



(12) 发明专利

(10) 授权公告号 CN 102918501 B

(45) 授权公告日 2016. 05. 11

(21) 申请号 201180028522. 7

(51) Int. Cl.

(22) 申请日 2011. 05. 20

G06F 9/44(2006. 01)

G06F 9/38(2006. 01)

(30) 优先权数据

12/787240 2010. 05. 25 US

审查员 李敏

(85) PCT国际申请进入国家阶段日

2012. 12. 10

(86) PCT国际申请的申请数据

PCT/US2011/037403 2011. 05. 20

(87) PCT国际申请的公布数据

W02011/149784 EN 2011. 12. 01

(73) 专利权人 英特尔公司

地址 美国加利福尼亚州

(72) 发明人 A. 佩古辛 A. 库里列夫

(74) 专利代理机构 中国专利代理(香港)有限公司

司 72001

代理人 马永利 李浩

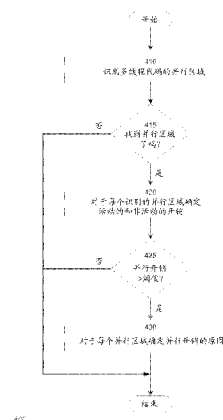
权利要求书5页 说明书8页 附图8页

(54) 发明名称

用于分析多线程应用的性能的方法和系统

(57) 摘要

提供确定多线程应用的特定问题的分析模型的方法和系统。在本发明的一个实施例中，多线程应用使用多个线程以用于执行，以及基于每个线程的当前状态，每个线程被分配给多个状态中的相应状态。通过这样做，基于对于每个线程而言在多个状态之间转移的数目，确定多线程应用的特定问题。在本发明的一个实施例中，分析模型使用工作者线程转移计数器或事件以对于多线程应用的每个并行区域或算法来确定哪个问题已经发生以及它对于并行区域或算法的可伸缩性有多少影响。



1. 一种分析在多线程框架上执行的多线程应用的性能的方法, 包括:

确定所述多线程应用的并行区域的并行开销是否超过阈值; 以及

响应于关于所述多线程应用的并行区域的并行开销超过阈值的确定, 确定所述并行区域的并行开销的一个或多个原因, 其中, 所述并行开销包括活动的并行开销和非活动的并行开销, 以及其中确定所述多线程应用的并行区域的并行开销是否超过阈值包括: 确定所述多线程应用的并行区域的活动的并行开销和 / 或非活动的并行开销是否超过阈值。

2. 根据权利要求 1 所述的方法, 其中, 确定所述并行区域的并行开销的一个或多个原因包括:

基于设置的监视时段来执行所述并行区域。

3. 根据权利要求 2 所述的方法, 其中, 所述并行区域的执行使用一个或多个工作者线程, 以及所述方法还包括:

响应于每个线程执行所述并行区域的一个或多个任务之一, 将所述一个或多个工作者线程中的每个分配给用户代码执行状态;

响应于每个线程在本地队列中寻找任务, 将所述一个或多个工作者线程中的每个分配给本地队列查找状态, 其中所述任务是所述并行区域的一个或多个任务的一部分;

响应于每个线程在另一队列中或在全局队列中寻找另一任务, 将所述一个或多个工作者线程中的每个分配给全局任务搜索状态, 其中所述另一任务是所述并行区域的一个或多个任务的一部分; 以及

响应于每个线程处于空闲状态, 将所述一个或多个工作者线程中的每个分配给等待状态。

4. 根据权利要求 3 所述的方法, 还包括:

确定对于所述一个或多个工作者线程中的每个而言在所述本地队列、所述另一队列和所述全局队列中任务的执行的频率是否超过另一阈值; 以及

响应于关于任务执行的频率超过另一阈值的确定, 报告每个工作者线程正在花费太多的它的生存期从所述本地队列中取任务以作为所述并行区域的并行开销的一个或多个原因之一。

5. 根据权利要求 3-4 中的任一项所述的方法, 还包括:

确定在所述本地队列中任务的执行与在所述另一队列和所述全局队列中任务的执行的比是否超过另一阈值; 以及

响应于关于在所述本地队列中任务的执行与在所述另一队列和所述全局队列中任务的执行的比超过另一阈值的确定, 报告每个工作者线程正在花费太多的时间寻找在所述另一本地队列中的另一任务或寻找在所述全局队列中的另一任务, 以作为所述并行区域的并行开销的一个或多个原因之一。

6. 根据权利要求 3-4 中的任一项所述的方法, 还包括:

确定对于所述一个或多个工作者线程中的每个而言从所述全局任务搜索状态到所述全局任务搜索状态和到所述等待状态的状态转移的频率是否超过另一阈值;

响应于关于对于所述一个或多个工作者线程中的每个而言从所述全局任务搜索状态到所述全局任务搜索状态和到所述等待状态的状态转移的频率超过另一阈值的确定, 报告所述并行区域的一个或多个任务小于所述一个或多个工作者线程, 以作为所述并行区域的

并行开销的一个或多个原因之一；以及

响应于关于对于所述一个或多个工作者线程中的每个而言从所述全局任务搜索状态到所述全局任务搜索状态和到所述等待状态的状态转移的频率没有超过另一阈值的确定以及关于所述多线程应用的并行区域的非活动的并行开销超过阈值的确定，报告所述一个或多个工作者线程中的至少一个已完成它们的任务但是不能帮助所述一个或多个工作者线程中的剩余工作者线程，以作为所述并行区域的并行开销的一个或多个原因之一。

7. 根据权利要求 1 所述的方法，其中，所述多线程框架至少部分地与开放多处理（OpenMP）、Intel® 线程构建块（TBB）、Intel® Cilk++、用于吞吐量计算的 Intel® C++（Ct）和 Microsoft® 并行模式库（PPL）之一相兼容。

8. 一种装置，包括：

一个或多个处理器；

一个或多个机器可读介质，耦合到所述一个或多个处理器，包含一个或多个机器可读指令，所述一个或多个机器可读指令响应于在所述一个或多个处理器上的执行而使得所述装置操作多线程系统，其中所述多线程系统包括：

操作系统，其被配置成由所述一个或多个处理器进行操作以使用并行线程库来执行多线程应用；

多线程应用性能分析器，其被配置成由所述一个或多个处理器进行操作以：

确定所述多线程应用具有次优并行算法；以及

使用线程生存周期状态机来确定所述次优并行算法的一个或多个性能问题。

9. 根据权利要求 8 所述的装置，其中，确定所述多线程应用具有次优并行算法包括：确定所述次优算法的并行开销可与所述多线程应用的性能增益相比或者大于所述性能增益。

10. 根据权利要求 8-9 中的任一项所述的装置，其中，所述次优并行算法的一个或多个性能问题包括次优粒度、不足的并行松弛度、以及过多的任务挪用中的一个或多个。

11. 根据权利要求 8 所述的装置，其中，所述线程生存周期状态机包括：执行状态，其中所述执行状态包括第一多个线程，所述第一多个线程中的每个处于执行所述次优并行算法的多个作业之一的状态；

本地队列查找状态，其中所述本地队列查找状态包括第二多个线程，所述第二多个线程中的每个处于搜索本地队列中的作业的状态，其中所述作业是所述次优并行算法的多个作业的一部分；

全局作业搜索状态，其中所述全局作业搜索状态包括第三多个线程，所述第三多个线程中的每个处于寻找在另一队列中或在全局队列中的另一作业的状态，其中所述另一作业是所述次优并行算法的多个作业的一部分；以及

