(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 118733417 A
(43)申请公布日 2024.10.01

(21)申请号 202410981539.3

(22)申请日 2024.07.22

(71)申请人 无锡先进技术研究院
地址 214122 江苏省无锡市滨湖区绣溪路50号KPARK商务中心2号楼

(72)发明人 张杨阳 刘汉旭 赵家熠 尚博文 谢汶兵

(74)专利代理机构 南京纵横知识产权代理有限公司 32224
专利代理师 马进

(51)Int.Cl.
G06F 11/34(2006.01)
G06F 11/30(2006.01)
G06F 9/448(2018.01)

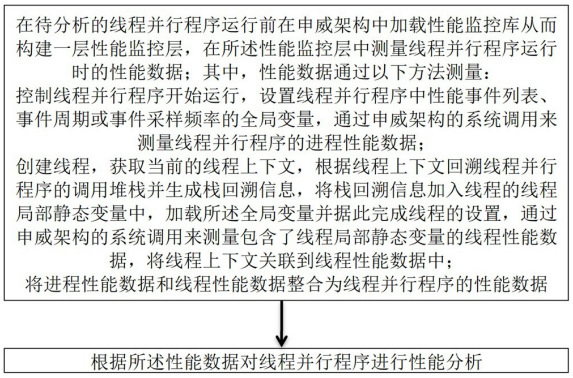
权利要求书2页 说明书7页 附图4页

(54)发明名称

一种基于申威架构的线程并行程序性能分析方法及系统

(57)摘要

本发明公开了一种基于申威架构的线程并行程序性能分析方法及系统,属于程序性能分析技术领域,方法包括在待分析的线程并行程序运行前在申威架构中加载性能监控库从而构建一层性能监控层,在所述性能监控层中测量线程并行程序运行时的性能数据;根据所述性能数据对线程并行程序进行性能分析;本发明通过预加载性能监控库,不影响程序原本代码结构,从而实现性能分析过程的低开销,从而能够将更多性能开销用于对线程并行编程模型的支持和优化。



CN 118733417 A

1.一种基于申威架构的线程并行程序性能分析方法,其特征在于,包括:

在待分析的线程并行程序运行前在申威架构中加载性能监控库从而构建一层性能监控层,在所述性能监控层中测量线程并行程序运行时的性能数据;

根据所述性能数据对线程并行程序进行性能分析;

其中,性能数据通过以下方法测量:

控制线程并行程序开始运行,设置线程并行程序中性能事件列表、事件周期或事件采样频率的全局变量,通过申威架构的系统调用来测量线程并行程序的进程性能数据;

创建线程,获取当前的线程上下文,根据线程上下文回溯线程并行程序的调用堆栈并生成栈回溯信息,将栈回溯信息加入线程的线程局部静态变量中,加载所述全局变量并据此完成线程的设置,通过申威架构的系统调用来测量包含了线程局部静态变量的线程性能数据,将线程上下文关联到线程性能数据中;

将进程性能数据和线程性能数据整合为线程并行程序的性能数据。

2.根据权利要求1所述的基于申威架构的线程并行程序性能分析方法,其特征在于,所述通过申威架构的系统调用来测量线程并行程序的进程性能数据包括:

调用申威架构中的Linux内核内置的性能事件计数器对进程中的各种事件进行计数,将计数结果作为进程性能数据;

其中,所述进程是线程并行程序运行过程中所执行的进程,所述性能事件计数器通过控制和状态寄存器来控制。

3.根据权利要求2所述的基于申威架构的线程并行程序性能分析方法,其特征在于,在计数过程中,若所述性能事件计数器溢出,则计数中断,此时通过事件计数中断处理程序重新设置所述性能事件计数器的初值,所述性能事件计数器从0开始对进程中的各种事件重新进行计数。

4.根据权利要求2所述的基于申威架构的线程并行程序性能分析方法,其特征在于,所述性能事件计数器设有一套或多套,若所述性能事件计数器设有多套则每套性能事件计数器的计数机制都相同。

5.根据权利要求1所述的基于申威架构的线程并行程序性能分析方法,其特征在于,所述根据线程上下文回溯线程并行程序的调用堆栈并生成栈回溯信息,包括:

步骤A、将所述线程上下文针对一个信号帧使用DWARF规则初始化,得到一个指向unw_cursor_t结构的指针;

步骤B、通过所述指针获取程序计数器值并保存,保存的程序计数器值遍历程序加载库模块获得所在库模块id和与所在库模块起始地址的偏移量,根据所述模块id和所述偏移量分析可执行文件中的代码段,从而识别代码段所在函数的起始地址和结束地址信息,确定函数边界;

