

一种基于访存依赖对的并行重放方法

应 欢^{1,2}, 王东辉¹, 王 喆^{2,3}

(1 中国科学院 声学研究所, 北京 100049; 2 中国科学院大学, 北京 100190;
3 中国科学院 计算技术研究所, 北京 100190)

摘 要: 确定性重放技术能够为并行程序提供复现执行的功能. 为了加速重放执行, 本文提出一种基于访存依赖对的并行重放方法, 该方法通过离线分析记录阶段生成的日志文件, 识别出日志文件中具有访存依赖关系的日志项, 在重放执行时仅按序执行上述日志项. 采用 PARSEC 测试集进行性能评估, 实验结果表明该方法在重放阶段引入的开销较小.

关键词: 并行程序; 共享内存; 确定性重放; 拓扑图

中图分类号: TP368

文献标识码: A

文章编号: 1000-7180(2017)01-0006-06

An Efficient Parallel Replay Method Based on Shared Memory Dependency Pair

YING Huan^{1,2}, WANG Dong-hui¹, WANG Zhe^{2,3}

(1 Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China;

2 University of Chinese Academy of Sciences, Beijing 100049, China;

3 Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Deterministic replay provides the possibility of reproducing parallel program execution. In order to get excellent performance in replay-runs, this paper proposes a parallel replay method based on shared memory dependency pair, which conducted by offline analysis of original log generated in record-runs to recognise the record items with memory dependency relationship, and each thread only executes these record items in order in replay-runs. Performance evaluation using PARSEC benchmark shows that this method introduces little overhead in replay-runs.

Key words: parallel program; shared memory; deterministic replay; topological graph

1 引言

随着多核处理器的流行, 并行编程成为多核平台上最有效的编程方式^[1], 是未来软件开发的主流趋势. 然而, 由于并行程序本身具有不确定性, 很难在调试过程中重现程序中的并发错误. 为了缓解程序员开发与调试并行程序的负担, LeBlanc 等人^[2]首次提出了确定性重放技术, 目前, 有关确定性重放技术的研究工作主要集中在如何低开销地记录并行程序的不确定性因素: 软件实现

的确定性重放方法尽量引入较小的运行时开销^[3], 硬件实现的确定性重放方法尽量不影响目前的处理器架构, 且生成的日志文件尽量小^[4]. 相反地, 并行程序在重放执行阶段的性能往往没有引起研究者足够的重视^[5].

实际上, 重放阶段的性能对确定性重放技术的应用场景有很大的影响. 在并行程序调试领域, 重放执行的速度严重影响了程序员调试程序时的用户体验, 程序员当然希望能够快速重放并行程序的执行, 从而能够快速收敛调试过程.

2 相关背景

InstantReplay^[2]、PinPlay^[6]、iDNA^[7]等研究工作通过插桩将并程序序中所有访存操作按序记录在全局日志文件中;在重放阶段逐条从日志文件中读取日志项,在每个访存操作执行前都判断当前执行是否与日志项匹配,重放执阶段性能为记录阶段性能的 3-4X。

UPLAY^[8]是中国科学院计算技术研究所体系结构国家重点实验室实现的低开销的确定性重放系统。该系统以页为粒度,通过记录并程序序中所有共享页面的访问权限在各线程间的转移顺序,来反映并程序序对共享内存的访问顺序。与其他软件实现的确定性重放系统的重放策略类似,UPLAY 采用串行匹配的方式实现重放。这种基于全局序的重放匹配策略效率非常低下:对于日志文件中的某一段连续日志项来说,无论它们是否相关,每个“线程”(“线程”特指 UPLAY 机制下并程序序中被替换成进程的线程)都必须按序匹配执行这些日志项中包含的操作信息,导致并程序序在重放执行时经常出现某些“线程”停顿。

3 基本思想

为了加速重放执行,提出基于访存依赖对的并行重放方法 FaPlay。其主要思想是并程序序在记录执行阶段通过同步“线程”间的操作建立的全局逻辑序关系中,存在很多具有冗余序关系的日志项,而这些日志项实际上是可以并行执行。如果能够识别出日志文件中不可并行的日志项,在重放执行时仅仅串行执行存在依赖关系的日志项,则能够大大提高重放执行的效率。虽然离线分析日志文件也会引入开销,然而离线分析一次日志文件的开销会均摊到多次重放过程中。FaPlay 的总体设计示意图如图 1 所示,下面以 UPLAY 为例,来具体介绍 FaPlay 中的各个环节。