等待状态，其中所述等待状态包括第四多个线程，所述第四多个线程中的每个处于空闲状态。

12. 根据权利要求 11 所述的装置，其中，使用线程生存周期状态机来确定所述次优并行算法的一个或多个性能问题包括：

在确定的时段内执行所述次优并行算法；

确定所述执行状态到所述本地队列查找状态的转移的数目与所述本地队列查找状态到所述全局作业搜索状态的转移和重复所述全局作业搜索状态的数目的总和；

确定所述总和与所述确定的时段的持续时间的比超过第一阈值；以及
识别所述次优并行算法的粒度太精细以作为所述次优并行算法的一个或多个性能问题中的一个问题。

13. 根据权利要求 11 所述的装置,其中,使用线程生存周期状态机来确定所述次优并行算法的一个或多个性能问题包括:

确定所述本地队列查找状态到所述全局作业搜索状态的转移和重复所述全局作业搜索状态的数目与所述执行状态到所述本地队列查找状态的转移的数目的比超过第二阈值；以及

识别所述次优并行算法的线性衍生以作为所述次优并行算法的一个或多个性能问题中的一个问题。

14. 根据权利要求 11-13 中的任一项所述的装置,其中,使用线程生存周期状态机来确定所述次优并行算法的一个或多个性能问题包括:

确定所述全局作业搜索状态到所述全局作业搜索状态和到所述等待状态的转移的数目是否超过第三阈值；响应于关于所述全局作业搜索状态到所述全局作业搜索状态和到所述等待状态的转移的数目超过第三阈值的确定,识别所述次优并行算法的过多等待以作为所述次优并行算法的一个或多个性能问题中的一个问题；以及

响应于关于所述全局作业搜索状态到所述全局作业搜索状态和到所述等待状态的转移的数目没有超过第三阈值的确定以及关于所述次优并行算法的非活动的并行开销超过第四阈值的确定,识别所述次优并行算法的不足的作业创建以作为所述次优并行算法的一个或多个性能问题中的一个问题。

15. 根据权利要求 8 所述的装置,其中,所述并行线程库至少部分地与开放多处理(OpenMP)、Intel® 线程构建块(TBB)、用于吞吐量计算的 Intel® C++ (Ct) 和 Microsoft® 并行模式库(PPL)之一相兼容。

16. 根据权利要求 8 所述的装置,其中,所述多线程应用性能分析器是所述并行线程库的一部分。

17. 一种设备,包括:

用于基于每个线程的当前状态,将多个线程中的每个分配给多个状态中的相应状态的装置；以及

用于至少部分地基于对于每个线程而言在所述多个状态之间转移的数目,确定多线程应用的并行区域的一个或多个问题的装置。

18. 根据权利要求 17 所述的设备,其中,所述多个状态包括:

执行状态,其中响应于每个线程处于执行所述多线程应用的并行区域的一部分的当前状态,将每个线程分配给所述执行状态；

本地队列查找状态,其中响应于每个线程处于搜索在本地队列中的作业的当前状态,将每个线程分配给所述本地队列查找状态,其中所述作业是所述多线程应用的并行区域的多个作业的一部分；

全局作业搜索状态,其中响应于每个线程处于寻找在另一队列中或在全局队列中的另一作业的当前状态,将每个线程分配给所述全局作业搜索状态,其中所述另一作业是所述多线程应用的并行区域的多个作业的一部分；以及

等待状态,其中响应于每个线程处于非活动的当前状态,将每个线程分配给所述等待状态。

19. 根据权利要求 18 所述的设备,其中,用于至少部分地基于对于每个线程而言在所述多个状态之间转移的数目而确定多线程应用的并行区域的一个或多个问题的所述装置包括:

用于在确定的时段内执行所述多线程应用的并行区域的装置;

用于确定所述执行状态到所述本地队列查找状态的转移的数目与所述本地队列查找状态到所述全局作业搜索状态的转移和重复所述全局作业搜索状态的数目的总和的装置;

用于确定所述总和与所述确定的时段的持续时间的比超过第一阈值的装置;以及

用于识别所述多线程应用的次优并行算法的粒度太精细以作为所述次优并行算法的一个或多个性能问题中的一个问题的装置。

20. 根据权利要求 18 所述的设备,其中,用于至少部分地基于对于每个线程而言在所述多个状态之间转移的数目而确定多线程应用的并行区域的一个或多个问题的所述装置包括:

用于确定所述本地队列查找状态到所述全局作业搜索状态的转移和重复所述全局作业搜索状态的数目与所述执行状态到所述本地队列查找状态的转移的数目的比超过第二阈值的装置;以及

用于识别所述多线程应用的次优并行算法的线性衍生以作为所述次优并行算法的一个或多个性能问题中的一个问题的装置。

21. 根据权利要求 18 所述的设备,其中,用于至少部分地基于对于每个线程而言在所述多个状态之间转移的数目而确定多线程应用的并行区域的一个或多个问题的所述装置包括:

用于确定所述全局作业搜索状态到所述全局作业搜索状态和到所述等待状态的转移的数目是否超过第三阈值的装置;以及

用于响应于关于所述全局作业搜索状态到所述全局作业搜索状态和到所述等待状态的转移的数目超过第三阈值的确定,识别所述多线程应用的并行区域的过多等待以作为所述多线程应用的并行区域的一个或多个问题中的一个问题的装置。

22. 根据权利要求 18-21 中的任一项所述的设备,其中,用于至少部分地基于对于每个线程而言在所述多个状态之间转移的数目而确定多线程应用的并行区域的一个或多个问题的所述装置包括:

用于确定对于所述一个或多个工作者线程中的每个而言所述全局任务搜索状态到所述全局任务搜索状态和到所述等待状态的转移的数目是否超过另一阈值的装置;

用于响应于关于所述全局作业搜索状态到所述全局作业搜索状态和到所述等待状态的转移的数目超过第三阈值的确定,识别所述多线程应用的并行区域的过多等待以作为所述多线程应用的并行区域的一个或多个问题中的一个问题的装置;以及

用于响应于关于所述全局作业搜索状态到所述全局作业搜索状态和到所述等待状态的转移的数目没有超过第三阈值的确定以及关于所述多线程应用的并行区域的非活动的并行开销超过第五阈值的确定,识别所述多线程应用的并行区域的不足的作业创建以作为

所述多线程应用的并行区域的一个或多个问题中的一个问题的装置。

用于分析多线程应用的性能的方法和系统

技术领域

[0001] 本发明涉及多线程应用,并且更具体地但不排他地,涉及确定使用多线程框架的多线程应用的特定问题的分析模型。

背景技术

[0002] 多线程框架,例如开放多处理(OpenMP)、Intel® 线程构建块(TBB)、Intel® Cilk++、用于吞吐量计算的 Intel® C++(Ct)、以及 Microsoft® 并行模式库(PPL),允许并行性以改进多线程应用的性能。多线程应用的优点可以在具有多个中央处理单元(CPU)或者具有带有多个核心的 CPU 的计算机系统上被观察到,因为多线程应用的每个线程使用 CPU/核心之一以用于并行执行。

[0003] 然而,如果多线程框架被不正确地使用以执行多线程应用,则并行性的优点可能被损害。图 1A 示出并行循环的现有技术代码 100。函数 $foo()$ 的粒度被设置为 1。取决于函数 $foo()$ 花费多长时间来执行,并行性的优点可能被损害,因为 1 的粒度太精细。