步骤C、通过所述指针获取前一个函数调用的帧地址,若该帧地址不位于线程的调用堆栈的底部,则返回STOP状态;若该帧地址不位于线程的调用堆栈的底部,且当前调用栈节点信息为空,则返回ERROR状态;若该帧地址不位于线程的调用堆栈的底部,且当前调用栈节点信息不为空,则根据所述线程上下文获取寄存器状态,使用从DWARF标准定义的调用帧信息CFA实现从当前函数调用的上下文信息恢复到前一个函数调用的状态,完成对指针的更新,返回OK状态;

步骤D、若步骤C返回OK状态,则回到步骤B;若步骤C返回ERROR状态,则使用所述指针保存的线程上下文从当前函数调用的上下文信息恢复到前一个函数调用的状态,完成对指针的更新,然后回到步骤B;

步骤F、循环执行步骤D,直至调用堆栈回溯完成,完成后生成栈回溯信息。

6.根据权利要求1所述的基于申威架构的线程并行程序性能分析方法,其特征在于,所述加载所述全局变量并据此完成线程的设置,包括:

根据所述全局变量设置每个线程私有的性能事件列表、事件周期或事件采样频率,每当创建一个线程,遍历该线程私有的性能事件列表,通过申威架构的系统调用来创建当前线程的性能事件计数器事件,若性能事件计数器事件创建成功则返回一个文件描述符,并创建mmap缓冲区以读取文件描述符被操作后产生的才能性能数据样本的集合,然后获取文件描述符的当前标志,将所述当前标志设置为支持异步IO中断响应,若当前标志设置成功则设置该文件描述符的所有者为当前线程,若文件描述符的所有者设置成功则重置该文件描述符的性能事件计数器,最后开启该文件描述符的性能事件计数器,完成线程的设置。

7.根据权利要求1所述的基于申威架构的线程并行程序性能分析方法,其特征在于,在所述测量线程并行程序运行时的性能数据的过程中,当软硬件计数器达到采样阈值时,触发采样中断信号,处理以下步骤:

通过sigaction结构体中的sa_siagction成员设置信号处理句柄,设置信号处理句柄函数的第3个参数void* context,调用生成栈回溯信息的操作,生成栈回溯信息;

通过申威架构的系统调用在采样中断信号触发时输出已测量到的性能数据,并将栈回溯信息关联到性能数据中。

8.一种基于申威架构的线程并行程序性能分析系统,其特征在于,包括:

性能数据测量模块,被配置为:在待分析的线程并行程序运行前在申威架构中加载性能监控库从而构建一层性能监控层,在所述性能监控层中测量线程并行程序运行时的性能数据;

性能分析模块,被配置为:根据所述性能数据对线程并行程序进行性能分析;

其中,性能数据通过以下方法测量:

控制线程并行程序开始运行,设置线程并行程序中性能事件列表、事件周期或事件采样频率的全局变量,通过申威架构的系统调用来测量线程并行程序的进程性能数据;

创建线程,获取当前的线程上下文,根据线程上下文回溯线程并行程序的调用堆栈并生成栈回溯信息,将栈回溯信息加入线程的线程局部静态变量中,加载所述全局变量并据此完成线程的设置,通过申威架构的系统调用来测量包含了线程局部静态变量的线程性能数据,将线程上下文关联到线程性能数据中;

将进程性能数据和线程性能数据整合为线程并行程序的性能数据。

一种基于申威架构的线程并行程序性能分析方法及系统

技术领域

[0001] 本发明涉及一种基于申威架构的线程并行程序性能分析方法及系统,属于程序性能分析技术领域。

背景技术

[0002] 随着计算需求的增长,多核处理器快速发展,推动了线程并行编程模型的创新。线程并行是一种并行计算的形式,它允许多个线程同时执行代码,从而提高程序的执行效率。线程是操作系统能够进行运行调度的最小单位,它被包含在进程之中,是进程中的实际运行单位。线程并行通常在共享内存模型中实现,其中所有线程共享同一块内存空间,这样线程之间就可以轻松地共享数据和通信,但也需要同步机制(如互斥锁mutexes、信号量semaphores和屏障barriers)来避免竞争条件和数据不一致。在多核处理器上,线程并行可以充分利用多核处理器的计算能力,每个核心可以独立执行一个或多个线程,从而实现真正的并行计算,提高系统资源的使用效率,显著加速应用程序的总执行时间。

[0003] 目前线程并行采样性能分析对高性能计算十分重要,然而申威架构的传统性能调优工具,如perf、OProfile等适用于通用型性能分析任务并不强调线程并行程序的深入分析,且对线程并行编程模型没有很好的支持和优化,在测量大规模线程并行计算应用程序时会产生大量性能开销。