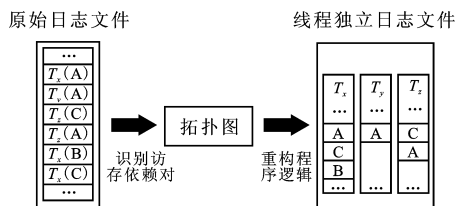


图 1 FaPlay 的总体设计

4 识别访存依赖对

并程序序中的访存竞争指多个线程同时访问同一个地址,且至少有一次访问是写操作。在以页为粒

度的确定性重放方法中,访存竞争指多个线程同时请求获得同一个共享页面的访问权限,且至少一个线程企图写访问该页面。定义 UPLAY 系统中的访存依赖对为不同“线程”对同一个共享页面的连续两次存在数据依赖的访问请求:当共享页面 P 的当前状态为 SHARED_READ,而此时其他“线程”请求获得 P 的写访问权限;当 P 的当前状态为 OWNED_WRITE,而此时多个线程均请求获得 P 的读访问权限;当 P 的当前状态为 OWNED_WRITE,而此时其他“线程”也请求获得 P 的写访问权限。这三种访存依赖对分别表示为 $RP_{x,y,z} \xrightarrow{E} WP_m, WP_x \xrightarrow{E} RP_{y,z,m}, WP_x \xrightarrow{E} WP_y$ 。

UPLAY 基于页保护机制来捕获并程序序对共享内存的访问。当“线程”由于缺少对某个共享页面的访问权限时会触发页保护异常,基于 CREW 协议^[9],如果该“线程”能够顺利获得该共享页面的访问权限,则仅修改全局页面权限表及在本进程空间设置相应的访问权限,并返回;否则,需要按序执行以下操作:加入等待队列、在全局公告板上注册自己的访问请求、阻塞等待、被持有访问权限的“线程”唤醒、从全局公告板上删除自己的访问请求、从等待队列中出队、再次查询 CREW 协议直到成功获得对该页面的访问权限。UPLAY 在“线程”查询 CREW 协议失败的情况下,将上述按序执行的操作均记录在日志文件中。然而,实际上只有页面异常处理中查询 CREW 协议并成功获得共享页面访问权限的事件,才真实反映共享页面的访问权限在“线程”间的转移顺序。

为了从具有全局逻辑序的原有日志文件中识别上述三种访存依赖对,首先给出访问权限转移区间的概念。

定义 1 访问权限的转移区间(Transfer Interval for Permission):对于某个共享页面 P ,其访问权限在两个或多个“线程”间的一次转移称为共享页面 P 的访问权限转移区间。其中,该次转移只能是以下三种情况之一:读访问权限转移到写访问权限、写访问权限转移到读访问权限、写访问权限转移到写访问权限。

更确切地说,共享页面的访问权限的转移,是一个持续动作:以该共享页面的所有持有者释放该共享页面为起点,以请求获得该页面的等待者成功获得该共享页面的访问权限为终点。因此,仅需要识别出日志文件中以下三种访存依赖对: $GP_{R,x,y,z} \xrightarrow{R} WP_m, GP_{W_x} \xrightarrow{R} RP_{y,z,m}, GP_{W_{t_x}} \xrightarrow{R} WP_y$ (G 指 UP-

LAY 记录的“线程”释放事件)。

图 2 列出 UPLAY 中与共享页面访问权限转移相关的几种日志项(在下文中相关日志项均用图中的标识符替代)。对于“线程” x 的日志项来说,定义“线程” x 获得共享页面 P 的访问权限的日志项模式为 $ai_x(P)$ 。其中,“线程” x 首次成功获得共享页面 P 的访问权限的日志项为 $A x_{iR}(P)$, 标记为模式 μx_{ix} ;“线程” x 首次申请没有成功,经过阻塞等待后最终获得 P 的读访问权限的日志项序列为 $A x_{iR}(P) \xrightarrow{R} B x_{iR}(P) \xrightarrow{R} C x_{iR}(P) \xrightarrow{R} D x_{iR}(P) \xrightarrow{R} A x_{iR}(P)$, 标记为模式 νx , 则 $\alpha = \mu$ or ν 。定义“线程” x 释放共享页面 P 的访问权限的日志项模式为 $\beta_x(P)$, 可以是主动释放 P , 标记为 θx , 也可以是在执行阻塞型函数期间被动释放 P , 标记为 ρx , 即 $\beta = \theta$ or ρ 。其中, $i = R$ or W 。在一个转移区间内,对于同一个“线程” x 来说,释放共享页面日志项 β 是获得共享页面日志项 α 的后继。

