创建任务的持续时间。

5. 如权利要求 3 所述的方法，其特征在于，包括提供所述关系的可视化。

6. 如权利要求 1 所述的方法，其特征在于，包括使用实际编码任务的持续时间来表示所述隐式创建任务的持续时间。

7. 如权利要求 6 所述的方法，其特征在于，包括使用一个以上实际编码任务的持续时间来表示所述隐式创建任务的持续时间。

8. 如权利要求 1 所述的方法，其特征在于，包括接收设定渲染目标的标识作为所述隐式创建任务。

9. 如权利要求 8 所述的方法，其特征在于，包括使所述设定渲染目标与至少一个绘图调用相关联。

10. 如权利要求 9 所述的方法，其特征在于，包括显示所述设定渲染目标作为所述绘图调用的子。

11. 一种存储指令的计算机可读介质，所述指令由计算机执行以：

接收在多核处理器中隐式创建的任务的标识；以及

使所述隐式创建任务与实际编码任务相关联。

12. 如权利要求 11 所述的介质，其特征在于，进一步存储用于提供所述实际编码和隐式创建的任务之间的关系的可视化的指令。

13. 如权利要求 11 所述的介质，其特征在于，进一步存储用于指示所述任务之间的关系的指令。

14. 如权利要求 13 所述的介质，其特征在于，进一步存储用于指示所述隐式创建任务的持续时间的指令。

15. 如权利要求 13 所述的介质，其特征在于，进一步存储用于提供所述关系的可视化的指令。

16. 如权利要求 11 所述的介质，其特征在于，进一步存储用于使用所述实际编码任务的持续时间来表示所述隐式创建任务的持续时间的指令。

17. 如权利要求 16 所述的介质，其特征在于，进一步存储用于使用一个以上实际编码任务的持续时间来表示所述隐式创建任务的持续时间的指令。

18. 如权利要求 11 所述的介质，其特征在于，进一步存储用于接收设定渲染目标的标识作为所述隐式创建任务的指令。

19. 如权利要求 18 所述的介质，其特征在于，进一步存储用于使所述设定渲染目标与至少一个绘图调用相关联的指令。

20. 如权利要求 19 所述的介质，其特征在于，进一步存储用于显示所述设定渲染目标

作为所述绘图调用的子类的指令。

21. 一种装置，包括：

多核处理器；以及

耦合到所述处理器的存储器，所述存储器存储用于接收多核处理器中隐式创建任务的标识并使所述隐式创建任务与实际编码任务相关联的指令。

22. 如权利要求 21 所述的装置，其特征在于，所述存储器存储用于提供所述实际编码和隐式创建的任务之间的关系的可视化的指令。

23. 如权利要求 21 所述的装置，其特征在于，所述装置用于指示所述任务之间的关系。

24. 如权利要求 21 所述的装置，其特征在于，所述装置是图形处理器。

25. 如权利要求 24 所述的装置，其特征在于，所述装置是单指令多数据多核处理器。

26. 如权利要求 21 所述的装置，其特征在于，所述装置用于指示所述隐式创建任务的持续时间。

27. 如权利要求 26 所述的装置，其特征在于，所述装置用于提供所述持续时间的可视化。

28. 如权利要求 27 所述的装置，其特征在于，所述装置用于使用所述实际编码任务的持续时间来表示所述隐式创建任务的持续时间。

29. 如权利要求 28 所述的装置，其特征在于，所述装置用于使用一个以上实际编码任务的持续时间来表示所述隐式创建任务的持续时间。

用于使用虚拟任务对多核处理器进行性能分析的关系建模

背景技术

[0001] 本发明一般涉及多核处理器，具体涉及多核机器的性能分析。

[0002] 现代的通用和图形处理器可包括一个或多个核。这些处理器可运行大量的线程。因此，在给定任务量以及可能运行的不同线程的数量的情况下，分析处理器的性能可能涉及复杂的任务。