[0004] 图 1B 示出具有动态调度的并行循环的现有技术代码 130。函数 $foo()$ 的粒度被设置为 3。动态调度需要分布式开销,并且取决于函数 $foo()$ 花费多长时间来执行,并行性的优点可能被损害,因为 3 的粒度太精细。

[0005] 图 1C 示出从仅仅一个线程衍生(spawn)或创建工作任务的现有技术代码 150。取决于变量 N 被设置为多大,现有技术代码 150 可能具有线性衍生问题,所述线性衍生问题具有相当大的活动的挪用(stealing)开销。例如,当变量 N 被设置为大于 100 时,这一执行比具有递归衍生的另一执行在规模(scale)上坏得多。

[0006] 现有技术代码 100、130 和 150 示出其中多线程应用可能被不正确地或低效地使用的可能情景。

附图说明

[0007] 根据本主题的下列详细描述,本发明的实施例的特征和优点将变得明显,其中:

[0008] 图 1A 示出并行循环的现有技术代码;

[0009] 图 1B 示出具有动态调度的并行循环的现有技术代码;

[0010] 图 1C 示出从仅仅一个线程衍生或创建工作任务的现有技术代码;

[0011] 图 2 示出根据本发明的一个实施例的平台的模块;

[0012] 图 3 示出根据本发明的一个实施例的基于工作者线程生存周期状态机的分析模型;

[0013] 图 4 示出根据本发明的一个实施例的分析多线程应用的性能的步骤的流程图;

[0014] 图 5A 示出根据本发明的一个实施例的对于多线程应用的每个并行区域确定并行开销的原因的步骤的流程图;

[0015] 图 5B 示出根据本发明的一个实施例的对于多线程应用的每个并行区域确定并行开销的原因的步骤的流程图;以及

[0016] 图 6 示出根据本发明的一个实施例的实施在这里公开的方法的系统。

具体实施方式

[0017] 在这里描述的本发明的实施例在附图中作为例子而不是作为限制被示出。为了说明的简单和清楚起见,在附图中示出的元件不一定是按比例绘制的。例如,为了清楚起见,一些元件的尺寸可能相对于其他元件被夸大。此外,在被认为适当的情况下,附图标记已在附图中被重复以指示对应的或类似的元件。在本说明书中对本发明的“一个实施例”或“实施例”的提及是指,结合该实施例所描述的特定的特征、结构、或特性被包括在本发明的至少一个实施例中。因此,在整个说明书中的各种位置中出现的短语“在一个实施例中”不一定都是指相同的实施例。

[0018] 本发明的实施例提供确定使用多线程框架的多线程应用的特定问题的分析模型。在本发明的一个实施例中,多线程应用使用多个线程以用于执行,并且基于每个线程的当前状态,每个线程被分配给多个状态中的相应状态。通过这样做,基于对于每个线程而言在多个状态之间转移的频率,确定多线程应用的特定问题。在本发明的一个实施例中,分析模型使用工作者线程转移计数器或事件以对于多线程应用的每个并行区域或算法来确定哪个问题已经发生以及它对于并行区域或算法的可伸缩性有多少影响。

[0019] 在本发明的一个实施例中,如果从用户的观点来看来自多线程应用的并行执行的性能好处是不重要的或不可接受的,则使用多线程框架或并行线程库来执行多线程应用是低效的或成问题的。性能好处包括但不限于更快的执行时间、更小数量的 CPU 节拍(tick)等等。例如,在本发明的一个实施例中,当执行多线程应用的并行开销可与来自多线程应用的并行执行的性能好处相比或者超过该性能好处时,使用多线程框架来执行多线程应用是低效的。

[0020] 图 2 示出根据本发明的一个实施例的平台 200 的模块。平台或系统 200 具有多个处理器和 / 或多核心处理器,并且操作系统(OS) 250 在所述多个处理器的至少一个处理器上或者在多核心处理器的一个核心上执行。OS 250 具有可以用来在平台 200 上执行多线程应用 210 的本地线程 255 的池。

[0021] 资源管理器 240 管理本地线程 255 的池并控制线程对于执行的可用性。任务调度器 230 调度要由来自本地线程 255 的池的可用线程执行的任务。并行线程库 220 包含可以使用本地线程 255 由用于并行执行的多线程应用 210 参考的或使用的功能。

[0022] 在本发明的一个实施例中,多线程应用 210 使用并行线程库 220 或多线程框架,所述多线程框架包括但不限于开放多处理(OpenMP)、Intel® 线程构建块(TBB)、Intel® Cilk++、用于吞吐量计算的 Intel® C++ (Ct)、Microsoft® 并行模式库(PPL)和任何其他多线程框架。

[0023] 在本发明的一个实施例中,并行线程库 220 具有确定多线程应用 210 具有次优并行算法以及使用分析模型来确定次优并行算法的一个或多个性能问题的逻辑。在本发明的另一实施例中,所述逻辑是 Intel® 并行放大器软件的一部分。在本发明的又一实施例中,所述逻辑可以是平台 200 的任何模块的一部分。

[0024] 虽然平台 200 的模块被描绘为分开的块,但是一个模块的操作可以由另一模块执行。例如,在一个实施例中,OS 250 可以执行资源管理器 240 的功能。在另一实施例中,并

行线程库 220 也可以与 OS 250 相集成。相关领域的普通技术人员将容易认识到,可以执行模块或功能的不同组合而不影响本发明的工作。

[0025] 图 3 示出根据本发明的一个实施例的基于工作者线程生存周期状态机的分析模型 300。当执行多线程应用时,招致或需要并行开销以便执行多线程应用的并行执行。并行开销包括但不限于线程维护、在工作者线程之间的作业分配等等。并行开销在本发明的一个实施例中被分类成活动的和非活动的并行开销。

[0026] 在本发明的一个实施例中,在过多的活动的并行开销会影响多线程应用的并行执行的情况下有两种情景。当一个或多个工作者线程花费太多的它们的生存期从它们的本地队列中取作业单元或任务时,活动的并行开销的第一情景发生。第一情景是由于多线程应用的并行区域的粒度被设置为太精细或太小的水平。当一个或多个工作者线程花费太多的它们的生存期在其他任务队列中或在全局作业管理者队列中寻找任务时,即工作者线程除了它自己的本地队列之外还从另一源中挪用任务时,活动的并行开销的第二情景发生。

[0027] 类似地,在本发明的一个实施例中,在过多的非活动的并行开销会影响多线程应用的并行执行的情况下有两种情景。当一个或多个工作者线程因为它们已完成它们的任务但是不能帮助仍在忙于执行它们的任务的剩余工作者线程而变为空闲时,非活动的并行开销的第一情景发生。所述一个或多个工作者线程的过多等待造成非活动的并行开销。

[0028] 当没有足够的作业或任务创建来充满或利用所有可用工作者线程时,非活动的并行开销的第二情景发生。工作者线程连续地从执行一个任务转移到搜索另一任务并且当没有可用任务时进入空闲状态。当新任务可用时,所有可用线程再次变为活动的。对于新任务没有分配的可用线程反复进行连续转移以搜索另一任务并且当没有可用任务时进入空闲状态。

[0029] 在本发明的一个实施例中,使用分析模型 300 在多线程应用的并行区域的执行期间识别活动的和非活动的并行开销的四种情景。活动的和非活动的并行开销的四种情景中的一个或多个可以在多线程应用的并行区域内发生。分析模型 300 具有工作者线程生存周期状态机,其具有描述工作者线程的当前状态的四个状态。每个工作者线程在它的生存期期间被分配给四个状态之一。