获取共享页面访问权限		释放共享页面访问权限	
标识符	记录项类型	标识符	记录项类型
A_x	REQUEST_PERMISSION_1	E_x	GIVEUP_PERMISSION_AS_DEMAND
B_x	REGISTER_WAIT_PAGE		
C_x	DELETE_WAIT_PAGE	F_x	PREEMPT_PERMISSION
D_x	REQUEST_PERMISSION_2		

图 2 共享页面访问权限转移相关的日志项

基于上述对“线程”获得共享页面访问权限的日志项模式和释放共享页面访问权限的日志项模式的定义,提出了基于转移区间的拓扑图建立算法 1,如下所示。

Algorithm 1 Build Transfer Interval Topology

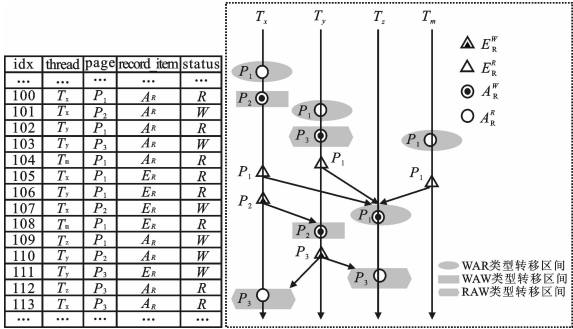
Input:
 record item list, LIST;

Output:
 transfer interval topology;

```
1: while (1) then
2:    $x = \text{SelectRequestItem}(\text{LIST});$  // select next request permission record item
3:   if  $x$  is NULL then
4:     return; // record file scanning is completed
5:   forin Queue then // Queue is working_queue stores request permission item
6:     if  $(x.\text{page\_addr} == .\text{page\_addr})$  is not true then
7:       Queue.add( $x$ ); // another transfer interval is started
8:       continue;
9:     if  $(.state == R) \&\& (.state == R)$  is true then
10:      BuildParallelRelation( $x$ ); // threads share READ page
11:     elseif  $(.state == R) \&\& (.state == W)$  is true then
12:      CompleteTransferInterval( $x$ , WAR);
13:      Queue.remove();
14:      Queue.add( $x$ ); // another transfer interval is started
15:     elseif  $(.state == W) \&\& (.state == R)$  is true then
16:       if IsBuiltDependencyRelation() is true then
17:         BuildParallelRelation( $x$ ); // threads shared
```

```
READ page
18:   else
19:     BuildDependencyRelation( $x$ , RAW); // first thread owned READ
20:   Queue.add( $x$ ); // another transfer interval is started
21:   elseif  $(.state == W) \&\& (.state == W)$  is true then
22:     Queue.add( $x$ ); // another transfer interval is started
23:     Queue.remove();
24:     if IsBuiltDependencyRelation() is true then
25:       CompleteTransferInterval( $x$ , NULL, RAW); // last transfer interval
26:       CompleteTransferInterval( $x$ , WAR); // current transfer interval
27:       Queue.remove();
28:     else
29:       CompleteTransferInterval( $x$ , WAW);
30:     continue;
```

在建立基于转移区间的拓扑图的过程中,实际上也识别出日志文件中的所有访存依赖对 $GP_{R,x,y,z} \xrightarrow{R} WP_m, GP_{W,x} \xrightarrow{R} RP_{y,z,m}, GP_{W,x} \xrightarrow{R} WP_y$ 。如图 3 所示,图 3(a)为原有日志文件中的一部分(只列出了与共享页面获取与释放相关的日志项),其中 idx 为日志项索引, $thread$ 为线程号, $page$ 指共享页面地址, $record_item$ 指日志项的类型, $status$ 指日志项中“线程”持有共享页面的访问权限类型(R 为读访问, W 为写访问)。基于算法 1 所示的拓扑图建立算法,将具有全局逻辑序的日志文件按照依据“线程”划分,对于每一个共享页面访问权限的转移建立转移区间,并同时识别出其中的访存依赖对,如图 3 (b)所示,依赖对由图中的箭头标识。