发明内容

[0004] 本发明的目的在于提供一种基于申威架构的线程并行程序性能分析方法及系统,解决现有技术中存在的性能开销大的问题。

[0005] 为实现以上目的,本发明是采用下述技术方案实现的:

第一方面,本发明提供了一种基于申威架构的线程并行程序性能分析方法,包括:

在待分析的线程并行程序运行前在申威架构中加载性能监控库从而构建一层性能监控层,在所述性能监控层中测量线程并行程序运行时的性能数据;

根据所述性能数据对线程并行程序进行性能分析;

其中,性能数据通过以下方法测量:

控制线程并行程序开始运行,设置线程并行程序中性能事件列表、事件周期或事件采样频率的全局变量,通过申威架构的系统调用来测量线程并行程序的进程性能数据;

创建线程,获取当前的线程上下文,根据线程上下文回溯线程并行程序的调用堆栈并生成栈回溯信息,将栈回溯信息加入线程的线程局部静态变量中,加载所述全局变量并据此完成线程的设置,通过申威架构的系统调用来测量包含了线程局部静态变量的线程性能数据,将线程上下文关联到线程性能数据中;

将进程性能数据和线程性能数据整合为线程并行程序的性能数据。

[0006] 进一步的,所述通过申威架构的系统调用来测量线程并行程序的进程性能数据包括:

调用申威架构中的Linux内核内置的性能事件计数器对进程中的各种事件进行计数,将计数结果作为进程性能数据;

其中,所述进程是线程并行程序运行过程中所执行的进程,所述性能事件计数器通过控制和状态寄存器来控制。

[0007] 进一步的,在计数过程中,若所述性能事件计数器溢出,则计数中断,此时通过事件计数中断处理程序重新设置所述性能事件计数器的初值,所述性能事件计数器从0开始对进程中的各种事件重新进行计数。

[0008] 进一步的,所述性能事件计数器设有一套或多套,若所述性能事件计数器设有多套则每套性能事件计数器的计数机制都相同。

[0009] 进一步的,所述根据线程上下文回溯线程并行程序的调用堆栈并生成栈回溯信息,包括:

步骤A、将所述线程上下文针对一个信号帧使用DWARF规则初始化,得到一个指向unw_cursor_t结构的指针;

步骤B、通过所述指针获取程序计数器值并保存,保存的程序计数器值遍历程序加载库模块获得所在库模块id和与所在库模块起始地址的偏移量,根据所述模块id和所述偏移量分析可执行文件中的代码段,从而识别代码段所在函数的起始地址和结束地址信息,确定函数边界;

步骤C、通过所述指针获取前一个函数调用的帧地址,若该帧地址不位于线程的调用堆栈的底部,则返回STOP状态;若该帧地址不位于线程的调用堆栈的底部,且当前调用栈节点信息为空,则返回ERROR状态;若该帧地址不位于线程的调用堆栈的底部,且当前调用栈节点信息不为空,则根据所述线程上下文获取寄存器状态,使用从DWARF标准定义的调用帧信息CFA实现从当前函数调用的上下文信息恢复到前一个函数调用的状态,完成对指针的更新,返回OK状态;

步骤D、若步骤C返回OK状态,则回到步骤B;若步骤C返回ERROR状态,则使用所述指针保存的线程上下文从当前函数调用的上下文信息恢复到前一个函数调用的状态,完成对指针的更新,然后回到步骤B;

步骤F、循环执行步骤D,直至调用堆栈回溯完成,完成后生成栈回溯信息。

[0010] 进一步的,所述加载所述全局变量并据此完成线程的设置,包括:

根据所述全局变量设置每个线程私有的性能事件列表、事件周期或事件采样频率,每当创建一个线程,遍历该线程私有的性能事件列表,通过申威架构的系统调用来创建当前线程的性能事件计数器事件,若性能事件计数器事件创建成功则返回一个文件描述符,并创建mmap缓冲区以读取文件描述符被操作后产生的才能性能数据样本的集合,然后获取文件描述符的当前标志,将所述当前标志设置为支持异步IO中断响应,若当前标志设置成功则设置该文件描述符的所有者为当前线程,若文件描述符的所有者设置成功则重置该文件描述符的性能事件计数器,最后开启该文件描述符的性能事件计数器,完成线程的设置。

[0011] 进一步的,在所述测量线程并行程序运行时的性能数据的过程中,当软硬件计数器达到采样阈值时,触发采样中断信号,处理以下步骤:

通过sigaction结构体中的sa_siagction成员设置信号处理句柄,设置信号处理

句柄函数的第3个参数void* context,调用生成栈回溯信息的操作,生成栈回溯信息;