[0003] 通常，痕迹 (trace) 是在处理器上运行的任务之间的时间顺序的图形描述。基于软件的痕迹分析使软件设计者能够理解任务间的操作顺序。

[0004] 然而，多核处理器可能需要更复杂的分析。

[0005] 附图简述

[0006] 图 1 是本发明的一个实施例的示意图；

[0007] 图 2 是描绘图 1 所示的实施例所使用的程序的流程图；

[0008] 图 3 是根据本发明的一个实施例的基于任务的关系模型的可视化；

[0009] 图 4 是根据本发明的一个实施例产生的父 / 子关系的可视化；

[0010] 图 5 是根据本发明的另一实施例的依赖关系的可视化；以及

[0011] 图 6 是根据一个实施例的虚拟任务可视化的描绘图；

[0012] 图 7 是根据一个实施例的另一虚拟任务可视化的描绘图；以及

[0013] 图 8 是一个实施例的流程图。

[0014] 详细描述

[0015] 根据本发明的一些实施例，代码组块调用任务之间的除时间顺序之外的关系可以被可视化。任务是用于调度和执行的任何普通工作单元。它可以是具有开始和结束的代码的任何部分。它的持续时间可被定义为用于执行该任务的循环数量。

[0016] 痕迹是任务之间的连接。关系模型根据父 - 子、同属、依赖性以及生产者和消费者给出任务之间的关系。也可使用其他关系。

[0017] 第一任务和由第一任务产生的第二任务之间存在父 / 子关系。当第一任务依赖第二任务来执行时，第一和第二任务存在依赖关系。生产者 / 消费者关系意味着第一任务创建数据并将其放置在缓冲器中，然后第二任务使用来自缓冲器中的该数据。

[0018] 原语是关系一方的任何实体。任务是一种类型的原语。另一种原语被称为参数。参数可以是缓冲器、名 - 值对、字符串、或与任务有关系的任何类型的标准数据类型或结构。也可使用其他原语。

[0019] 参照图 1，性能分析工具 10 可包括控件 12，该控件 12 可以是处理器。该处理器可以是多核处理器。在一个实施例中，该处理器可以是图形处理器，而在一个实施例中，该处理器可以是单指令多数据 (SIMD) 多核处理器。控件 12 被耦合到存储器 14，该存储器 14 可储存该工具的图形用户界面 (GUI)16 或前端、编码原语和原语之间的关系的多个序列或应用编程接口 (API)18、以及提供可用功能的工具箱的库 20。该控件可耦合到输入 / 输出 22 以允许用户输入信息以及接收输出。显示器 24 可用于使包括任务的原语之间的关系可视化。

[0020] 性能分析工具10可被软件开发人员用于通过了解原语（诸如在该软件内的任务）之间的关系来提高他们的软件性能。通过理解这些关系，软件开发人员可了解如何提高软件的性能。

[0021] 一般而言，软件开发人员开发两个版本的代码。代码的一个版本是仅执行需要的功能（诸如游戏）的代码。代码的另一版本内包括API 18，在一些实施例中，API 18创建原语关系的可视化（但是，在其他实施例中，在没有可视化的情况下二进制输出可被用于进一步分析）。具有API版本的代码被称为操纵代码。将操纵代码流送到前端图形用户界面16使设计者能够看到代码的进展。它显示了在代码中运行的线程、在线程内的任务、以及最重要的那些任务之间的功能关系。

[0022] 参照图2，通过应用编程接口18实现的序列由接收线程选择开始，如框22所示。接着，如框24所示，选择原语标识符。在框26，该标识符被登记。而后，在框28，向原语分配标识符。