[0030] 第一状态是用户代码执行状态 310,其中每个线程在它处于执行多线程应用的并行区域的一部分的当前状态时被分配给用户代码执行状态 310。当执行调度器旁路 316 时,工作者线程保持在用户代码执行状态 310。理想地,每个工作者线程应当保持在用户代码执行状态 310 以使活动的和非活动的并行开销最小化。

[0031] 第二状态是本地队列查找状态 320,其中每个线程在它处于在它的本地队列中搜索作业或任务的当前状态时被分配给本地队列查找状态 320。每个线程在它完成它的当前任务之后进入本地队列查找状态 320,并且在它的本地任务队列中搜索新任务。

[0032] 第三状态是全局任务搜索状态 330,其中每个线程在它处于在另一任务队列或在全局队列中搜索任务的当前状态时被分配给全局任务搜索状态 330。每个线程当它在本地队列查找状态 320 期间在它的本地队列中不能找到任务时进入全局任务搜索状态 330。

[0033] 第四状态是等待状态 340,其中每个线程在它处于非活动或空闲状态的当前状态时被分配给等待状态 340。每个线程当它在全局任务搜索状态 330 期间不能找到任务时进入等待状态 340。当新任务变为可用时,处于等待状态 340 的每个线程回到全局任务搜索状

态 330 以寻找新任务。状态转移路径 312、314、316、322、332、334、336 和 338 示出每个线程在分析模型 300 的四个状态之间的状态转移。

[0034] 在本发明的一个实施例中,在多线程应用的并行区域执行期间测量或计算每个线程在分析模型 300 的四个状态之间的状态转移的数目。在本发明的一个实施例中,整个多线程应用被执行以确定每个线程在分析模型 300 的四个状态之间的状态转移的数目。在本发明的另一实施例中,多线程应用的仅仅一部分被执行以确定每个线程在分析模型 300 的四个状态之间的状态转移的数目。要被执行的多线程应用的部分是基于但不限于测量时段、多线程应用的总运行时间的百分比等等。

[0035] 为了识别在多线程应用的并行区域的执行期间存在活动的和非活动的并行开销的四种情景中的哪个情景,在本发明的一个实施例中使用每个线程在分析模型 300 的四个状态之间的状态转移的经测量数目。在本发明的另一实施例中,使用在分析模型 300 的四个状态之间的状态转移的频率。例如,在本发明的一个实施例中,所述频率是从状态转移的经测量数目与测量时段的持续时间的比来确定的。

[0036] 在本发明的一个实施例中,当对于每个线程而言任务或作业执行的频率超过阈值时,识别或确定活动的并行开销的第一情景。在本发明的一个实施例中,确定由每个工作者线程所执行的任务的数目。由每个工作者线程所执行的任务的数目是通过把从它的本地队列中取的任务的数目与从另一任务队列或全局队列中取的任务的数目相加来确定的。对于每个线程而言任务或作业执行的频率是通过由每个工作者线程所执行的任务的数目与测量时段的持续时间的比来确定的。

[0037] 对于每个线程而言从它的本地队列中取的任务的数目是从如在分析模型 300 中所示的、从用户代码执行状态 310 到本地队列查找状态 320 的状态转移路径 312 的出现的数目来确定的。对于每个线程而言从另一任务队列或全局队列中取的任务的数目是从本地队列查找状态 320 到全局任务搜索状态 330 的状态转移和反复进行全局任务搜索状态 330 的出现的数目来确定的。

[0038] 当识别活动的并行开销的第一情景时,把关于活动的并行开销的原因是由于每个工作者线程花费太多的它的生存期从本地队列中取任务(即多线程应用的并行区域的任务的粒度太精细)的报告发送给用户。

[0039] 当对于每个线程而言从另一任务队列或全局队列中取的任务的数目与从它的本地队列中取的任务的数目的比超过阈值时,识别或确定活动的并行开销的第二情景。在本发明的一个实施例中,阈值被设置为 1,即当与从它的本地队列中取的任务相比每个线程正在执行从其他线程挪用的更多的任务时,第二情景发生。

[0040] 当识别活动的并行开销的第二情景时,把关于活动的并行开销的原因是由于每个工作者线程花费太多的时间在全局队列中或在另一队列中寻找任务(即多线程应用的并行区域具有线性衍生问题)的报告发送给用户。

[0041] 在本发明的一个实施例中,当从全局任务搜索状态 330 到全局任务搜索状态 330 和到等待状态 340 的状态转移的频率超过特定阈值时,识别或确定非活动的并行开销的第二情景。这在分析模型 300 中通过重复或反复进行全局任务搜索状态 330 的状态转移路径 336 和从全局任务搜索状态 330 到等待状态 340 的状态转移路径 332 来示出的。

[0042] 当识别非活动的并行开销的第二情景时,把关于非活动的并行开销的原因是因为

并行区域的任务的数目小于可用的或自由的工作者线程的数目(即多线程应用的并行区域的不足的作业或任务创建)的报告发送给用户。

[0043] 在本发明的一个实施例中,当从全局任务搜索状态 330 到全局任务搜索状态 330 和到等待状态 340 的状态转移的频率没有超过特定阈值时并且当非活动的并行开销已经超过阈值时,识别或确定非活动的并行开销的第一情景。当识别非活动的并行开销的第一情景时,把关于活动的并行开销的原因是因为一个或多个工作者线程已完成它们的任务但是不能帮助剩余工作者线程(即线程的过多等待)的报告发送给用户。

[0044] 当识别非活动的并行开销的第一情景时,把关于非活动的并行开销的原因是并行性的粒度太粗的报告发送给用户。这意味着,工作被划分成太大的块,这限制所创建的块的数目,其又限制并行性。

[0045] 当活动的和非活动的并行开销的一个或多个情景被识别时,在本发明的一个实施例中它允许基于生成的特定报告来校正多线程应用的特定并行区域。这允许多线程应用的并行执行的改进。本发明的实施例允许用户解释分析模型 300 的结果以基于报告或结论来改正多线程应用中的错误。在本发明的一个实施例中,除了对于每个情景的特定报告以外,还给出解决特定问题或情景的启示或建议。

[0046] 分析模型 300 是基于工作者线程生存周期状态的,而不是基于任务的。这允许使用确定活动的和 / 或非活动的并行开销的原因的类似方法来分析基于任务的和非任务的多线程框架。在本发明的一个实施例中,分析模型可被集成到 Intel® 并行放大器软件中。

[0047] 对于活动的和 / 或非活动的并行开销的四种情景的描述并不意味着是限制性的。相关领域的普通技术人员将容易认识到,可以识别其他情景而不影响本发明的工作。类似地,分析模型 300 中的四个状态并不意味着是限制性的。相关领域的普通技术人员将容易认识到,可以添加其他状态而不影响本发明的工作。

[0048] 图 4 示出根据本发明的一个实施例的分析多线程应用的性能的步骤的流程图 400。在步骤 410,流程 400 识别多线程代码或应用的并行区域。在步骤 415,流程 400 检查是否已找到并行区域。如果是的话,则流程 400 转到步骤 420 以对于每个识别的并行区域确定活动的和非活动的并行开销。如果否的话,则流程 400 结束。

[0049] 在步骤 425,流程 400 检查活动的和非活动的并行开销是否超过阈值。在本发明的一个实施例中,把单个阈值与活动的和非活动的并行开销进行比较。在本发明的另一实施例中,把分开的阈值分别与活动的和非活动的并行开销进行比较。如果是的话,则流程 400 转到步骤 430 以对于多线程代码或应用的每个并行区域确定并行开销的原因。如果否的话,则流程 400 结束。