通过申威架构的系统调用在采样中断信号触发时输出已测量到的性能数据,并将栈回溯信息关联到性能数据中。

[0012] 第二方面,本发明提供了一种基于申威架构的线程并行程序性能分析系统,包括:

性能数据测量模块,被配置为:在待分析的线程并行程序运行前在申威架构中加载性能监控库从而构建一层性能监控层,在所述性能监控层中测量线程并行程序运行时的性能数据;

性能分析模块,被配置为:根据所述性能数据对线程并行程序进行性能分析;

其中,性能数据通过以下方法测量:

控制线程并行程序开始运行,设置线程并行程序中性能事件列表、事件周期或事件采样频率的全局变量,通过申威架构的系统调用来测量线程并行程序的进程性能数据;

创建线程,获取当前的线程上下文,根据线程上下文回溯线程并行程序的调用堆栈并生成栈回溯信息,将栈回溯信息加入线程的线程局部静态变量中,加载所述全局变量并据此完成线程的设置,通过申威架构的系统调用来测量包含了线程局部静态变量的线程性能数据,将线程上下文关联到线程性能数据中;

将进程性能数据和线程性能数据整合为线程并行程序的性能数据。

[0013] 与现有技术相比,本发明所达到的有益效果是:

本发明提供的一种基于申威架构的线程并行程序性能分析方法及系统,通过预加载性能监控库,不影响程序原本代码结构,从而实现性能分析过程的低开销,从而能够将更多性能开销用于对线程并行编程模型的支持和优化。

附图说明

[0014] 图1是本发明实施例提供的一种基于申威架构的线程并行程序性能分析方法的流程图之一;

图2是本发明实施例提供的一种基于申威架构的线程并行程序性能分析方法的流程图之二;

图3是本发明实施例提供的线程并行部分性能监控流程图;

图4是本发明实施例提供的内存中用户地址空间映射关系图。

具体实施方式

[0015] 下面结合附图对本发明作进一步描述,以下实施例仅用于更加清楚地说明本发明的技术方案,而不能以此来限制本发明的保护范围。

[0016] 实施例1。

[0017] 如图1所示,本发明提供了一种基于申威架构的线程并行程序性能分析方法,包括:

S1、在待分析的线程并行程序运行前在申威架构中加载性能监控库从而构建一层性能监控层,在所述性能监控层中测量线程并行程序运行时的性能数据。

[0018] 步骤S1包括步骤a、步骤b、步骤c、步骤d、步骤e、步骤f。

[0019] 步骤a:被分析的线程并行程序运行前在申威架构中使用LD_PRELOAD预加载性能

监控库(如图2所示),在不影响应用程序本身代码结构的前提下,使用该性能监控库在应用程序运行期间嵌入性能测量代码。覆写程序运行时的关键函数基本初始化操作并提供性能测量接口,设置采样中断信号和中断处理操作。

[0020] LD_PRELOAD是 Linux/Unix 系统的一个环境变量,它影响程序的运行时的链接(Runtime linker),它允许在程序运行前定义优先加载的动态链接库。这个功能主要就是用来有选择性的载入不同动态链接库中的相同函数。通过这个环境变量,可以在主程序和其动态链接库的中间加载别的动态链接库,甚至覆盖正常的函数库。

[0021] 步骤b:如图2所示,覆写程序启动入口函数获取程序启动入口控制权,嵌入进程性能测量代码,控制线程并行程序开始运行,设置性能事件列表、事件周期/事件采样频率的全局变量,开启进程的文件描述符对应的性能事件计数器,通过申威架构的系统调用来采样测量进程性能数据。

[0022] 步骤c:如图2所示,覆写线程创建函数获取线程创建控制权,创建线程,嵌入线程性能测量代码,获取当前的线程上下文,回溯线程并行程序的调用堆栈,生成栈回溯信息,将栈回溯信息加入线程局部静态变量中,加载已设置的性能事件列表、事件周期/事件采样频率的全局变量设置线程的相关变量,开启每个线程的文件描述符对应的性能事件计数器,通过申威架构的系统调用来采样测量线程性能数据并与线程上下文相关联。

[0023] 步骤c中生成栈回溯信息详细分为如下步骤:

(1)覆写线程创建函数,在创建线程时获取当前线程所在帧的上下文(即当前的线程上下文),线程上下文包括申威架构调用线程的CPU寄存器集(申威32个64位整数寄存器和32个64位浮点寄存器)和程序计数器的值、当前上下文的栈指针信息(包含栈指针寄存器R30的值)和屏蔽信号集(包含申威架构支持的所有信号,例如子进程退出信号20、进程继续信号19)等。