[0023] 此时，设计者于是输入选定原语与其他原语之间的关系。所选原语被称为“这个”原语，而与这个原语有关系的原语被称为“那些”原语。

[0024] 在一些实施例中，序列18自动隐含超出那些通过设计者所输入的一些关系。举例而言，如果第一任务是第二任务的父，则意味着这第二任务是该父的子。同样，如果第一任务与第二任务相关并且第二任务与第三任务相关，则意味着该第一任务与第三任务也相关并隐含该关系的本质。这可被称为传递关系或其他关系所隐含的关系。在一些实施例中，传递关系的使用减少了用户数据输入的负担。此外，可能有一个或N个扇出关系。举例而言，在一些实施例中，单个父可产生任何数量的子，且所有那些子不需要另行编码。

[0025] 此外，序列18向各个原语标识符提供时间戳。它也提供任何任务开始与结束的时间作为输出。因此，在一个实施例中，它可将一连串的线程显示为图表中的行。每个行可包括线程中任务的序列。此外，嵌套任务也可通过可视化表示，诸如将嵌套任务显示为延伸出在其中执行该嵌套任务的任务。用户可点击特定任务，且响应于该任务选择，可以图形化地显示其关系。

[0026] 相对基于时间的跟踪系统，系统10可以是基于关系的。任务和不同缓冲器间的时间关系没有任务之间的关系重要。在一些实施例中，示出了任务内的关系，甚至在一些实施例中这些任务无相对时序。因此，在该线程内，任务的时间顺序可显示，但在一些实施例中，不同线程中的任务的时间顺序可能不显示。在一些实施例中，相对于时间关系，功能关系可以被可视化。功能关系是任务之间的除时间顺序之外的任何关系。

[0027] 在一个实施例中，各个任务可以由矩形表示，矩形在x方向的长度是执行该功能需要的时长的函数。在一些实施例中，可使用鼠标点击和拖动的技术选择多个任务以显示那些选定任务和任何其他任务之间的关系。

[0028] 因此，作为简单示例，参照图3，存在四个原语，包括三个任务36、38和42以及一个缓冲器40形式的参数。箭头44、52、50以及48示出了这些任务之间的关系。例如，顶点任务38可将其结果放置在缓冲器40中，并且那些结果可由任务42使用。因此，任务38是生产者而任务42是消费者。由于顶点任务38由绘图任务36创建，因此箭头52表示该顶点任务是绘图任务36的子。同样，像素任务42依赖于顶点任务38执行，因此箭头46表示该依赖性。因此，在一些情况下，任务之间可以有多重关系。此外，也存在多个对象类型。

[0029] 回到图 2，在框 28，在向原语分配标识符后，接着输入这个原语的关系，如框 30 所示。虽然在此描述了手动技术，但也可构想自动技术，在自动技术中代码分析导致在无需用户干预的情况下就可自动输入关系。

[0030] 而后，在菱形 32 中，检查确定是否还有原语需要编码。如果有，则该流程重复。否则，菱形 34 处的检查确定是否还有线程需要编码。如果有，则该流程重复。否则，该流程结束。

[0031] 图 4 示出了父 / 子关系的可视化示例。在一个实施例中，可视化可以是具有鼠标可选任务或原语描述的图形用户界面。在该示例中，仅示出四个线程。对于每个线程，任务由矩形表示。

[0032] 在这种情况下，用户已选择了第二线程中的父任务 B。作为响应，父 / 子关系的可视化自动生成。换句话说，箭头从任务 B 延伸到第三线程中的一连串任务 C，任务 C 是任务 B 的子。此外，箭头从第三线程中的任务 C 延伸至第四线程中的任务 C，该任务 C 是父任务 B 的孙。

[0033] 图 5 是依赖关系的可视化。图 5 对应于图 4，除了现在用户选择的任务 B 依赖于任务 A、C、D 和 E。即，任务 B 依赖来自任务 A 的输入，如由标记为“传入依赖性”的箭头所表示。任务 C、D 和 E 依赖标记为“传出依赖性”的任务 B 的输出。因此，可视化显示了不同线程的任务之间的功能关系。