(a)原有日志文件 (b) 基于转移区间的拓扑图

图 3 建立拓扑图示意图

5 重构程序逻辑

原先具有全局逻辑序的日志文件按照“线程”划分,在重放阶段每个“线程”读取各自的日志文件。为了在重放执行阶段,继续维持各个“线程”间的访存依赖对间的逻辑序关系,采用条件变量集合(Cond Set)来重构拓扑图中的程序逻辑——访存依赖对的后继必须在依赖对的前驱执行之后才能执行。其原

共享内存访问顺序,如表 1 所示. 其中,虽然 x264 程序中访存依赖对的数量占日志项总数的比例很小,但是与其他测试程序不同,x264 测试程序频繁创建、回收线程,频繁创建、回收 256 个线程,与记录阶段一样,在重放阶段我们也需要相应的创建、回收 256 个进程,其性能开销主要来源于频繁的创建、回收“线程”. 即使采用进程复用优化策略^[8],UPLAY 仍需要添加合适的同步操作保证记录时被复用的“线程”在重放阶段也同样按序重放,不能并行执行. 因此与串行重放相比,x264 测试程序在并行重放方法下重放执行速度并没有得到很大的提高.

表 1 有效日志项的统计

	c_1	c_2	$c_2/c_1/\%$
ferret	2 796 141	14 469	0.52
facesim	3 362 906	254 479	7.57
raytrace	5 209 218	615 618	11.82
bodytrack	296 104	10 535	3.56
vips	912 205	51 793	5.68
x264	739 048	3 023	0.41
blackscholes	1 158	13	1.12
swaptions	451 462	78	0.02
caneal	104 592 850	2 539 559	2.43
dedup	4 551 041	287 056	6.31
streamcluster	4 662 325	306 523	6.57

c_1 :原日志文件中日志项总数 c_2 :具有访存依赖关系的日志项数量

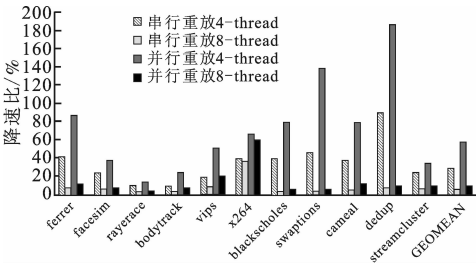


图 5 并行重放与串行重放性能统计

我们统计了 PARSEC 测试集在记录阶段、并行重放阶段的性能,如图 6 所示. 并行重放方法的主要目的是识别原有日志文件中的访存依赖对的日志项,并将具有全局逻辑序的单个日志文件拆分为“线程”独立日志文件,从而使得在并行程序重放执行阶段,原有日志文件中可并行的日志项并行执行,具有依赖关系的日志项串行执行. 实际上,与并程序的记录执行相比,两个阶段下每个“线程”触发页保护异常的次数并没有变化. 并行重放执行虽然从以下几个方面做了优化:(1)减少了记录日志的操作;(2)减少了共享页面访问控制中的同步操作,例如在修改全局页面权限表中的特定项时需要抢占私有锁;(3)减少原有并行程序中的互斥锁操作. 但是重放阶段通过添加等待-唤醒操作来维持具有依赖关系的

日志项,而这种操作需要在用户态和内核态切换. 因此,与记录执行相比,并程序的并行重放执行性能略微有些降低,尤其是那些共享页面竞争激烈的测试程序,如 raytrace、caneal. 这两个测试程序的典型特征是子线程会频繁抢占主线程分配的堆区,因此并行重放执行阶段的性能明显低于记录执行阶段的性能. 另外,由于 blackscholes、swaptions 程序中未采用显式同步操作,且没有明显的共享页面竞争,因此记录执行与重放执行的性能相当.