[0024] (2)该线程上下文使用DWARF规则初始化一个指向unw_cursor_t结构的指针,并明确指出该初始化是针对一个信号帧,即当程序接收到信号时,操作系统会在程序的当前栈上插入一个信号帧,这个信号帧包括了信号处理之前的程序上下文。

[0025] unw_cursor_t结构的代码为:

```
typedef struct unw_cursor
{
   //申威架构UNW_TDEP_CURSOR_LEN为4096
   unw_word_t opaque[UNW_TDEP_CURSOR_LEN];
}unw_cursor_t;
```

DWARF:Debugging With Attributed Record Formats-使用属性化记录格式进行调试,是一种用于调试信息的标准格式,通常与编译器一起使用。它提供了一种有效的方法来生成、存储和访问程序符号和调试信息,这些信息可以在程序崩溃或其他错误情况下帮助开发人员调试分析代码。

[0026] (3)该指针获取保存的程序计数器值并将其保存。保存的程序计数器值遍历程序加载库模块(可执行库和共享库)获得所在库模块id和与库模块起始地址的偏移量,分析可执行文件中的代码段识别代码段所在函数的起始和结束地址信息,确定函数边界。

[0027] (4)定义指向unw_cursor_t结构的指针获取申威架构链接寄存器(R26)的值,即前

一个函数调用的帧地址,使用该帧地址判断该地址是否位于线程调用堆栈的底部,如果不是则返回STOP状态。如果该地址不在线程调用堆栈的底部,判断当前调用栈节点信息是否为空,若为空则返回ERROR状态,否则根据当前帧的上下文获取寄存器状态,使用从DWARF标准定义的调用帧信息CFA实现从当前函数调用的上下文信息恢复到前一个函数调用的状态,更新该指针。

[0028] (5)在第4步操作成功后返回OK状态,重复第3步操作。若第4步返回ERROR状态,则使用unw_cursor_t结构指针保存的上下文信息从当前函数调用的上下文信息恢复到前一个函数调用的状态,更新该指针,操作成功后,重复第3步操作。

[0029] (6)第5步和第6步返回操作成功,则返回第5步循环,直到堆栈回溯完成。完成后生成栈回溯信息,将其加入线程局部静态变量中。

[0030] 如图3所示,步骤c中覆写线程创建函数获取线程创建控制权,设置线程的相关变量,包括:获取步骤b中已设置的全局变量设置每个线程私有的性能事件列表、事件周期/事件采样频率。每当创建一个线程,遍历该线程私有性能事件列表,通过系统调用创建当前线程的性能事件计数器事件,创建成功则返回一个文件描述符,并创建mmap缓冲区以准备读取文件描述符被操作后产生的采样性能数据样本的集合。接着获取文件描述符当前标志,并将其设置为支持异步IO中断响应,如果设置成功,获取当前线程号,设置该文件描述符所有者为当前线程。如果设置成功,重置该文件描述符的性能事件计数器。最后开启该文件描述符的性能事件计数器,开始监控性能数据。

[0031] 步骤b、步骤c中通过系统调用来采样测量进程性能数据和线程性能数据,是由Linux内核内置的性能事件计数器框架负责,核心设置5套通用性能事件计数器,每套计数器的计数机制相同,可选择其中一套或多套计数器同时计数。在相关控制和状态寄存器的控制下,实现对核心内部各种事件的计数,性能事件计数器溢出可产生性能采样中断。性能事件计数器的初始值、计数使能控制、计数的事件选择都通过控制和状态寄存器来控制。系统通过事件计数中断处理程序,实现事件采样功能。与通用事件计数相关的控制和状态寄存器如表1所示:

表1-控制和状态寄存器信息表

名称	含义
INT_EN	所有事件计数中断使能
INT_STAT	所有事件计数中断状态
INT_CLR	所有事件计数中断状态清除
PC_INT	性能计数中断状态寄存器
PCx_CTL	选择性能事件计数器 x 的计数事件、事件计数器 x 溢出中断使能、事件计数器 x 计数使能
PCx_VALUE	事件计数器 x 的计数值

[0032] 通用事件计数一般流程如下:

配置PCx_VALUE设置计数初值;

配置PCx_CTL指定计数事件、计数模式、中断使能和计数使能;

性能事件计数器计数:事件计数使能打开后,硬件就会根据相应的控制和状态寄存器选择的计数事件,每发生一次控制和状态寄存器所选择的事件,PCx_VALUE的值加一;

计数器溢出:当PCx_VALUE中的计数器溢出时,若事件计数PCx_CTL中断使能打开,则产生性能计数中断请求,硬件置PC_INT对应中断状态位有效;