[0034] 在一些诸如渲染通道 (rendering pass) 的图形应用中，在此之前所描述的技术不能启用任务可视化。然而，渲染通道可以是由程序执行的操作的有用度量，且可以是用于在应用开发人员水平进行概况分析的理想工具。

[0035] 因为图形应用编程接口不包含实际上表示渲染通道的显函数，由于渲染通道在一连串的 API 调用中隐式创建，所以它不能使用在此之前所描述的技术来编码。因此，这样隐式创建的任务可利用虚拟任务来可视化。作为更具体的示例，以下调用示例可在 Direct X API 中进行：

[0036] SetRenderTarget(1)

[0037] Draw(draw_args_1)

[0038] Draw(draw_args_2)

[0039] SetRenderTarget(2)

[0040] Draw(draw_args_3)

[0041] 在该代码中发生的是绘图 1 和绘图 2 是第一渲染目标操作的部分，而绘图 3 是第二渲染目标操作的部分。该操作在硬件中执行，且仅造成操作本身的命令是绘图命令。设置渲染目标的命令仅仅是状态设定，而不产生操作本身。为解决这个问题，在一些实施例中可使用虚拟任务。虚拟任务可在设置渲染目标调用中被创建：

[0042]

```
SetRenderTarget(x) {
    TAL_BeginVirtualTask("RenderTarget");
    TAL_SetCurrentTaskID(x->renderTargetID);
    TAL_EndVirtualTask()
}
```

[0043] 在常规任务的情况下，由于任务的持续时间可能是感兴趣的，所以开始和结束调用的时间戳被记录。然而，在以上设定渲染目标的示例中，实际任务一完成就结束；相反，它是稍后完成的真实操作的占位符。

[0044] 为将实际持续时间给予虚拟任务，其他非隐式创建的或实际编码任务可能与虚拟任务相关联。举例而言，当创建一绘图调用任务时，该绘图调用任务是实际虚拟任务的子。就该虚拟任务的概念而言，设定渲染目标的持续时间可被给予一持续时间，从而呈现给用户渲染目标虚拟任务的列表并基于它们所有的相关任务计算它们的持续时间。

[0045] 然后，通过利用虚拟任务的子任务中的最小的和最大的子任务，虚拟任务可在时间线上被可视化，如图 6 中所示。根据父 - 子关系，可产生如图 6 所示的可视化。举例而言，渲染目标 1 的基本任务可被绘制成绘图 1 和绘图 2 的子，而渲染目标 2 可被示为绘图 3 的子。因而，渲染目标 1 和渲染目标 2 获取一持续时间，即使该持续时间基本是相关绘图调用的持续时间，诸如设定渲染目标 1 情况下的绘图 1 和绘图 2 以及设定渲染目标 2 情况下的绘图 3。

[0046] 虚拟任务也可用于显示更多有关复杂任务的信息，以使在没有详细信息的情况下完成基本概况分析。在渲染目标示例中，仅虚拟任务（渲染目标 2、渲染目标 3）可以在应用运行中显示，如图 7 所示。在大量线程的情况下，在有些情况下，图 7 可比图 6 所示描述得更清楚，。例如，在 128 个不同线程的情况下，它们各个本身包含几十万个任务，设定渲染目标 2 执行最长，而设定渲染目标 3 可能依赖于设定渲染目标 2 的完成。这一见解可使设计者能够或者停止概况分析而立刻采取行动，也许通过允许设定渲染目标 2 和 3 共同执行，或者通过优化设定渲染目标 2，既然已知这是耗费时间的。

[0047] 参照图 8，用于输入虚拟任务的序列 60 可以是一个模块或 API 18 的一部分，如图 2 所示。最初，虚拟任务开始，如框 62 所示，如同它是一个真实的任务。而后，也如同该虚拟任务是真实任务，在框 64 设定当前任务标识符。接着，在框 66，该任务与另一任务相关联。例如，设定渲染目标虚拟任务可与一个或多个绘图调用相关联，如例如图 6 所示。该关联提供虚拟任务持续时间。最后，虚拟任务结束（框 68）。