[0050] 图 5A 示出根据本发明的一个实施例的对于多线程应用的每个并行区域确定并行开销的原因的步骤的流程图 500。为了清楚起见,参照图 3 来描述图 5A。

[0051] 在步骤 510,所述流程设置监视时段并复位所有的状态转移路径计数器。在本发明的一个实施例中,对于状态转移路径 312、314、316、322、332、334、336 和 338 中的每个保持计数器。在步骤 515,流程 500 在设置的监视时段内执行多线程应用的每个并行区域。在所述执行期间,当多线程应用的每个并行区域的每个线程在分析模型 300 中的四个状态 310、320、330 和 340 之间转移时,递增相应状态转移路径计数器。

[0052] 在步骤 520,流程 500 确定对于每个工作者线程而言任务或作业执行的频率。在步

骤 525, 流程 500 确定对于每个工作者线程而言任务执行的频率是否超过阈值。例如, 在本发明的一个实施例中, 对于与多线程应用的执行相比不重要的 TBB 并行开销而言, 任务可能必须是大于 5000 个 CPU 节拍。假设在以 2 吉赫 (GHz) 的时钟速度运行的 CPU 上执行多线程应用, 400,000 个任务应当以每秒每个线程来执行。阈值在本发明的一个实施例中被设置为每秒 400,000 个任务。

[0053] 如果在步骤 525 中阈值被超过, 则流程 500 转到步骤 530 以报告并行区域的特定问题。在本发明的一个实施例中, 步骤 530 报告: 活动的并行开销是由于每个工作者线程花费太多的它的生存期从本地队列中取任务。如果在步骤 525 中阈值没有被超过, 则流程 500 转到步骤 535。

[0054] 在步骤 535, 流程 500 确定从另一任务队列或全局队列中取的任务 (即挪用的任务) 的数目与从对于每个线程而言它的本地队列中取的任务的数目的比。在步骤 540, 流程 500 确定所述比是否超过阈值。如果是的话, 则流程 500 转到步骤 545 以报告并行区域的特定问题。在本发明的一个实施例中, 步骤 545 报告: 对于活动的并行开销的原因是由于每个工作者线程花费太多的时间寻找在另一任务队列中或在全局队列中的任务。如果否的话, 流程 500 转到图 5B 中的步骤 5B。

[0055] 图 5B 示出根据本发明的一个实施例的对于多线程应用的每个并行区域确定并行开销的原因的步骤的流程图 550。流程 550 从步骤 5B 转到步骤 560。在步骤 560, 流程 550 确定非活动的并行开销是否超过阈值。如果否的话, 则流程 550 结束。如果是的话, 流程 550 转到步骤 565 以确定从全局任务搜索状态 330 到全局任务搜索状态 330 和到等待状态 340 的状态转移的频率。例如, 在本发明的一个实施例中, 流程 550 确定对于状态转移路径 336 和 332 的状态转移路径计数器的总和与设置的监视时段的持续时间的比。从这个比获得对于每个线程而言从全局任务搜索状态 330 到全局任务搜索状态 330 和到等待状态 340 的状态转移的频率。

[0056] 在步骤 570, 流程 550 确定对于每个工作者线程而言从全局任务搜索状态 330 到全局任务搜索状态 330 和到等待状态 340 的状态转移的频率是否超过阈值。如果是的话, 则流程 550 转到步骤 575 以报告并行区域的特定问题, 并且流程 550 结束。在本发明的一个实施例中, 步骤 575 报告: 对于非活动的并行开销的原因是因为并行区域的任务的数目小于可用的或自由的工作者线程的数目。

[0057] 如果否的话, 则流程 550 转到步骤 580 以报告并行区域的特定问题, 并且流程 550 结束。在本发明的一个实施例中, 步骤 580 报告: 对于非活动的并行开销的原因是因为一个或多个工作者线程已完成它们的任务但是不能帮助剩余工作者线程。

[0058] 图 4、5A 和 5B 中所示的步骤并不意味着是限制性的。相关领域的普通技术人员将容易认识到, 可以使用图 4、5A 和 5B 中所示的步骤的其他序列而不影响本发明的工作。例如, 在本发明的一个实施例中, 图 4、5A 和 5B 中所示的一些步骤被并行地执行。在本发明的一个实施例中, 对于步骤 425、525、540、565 和 575 所需的阈值可以基于来自特定并行线程库或多线程框架的开发者的推荐来确定。在本发明的另一实施例中, 对于步骤 425、525、540、565 和 575 所需的阈值可以基于执行分析模型 300 中四个状态 310、320、330 和 340 中的每个状态的成本来确定。

[0059] 图 6 示出根据本发明的一个实施例的实施在这里公开的方法的系统。系统 600 包

括但不限于台式计算机、膝上型计算机、上网本、笔记本电脑、个人数字助理(PDA)、服务器、工作站、蜂窝电话、移动计算设备、因特网装置、或任何其他类型的计算设备。在另一实施例中,用来实施在这里公开的方法的系统 600 可以是片上系统(SOC)的系统。

[0060] 处理器 610 具有执行系统 600 的指令的处理核心 612。处理核心 612 包括但不限于取指令的预取逻辑、译码指令的译码逻辑、执行指令的执行逻辑等等。处理器 610 具有对系统 600 的指令和 / 或数据进行高速缓存的高速缓冲存储器 616。在本发明的另一实施例中,高速缓冲存储器 616 包括但不限于第 1 级、第 2 级、和第 3 级高速缓冲存储器或在处理器 610 内的高速缓冲存储器的任何其他配置。

[0061] 存储器控制中心(MCH) 614 执行使处理器 610 能够访问包括易失性存储器 632 和 / 或非易失性存储器 634 的存储器 630 并与该存储器 630 通信的功能。易失性存储器 632 包括但不限于同步动态随机存取存储器(SDRAM)、动态随机存取存储器(DRAM)、RAMBUS 动态随机存取存储器(RDRAM)、和 / 或任何其他类型的随机存取存储器设备。非易失性存储器 634 包括但不限于 NAND 闪速存储器、相变存储器(PCM)、只读存储器(ROM)、电可擦可编程只读存储器(EEPROM)、或任何其他类型的非易失性存储器设备。

[0062] 存储器 630 存储要由处理器 610 执行的信息和指令。存储器 630 还可以存储临时变量或其他中间信息,同时处理器 610 正在执行指令。芯片组 620 经由点对点(PtP)接口 617 和 622 而与处理器 610 连接。芯片组 620 使处理器 610 能够连接到系统 600 中的其他模块。在本发明的一个实施例中,接口 617 和 622 根据诸如 Intel® QuickPath Interconnect (QPI) 等等之类的 PtP 通信协议来操作。

[0063] 芯片组 620 连接到显示设备 640,显示设备 640 包括但不限于液晶显示器(LCD)、阴极射线管(CRT)显示器、或任何其他形式的视觉显示设备。在本发明的一个实施例中,处理器 610 和芯片组 620 被合并成 SOC。另外,芯片组 620 连接到一个或多个总线 650 和 655,所述一个或多个总线 650 和 655 互连各种模块 674、660、662、664 和 666。如果在总线速度或通信协议上存在失配,则可以把总线 650 和 655 经由总线桥 672 互连在一起。芯片组 620 与非易失性存储器 660、存储设备 662、键盘 / 鼠标 664 和网络接口 666 相耦合,但不限于非易失性存储器 660、存储设备 662、键盘 / 鼠标 664 和网络接口 666。在本发明的一个实施例中,固态驱动器 102 是存储设备 662。