记录中断状态:硬件根据PC_INT的所有中断状态,设置INT_STAT对应位有效;

硬件中断:若INT_EN对应位有效,核心响应性能计数中断,进入事件计数中断处理程序;

中断处理:性能事件计数器溢出后从0开始重新计数,中断处理程序重新设置性能事件计数器的初值。

[0033] 性能事件计数器框架除支持设置采样阈值外,还支持设置采样频率即每秒采样多少次。

[0034] 步骤a、步骤b和步骤c中,性能数据采样设置采样中断信号PERF_SIGNAL,使用sigaction结构体设置信号处理器,当软硬件计数器达到采样阀值时,触发中断信号处理如下步骤:

通过sigaction结构体中的sa_siagction成员设置信号处理句柄,设置信号处理句柄函数的第3个参数void* context,调用生成栈回溯信息的操作,生成栈回溯信息;

通过申威架构的系统调用在采样中断信号触发时输出已测量到的性能数据,并将栈回溯信息关联到性能数据中。

[0035] 步骤d:如图2所示,覆写线程退出函数获取线程退出控制权,线程并行结束,关闭每个线程的性能事件计数器,退出线程测量程序。

[0036] 步骤e:如图2所示,覆写程序退出函数获取程序退出控制权,关闭所有性能测量代码,输出线程并行程序的性能数据(由进程性能数据和线程性能数据整合得到)。

[0037] 在本实施例中,内存中用户地址空间映射关系如图4所示。

[0038] S2、根据所述性能数据对线程并行程序进行性能分析。

[0039] 根据步骤S1测量到的性能数据对线程并行程序进行性能分析,得到性能分析结果。

[0040] 实施例2。

[0041] 本发明提供了一种基于申威架构的线程并行程序性能分析系统,包括:

性能数据测量模块,被配置为:在待分析的线程并行程序运行前在申威架构中加载性能监控库从而构建一层性能监控层,在所述性能监控层中测量线程并行程序运行时的性能数据;

性能分析模块,被配置为:根据所述性能数据对线程并行程序进行性能分析;

其中,性能数据通过以下方法测量:

控制线程并行程序开始运行,设置线程并行程序中性能事件列表、事件周期或事件采样频率的全局变量,通过申威架构的系统调用来测量线程并行程序的进程性能数据;

创建线程,获取当前的线程上下文,根据线程上下文回溯线程并行程序的调用堆栈并生成栈回溯信息,将栈回溯信息加入线程的线程局部静态变量中,加载所述全局变量并据此完成线程的设置,通过申威架构的系统调用来测量包含了线程局部静态变量的线程性能数据,将线程上下文关联到线程性能数据中;

将进程性能数据和线程性能数据整合为线程并行程序的性能数据。

[0042] 本领域内的技术人员应明白,本申请的实施例可提供为方法、系统、或计算机程序产品。因此,本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且,本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

[0043] 本申请是参照根据本申请实施例的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器,使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0044] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中,使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品,该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0045] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上,使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理,从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0046] 以上所述仅是本发明的优选实施方式,应当指出,对于本技术领域的普通技术人员来说,在不脱离本发明技术原理的前提下,还可以做出若干改进和变形,这些改进和变形也应视为本发明的保护范围。