[0048] 简而言之，虚拟任务可用于为复杂、高度并行软件建立概况分析工具。没有它们，概况分析工具不可能显示得比执行 API 的机制更多，这在一些情况下会妨碍获得简单且关键的性能。在一些实施例中，利用虚拟任务，得到一抽象允许这样的问题从开始时更清楚，从而实现更高产以及高效的自上而下的概况分析。

[0049] 本文中所描述的图形处理技术可在各种硬件体系结构中实现。例如，图形功能可被集成在芯片组中。替代地，可使用分立的图形处理器。作为另一实施例，图形功能可由包括多核处理器的通用处理器实现。

[0050] 在本说明书通篇中对“一个实施例”或“一实施例”的引用意味着结合该实施例描

述的特定特征、结构或特性包括在本发明包含的至少一个实现中。因此，短语“一个实施例”或“在一实施例中”的出现不一定指代同一实施例。此外，特定特征、结构或特性可按照与所说明的特定实施例不同的其他适当形式来创立，而且所有此类形式可包含在本申请的权利要求中。

[0051] 虽然已经关于有限个实施例描述了本发明，但本领域技术人员将会理解从中得出的多种修改和变化。所附权利要求旨在覆盖落入本发明的真实精神和范围中的所有这些修改和变化。

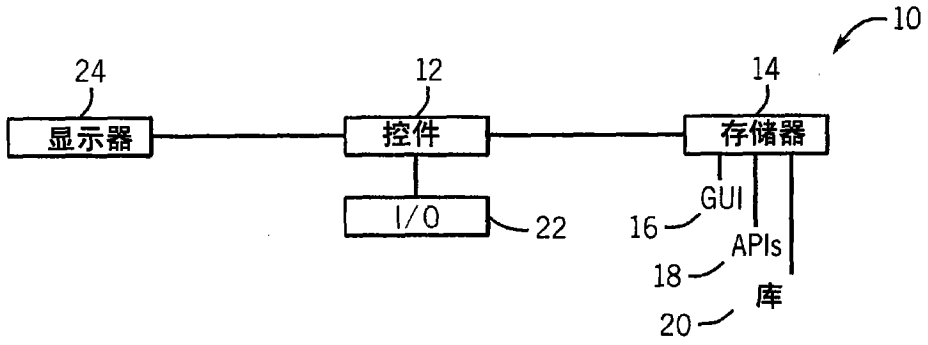

10
24
显示器
12
控件
14
存储器
I/O
22
GUI
16
APIs
18
库
20

图 1

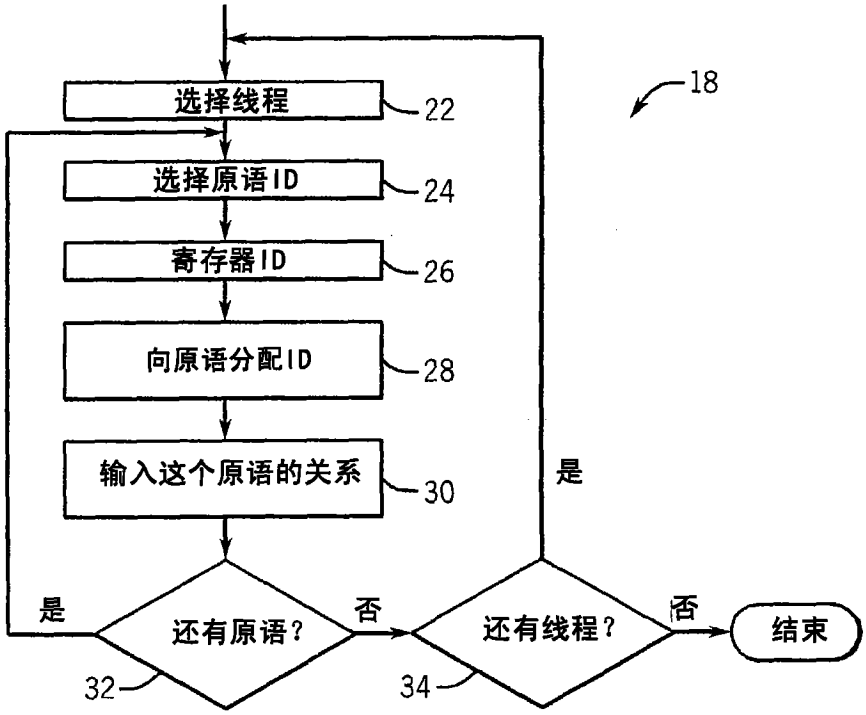

18
选择线程
22
选择原语ID
24
寄存器ID
26
向原语分配ID
28
输入这个原语的关系
30
是
是
还有原语?
否
还有线程?
否
结束
32
34