[0064] 存储设备 662 包括但不限于固态驱动器、硬盘驱动器、通用串行总线闪速存储器驱动器、或任何其他形式的计算机数据存储介质。网络接口 666 使用任何类型的公知网络接口标准来实施,包括但不限于以太网接口、通用串行总线(USB)接口、外设部件互连(PCI)直通接口、无线接口和 / 或任何其他合适类型的接口。无线接口根据 IEEE 802.11 标准及其相关家族、家庭插头 AV (HPAV)、超宽带(UWB)、蓝牙、WiMax、或任何形式的无线通信协议来操作,但不限于 IEEE 802.11 标准及其相关家族、家庭插头 AV(HPAV)、超宽带(UWB)、蓝牙、WiMax、或任何形式的无线通信协议。

[0065] 虽然图 6 中所示的模块被描绘为系统 600 内分开的块,但是由这些块中的一些执行的功能可以被集成在单个半导体电路内,或者可以使用两个或更多个分开的集成电路来实施。例如,虽然高速缓冲存储器 616 被描绘为处理器 610 内分开的块,但是高速缓冲存储器 616 可以分别被结合在处理器核心 612 内。在本发明的另一实施例中,系统 600 可以包括多于一个处理器 / 处理核心。

[0066] 在这里公开的方法可以以硬件、软件、固件、或它们的任何其他组合来实施。虽然描述了所公开的主题的实施例的例子，但是相关领域的普通技术人员将容易认识到，可以替换地使用实施所公开的主题的许多其他方法。在前面的描述中已经描述了所公开的主题的各种方面。为了解释起见，阐述了特定的数目、系统和配置以便提供对于本主题的透彻理解。然而，对于具有本公开的益处的相关领域的技术人员而言下述是显然的，即本主题可以在没有特定细节的情况下被实践。在其他实例中，公知的特征、部件、或模块被省略、简化、组合、或分割以免模糊所公开的主题。

[0067] 在这里使用的术语“能够操作”是指，设备、系统、协议等等当设备或系统处于断电状态时能够操作或适于操作以用于它的期望功能。所公开的主题的各种实施例可以以硬件、固件、软件、或它们的组合来实施，并且可以参照或结合程序代码来描述，诸如指令、函数、过程、数据结构、逻辑、应用程序、用于设计的模拟、仿真和制造的设计表示或格式之类的程序代码当由机器访问时导致机器执行任务、定义抽象数据类型或低级别硬件上下文、或产生结果。

[0068] 在附图中所示的技术可以使用在一个或多个计算设备(例如通用计算机或计算设备)上存储和执行的代码和数据来实施。这样的计算设备使用机器可读介质(例如机器可读存储介质(例如磁盘、光盘、随机存取存储器、只读存储器、闪速存储器设备、相变存储器))和机器可读通信介质(例如电、光、声或其他形式的传播信号(例如载波、红外信号、数字信号等等))来存储和传送(内部地和通过网络与其他计算设备)代码和数据。

[0069] 虽然已经参照说明性实施例描述了所公开的主题，但是该描述并不打算在限制性的意义上被解释。说明性实施例的各种修改以及本主题的其他实施例(其对于所公开的主题所涉及的领域的技术人员而言将是显然的)被认为处于所公开的主题的范围内。

```
// 并行循环的TBB代码
element_t array[N];
size_t grain_size = 1;
tbb::parallel_for(tbb::blocked_range<size_t>(0, N, grain_size), [&array](const tbb::blocked_range<size_t>& r)
{
    for(size_t i = r.begin(); i < r.end(); ++i)
    {
        foo(array[i]);
    }
}, simple_partitioner());
```

100
(现有技术)

图 1A

```
// 具有动态调度的OpenMP并行循环
element_t array[N];
#pragma omp parallel for schedule(dynamic, 3)
for(size_t i = 0; i < N; ++i)
{
    foo(array[i]);
}
```

130
(现有技术)

图 1B

```
// 从仅仅一个线程衍生出工作任务的TBB代码。  
element__t array[N];  
tbb::task_group tg;  
for(size__t i = 0; i < N; ++i)  
{  
    tg.run([&array, i]() { foo(array[i]); });  
}
```

150

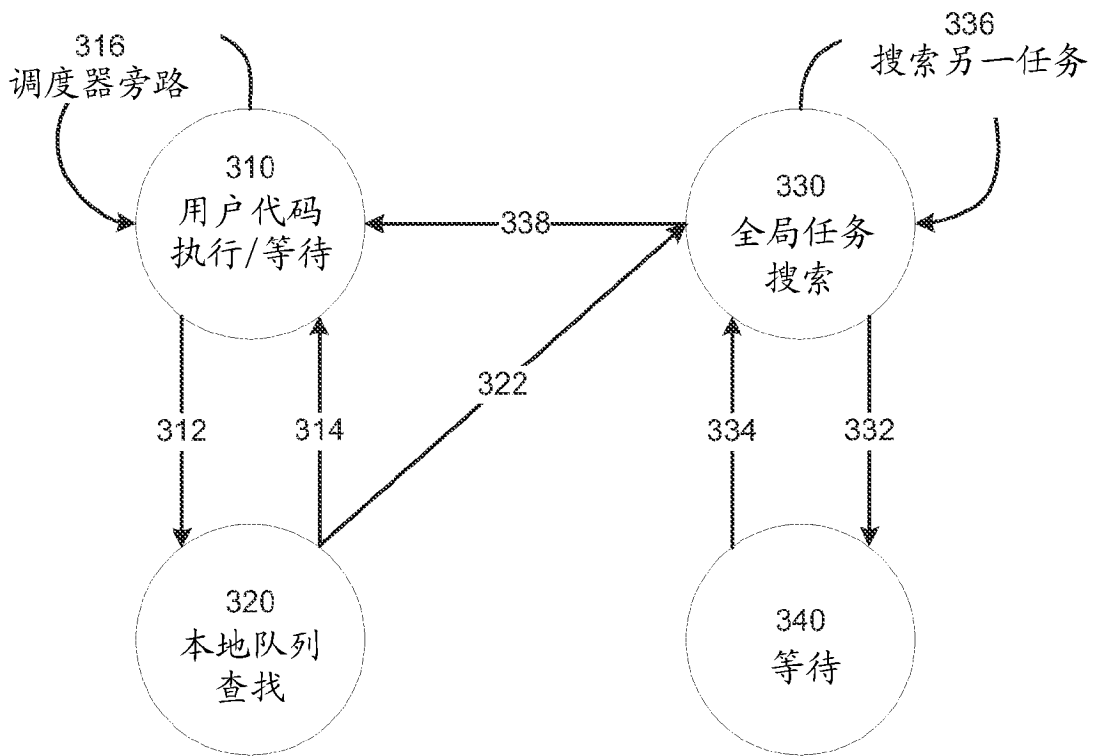
(现有技术)