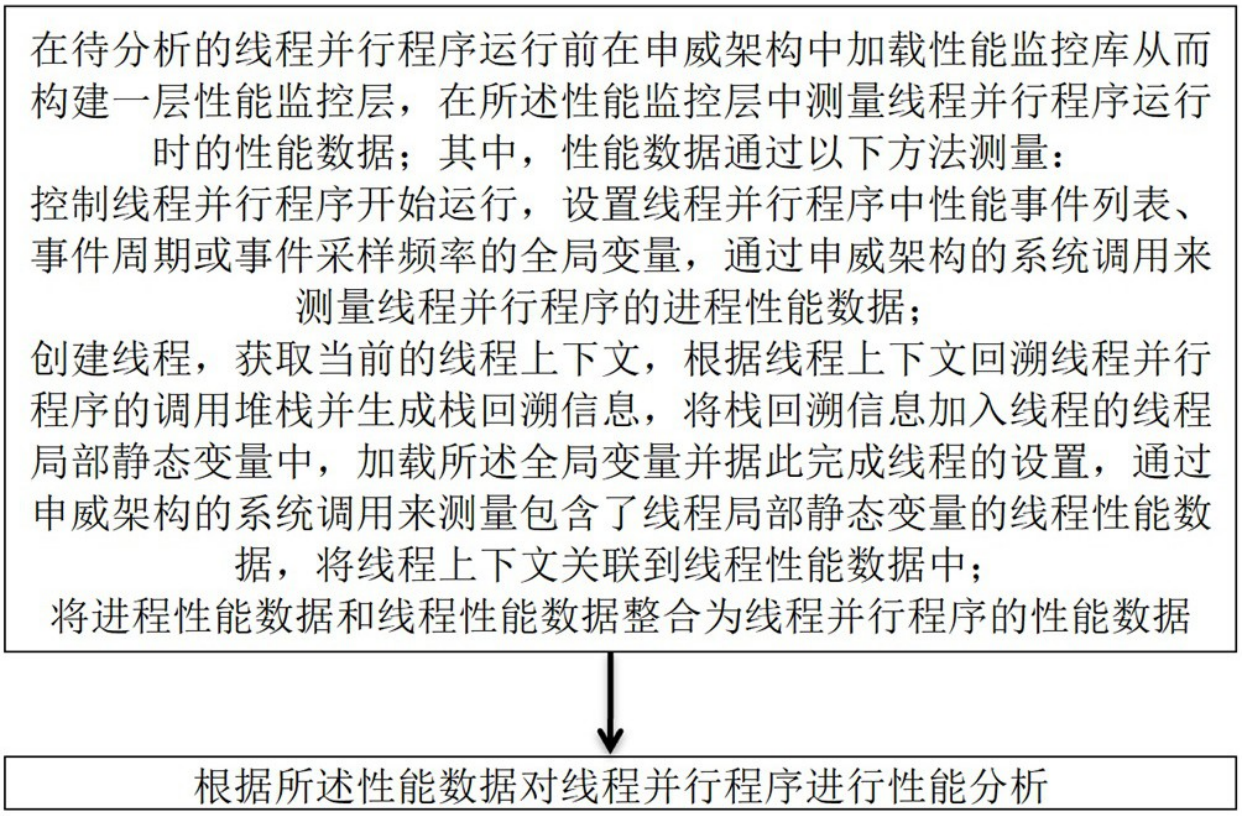

在待分析的线程并行程序运行前在申威架构中加载性能监控库从而构建一层性能监控层，在所述性能监控层中测量线程并行程序运行时的性能数据；其中，性能数据通过以下方法测量：
控制线程并行程序开始运行，设置线程并行程序中性能事件列表、事件周期或事件采样频率的全局变量，通过申威架构的系统调用来测量线程并行程序的进程性能数据；
创建线程，获取当前的线程上下文，根据线程上下文回溯线程并行程序的调用堆栈并生成栈回溯信息，将栈回溯信息加入线程的线程局部静态变量中，加载所述全局变量并据此完成线程的设置，通过申威架构的系统调用来测量包含了线程局部静态变量的线程性能数据，将线程上下文关联到线程性能数据中；
将进程性能数据和线程性能数据整合为线程并行程序的性能数据
根据所述性能数据对线程并行程序进行性能分析