图 2

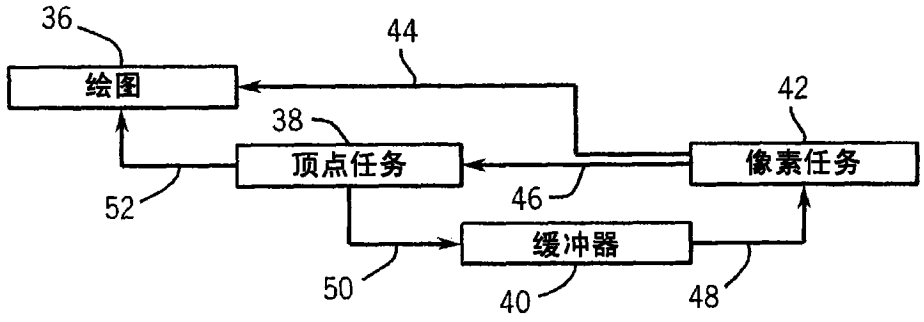

36
绘图
44
42
38
顶点任务
像素任务
52
46
缓冲器
50
40
48

图 3

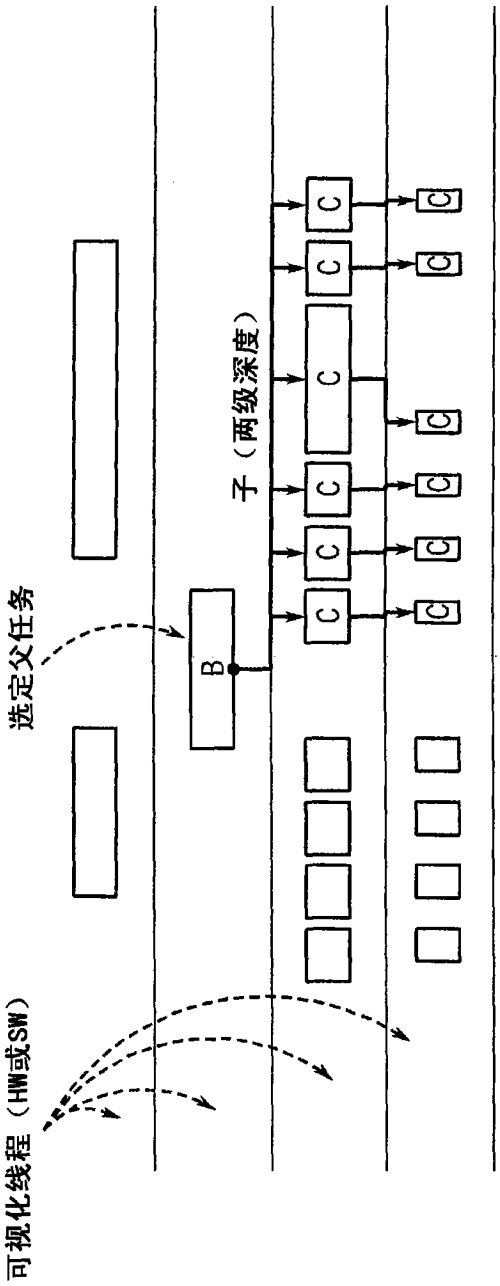


图 4

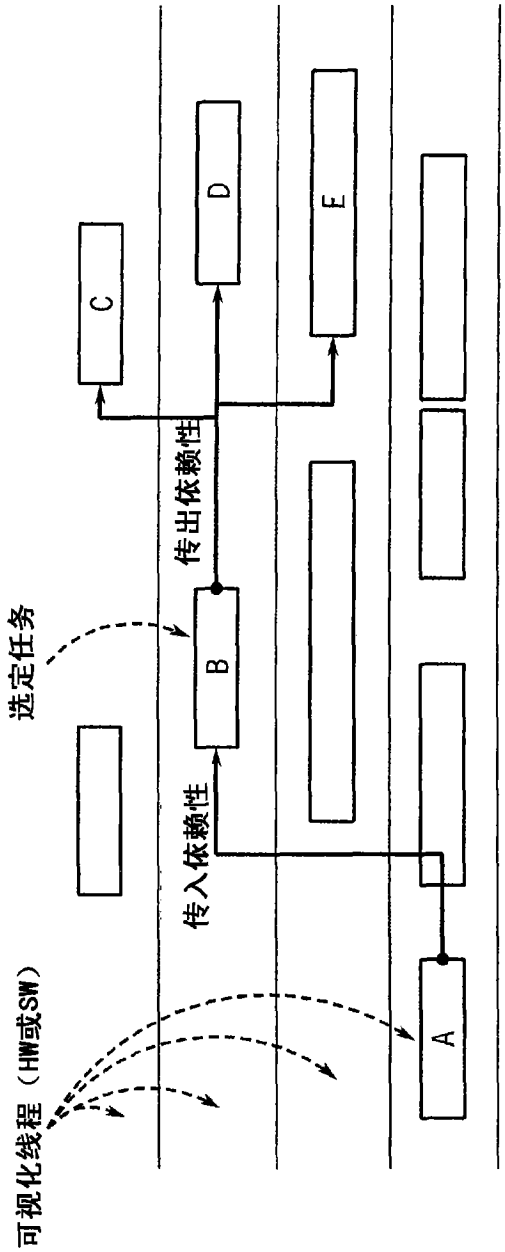

C
D
E
传出依赖性
选定任务
B
传入依赖性