图 1C

200



图 2



300

图 3

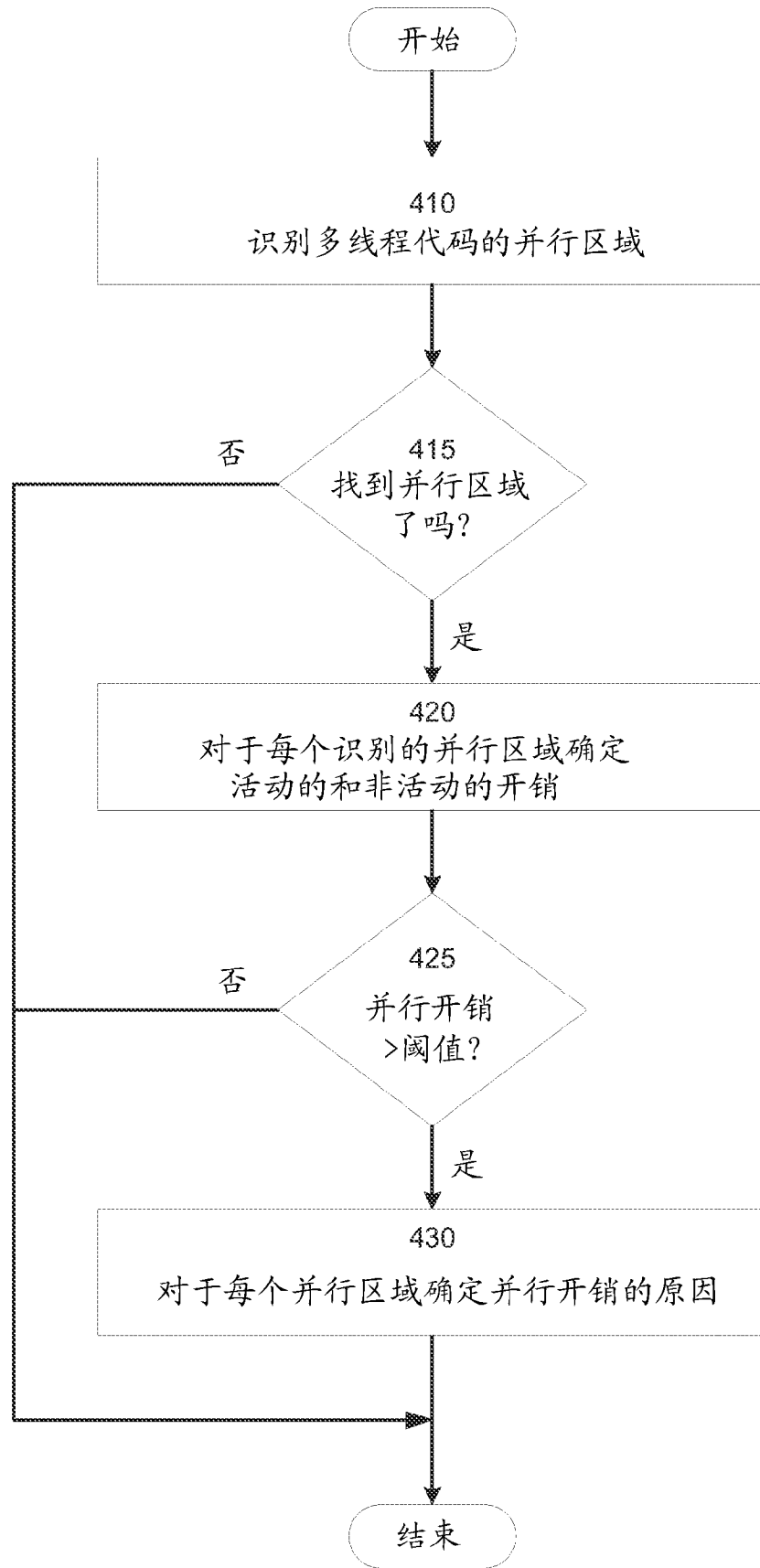
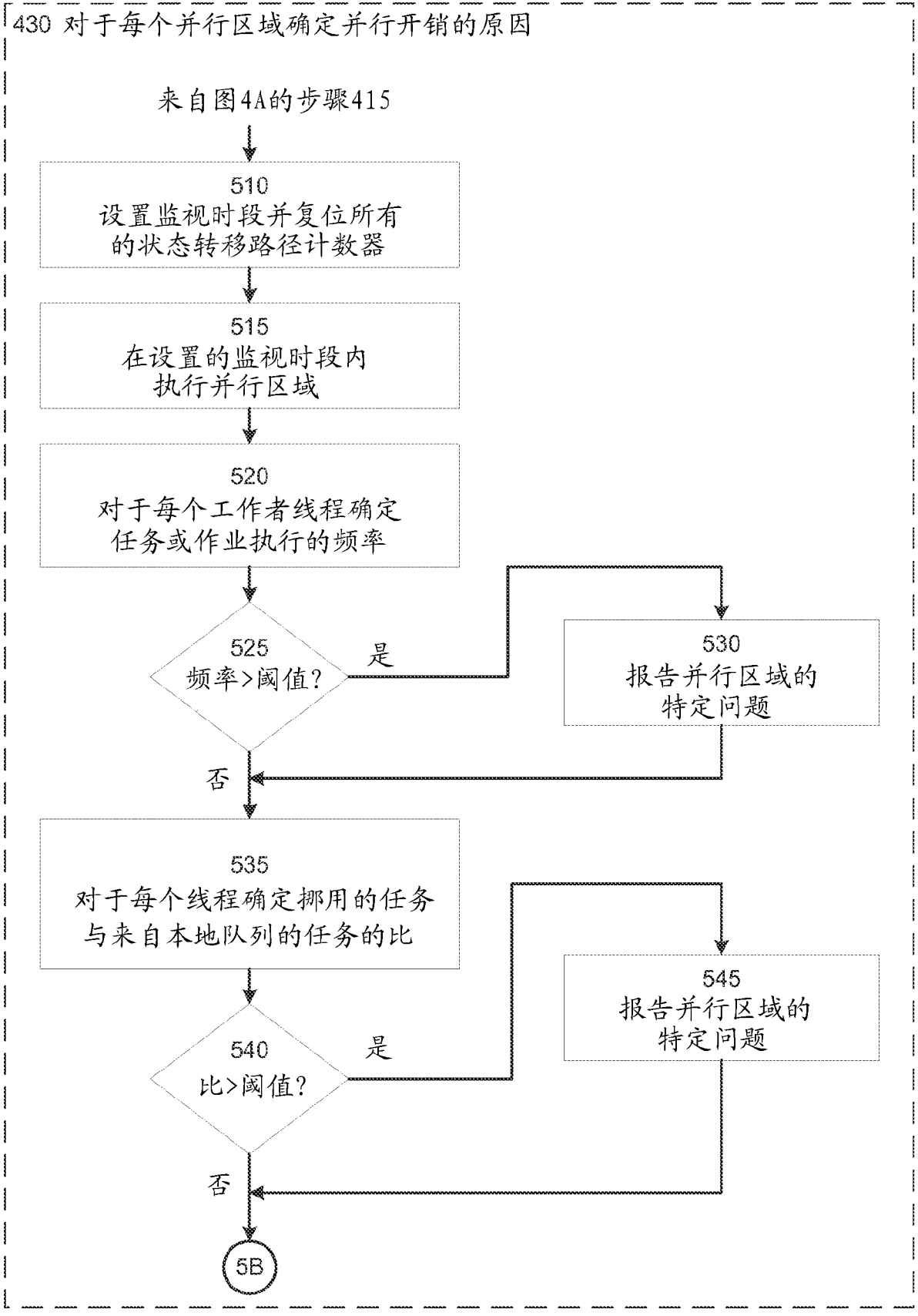