图 1

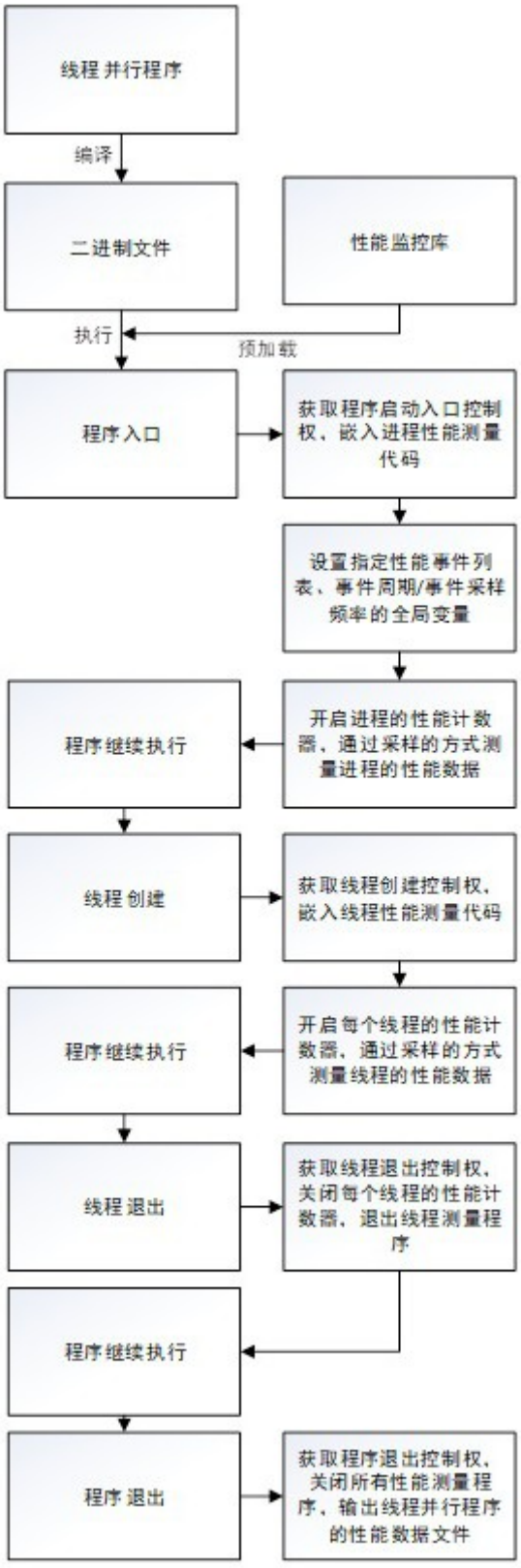

线程并行程序
编译
二进制文件
性能监控库
执行
预加载
程序入口
获取程序启动入口控制权，嵌入进程性能测量代码
设置指定性能事件列表、事件周期/事件采样频率的全局变量
开启进程的性能计数器，通过采样的方式测量进程的性能数据
程序继续执行
线程创建
获取线程创建控制权，嵌入线程性能测量代码
开启每个线程的性能计数器，通过采样的方式测量线程的性能数据
程序继续执行
线程退出
获取线程退出控制权，关闭每个线程的性能计数器，退出线程测量程序
程序继续执行
程序退出
获取程序退出控制权，关闭所有性能测量程序，输出线程并行程序的性能数据文件

图 2

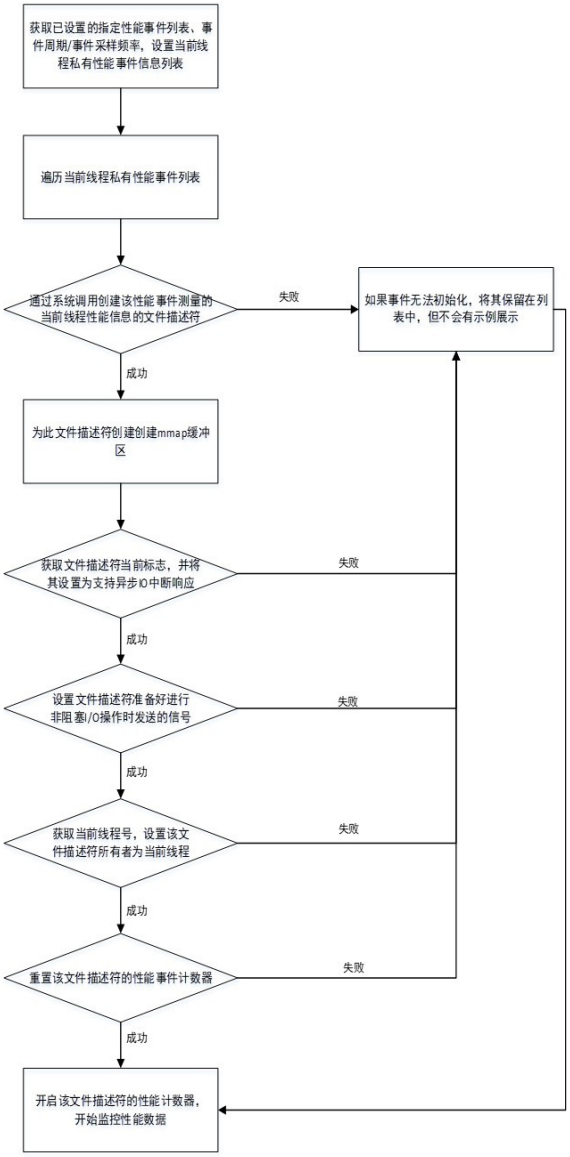

获取已设置的指定性能事件列表、事件周期/事件采样频率，设置当前线程私有性能事件信息列表
遍历当前线程私有性能事件列表
通过系统调用创建该性能事件测量的当前线程性能信息的文件描述符
失败
如果事件无法初始化，将其保留在列表中，但不会有示例展示
成功
为此文件描述符创建创建mmap缓冲区
获取文件描述符当前标志，并将其设置为支持异步IO中断响应
失败
成功
设置文件描述符准备好进行非阻塞I/O操作时发送的信号
失败
成功
获取当前线程号，设置该文件描述符所有者为当前线程
失败
成功
重置该文件描述符的性能事件计数器
失败
成功
开启该文件描述符的性能计数器，开始监控性能数据

图 3

图 4