可视化线程（HW或SW）
A

图 5

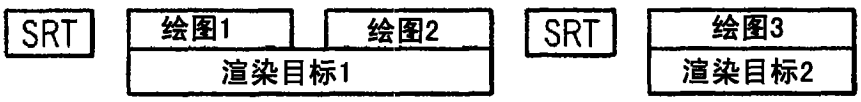

SRT
绘图1
绘图2
渲染目标1
SRT
绘图3
渲染目标2

图 6

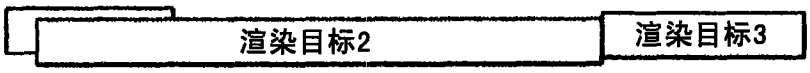

渲染目标2
渲染目标3

图 7

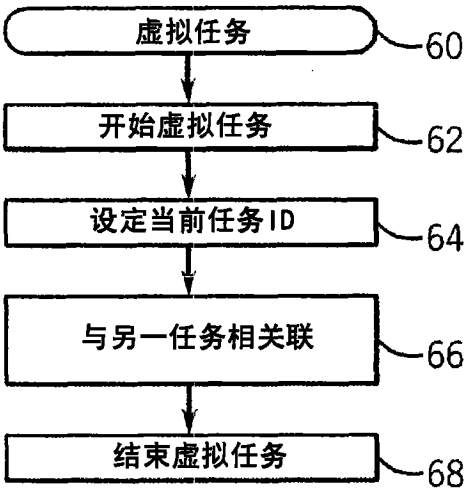


图 8