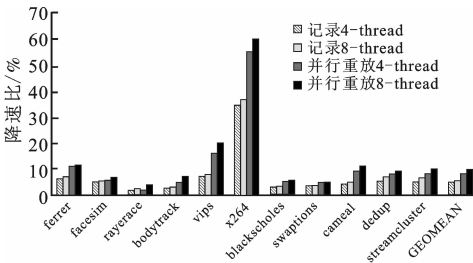


图 6 记录阶段与并行重放性能统计

7 结束语

本文提出一种基于访存依赖对的并行重放方法来实现快速重放. 该方法的核心思想是静态分析日志文件中的所有日志项,并根据每个“线程”获取共享页面访问权限的日志项为每个共享页面建立访问权限转移区间,同时识别其中的访存依赖对,并添加合适的条件变量集合来维持这些日志项之间的依赖关系,从而保证重放执行阶段各个“线程”仅按序执行这些日志项. 实验结果表明,本文提出的基于访存依赖对的并行重放方法能够有效去除原有日志文件中冗余的序关系,从而提高重放执行的效率.

参考文献:

[1] Pokam G, Pereira C, Danne K, et al. Hardware and software approaches for deterministic multi-processor replay of concurrent programs[J]. Intel Technol, 2009,13(4):20-41.

[2] Leblanc T J, Mellor-Crummey J M. Debugging parallel programs with instant replay[J]. IEEE Trans. Comput, 1987,34(4):471-482.

[3] Liu P, Zhang X, Tripp O, et al. Light: Replay via tightly bounded recording [C]//PLDI 2015. USA, Portland,2015: 55-64.

[4] Honarmand N, Torrellas J. Replay debugging: Leveraging record and replay for program debugging[C]// ISCA 2014. USA, Minnesota, 2014:445-456.

[5] 唐士斌. 多线程程序的确定性调试方法[D]. 北京:中

国科学院计算技术研究所, 2014.

[6] Patil H, Pereira C, Stallcup M, et al. PinPlay: a framework for deterministic replay and reproducible analysis of parallel programs[C]//CGO 2010. Canada, Toronto, 2010:2-11.

[7] Bhansali S, Chen W K, de Jong S, et al. Framework for instruction-level tracing and analysis of program executions[C]//VEE 2006. Canada, Ottawa, 2006: 154-163.

[8] 应欢, 王东辉, 武成岗, 等. 适用于商用系统环境的低开销确定性重放技术[J]. 吉林大学学报(工学版), 2017, 49(1):1782-1790.

[9] Courtois P J, Heymans F, Parnas D L. Concurrent Control with ‘eaders’ and ‘writers’ [J]. Commun. ACM, 1971, 14(10):667-668.

[10] Bienia C, Kumar S, Singh J P, et al. The PARSEC benchmark suite: characterization and architectural implications [C]//PACT 2008. Canada, Toronto, 2008:72-81.

作者简介:

应欢 女, (1988-), 博士研究生. 研究方向为并程序调试. E-mail:yinghuan1022@126.com.

王东辉 男, (1973-), 博士生导师. 研究方向为 VLSI 信号处理.

王喆 男, (1990-), 博士研究生. 研究方向为动态编译与软件安全.

(上接第 5 页)

[4] 辛明瑞, 时晨, 高德远, 等. 面向空间应用的片上系统集成技术研究[J]. 微电子学与计算机, 2006, 23(6): 194-197.

[5] GRLIB IP Core User’s Manual (Version 1. 2. 0 - B4121)[M]. New York: Aeroflex Gaisler Inc. 2012.

[6] Niu G, Krithivasan R, Cressler J D, et al. A comparison of SEU tolerance in high-speed SiGe HBT digital logic designed with multiple circuit architectures[J]. Nuclear Science, IEEE Transactions on, 2002, 49(6): 3107-3114.

[7] 武振平, 付方发, 肖立伊. 基于 VHDL 故障注入的处理器敏感性分析[J]. 微电子学与计算机, 2012, 29(10): 51-54.

[8] 黄海林, 唐志敏, 许彤. 龙芯 1 号处理器的故障注入方法与软错误敏感性分析[J]. 计算机研究与发展, 2006, 43(10): 1820-1827.

[9] 马兴瑞, 张永维, 白照广. 实践五号卫星及其飞行成果[J]. 中国航天, 1999, 21(11):3-8.

作者简介:

赵昌兵 女, (1992-), 硕士. 研究方向为先进集成系统与 SoC. E-mail:hitzcb@163.com.

付方发 男, (1984-), 博士, 讲师. 研究方向为片上网络体系结构研究.

肖立伊 女, (1961-), 博士, 教授, 博士生导师. 研究方向高可靠集成电路设计、SoC/IP 核设计与标准化.