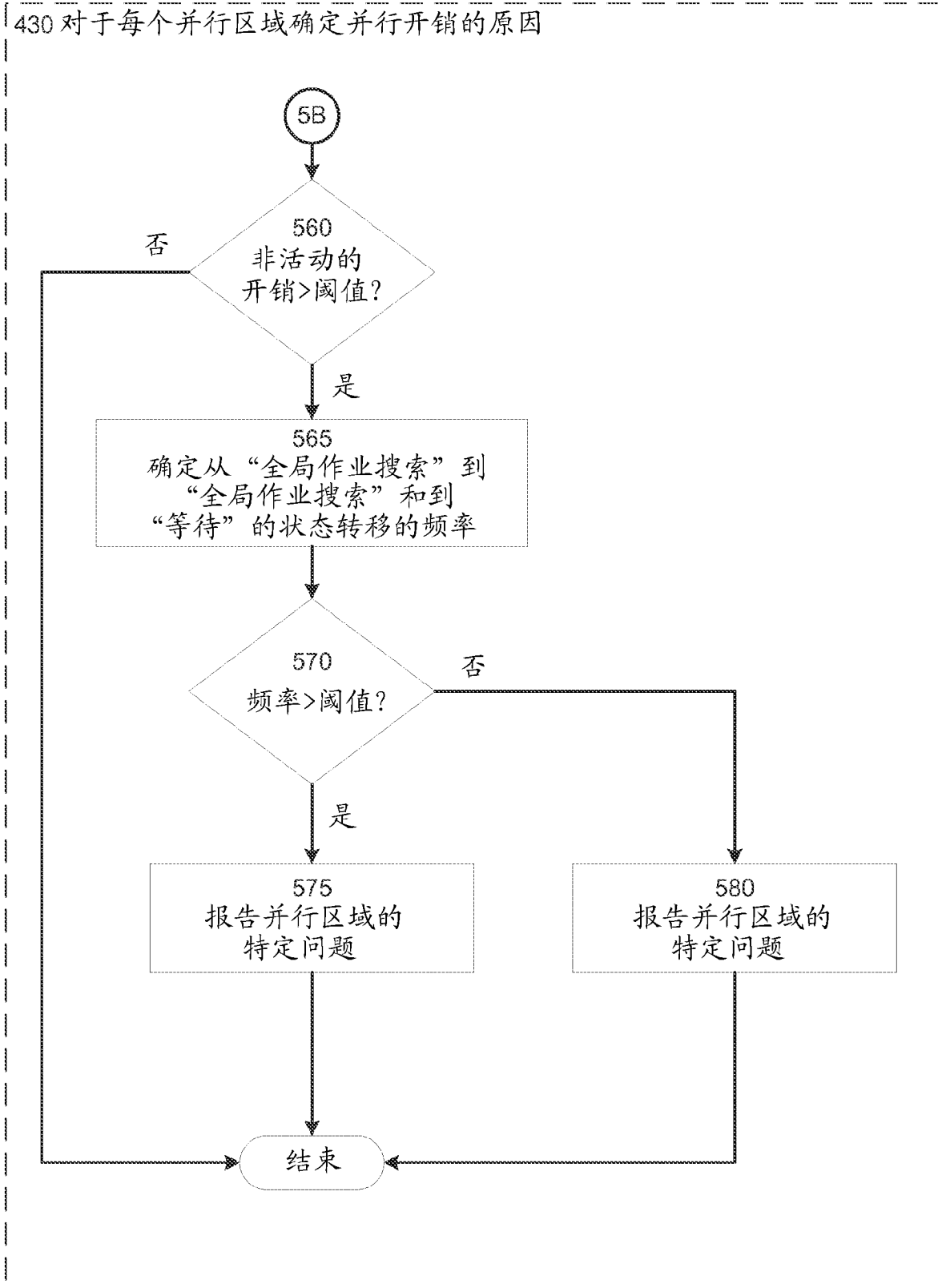
图 4



500

图 5A

430 对于每个并行区域确定并行开销的原因



550

图 5B

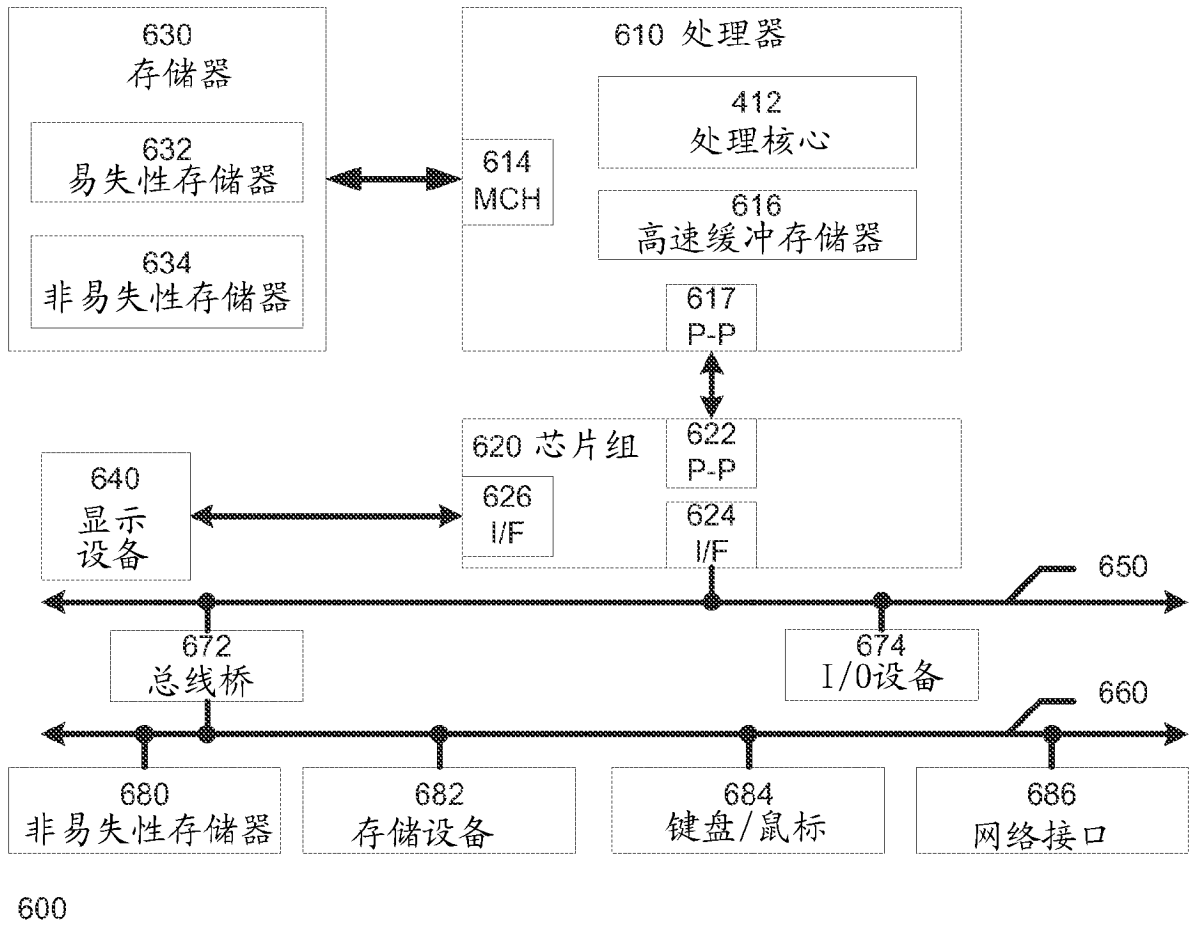


图 6