文章编号：0253-374X(2013)12-1889-09

DOI:10.3969/j.issn.0253-374x.2013.12.021

一种同时多线程指令队列竞争缓解策略

江建慧，刘　宇，朱一南，钱剑琤

（同济大学 软件学院，上海 201804）

摘要：同时多线程结构利用线程级并行和指令级并行的优势，提高了指令吞吐率，但线程对关键资源（如指令队列）的竞争会削弱这种优势，造成资源浪费，又会降低处理器性能. 提出了指令队列利用参数，通过分析指令队列利用率与处理器性能的关系，用实验评估了在四线程情况下，典型静态指令队列竞争缓解策略（如 Dwarn，2OP_Block，Static）及其组合对处理器性能的影响. 给出了 load 依赖链模型，分析了基于 load 依赖链的基准程序线程特性，提出了一种结合线程特性的指令队列竞争缓解策略. 实验结果表明，该策略能够加速执行指令吞吐率较高的线程，通过提升此类线程的性能使整体指令吞吐率进一步增加.

关键词：同时多线程；指令队列；load 依赖链；竞争缓解策略；线程特性

中图分类号：TP368.5　　**文献标志码**：A

A Kind of Instruction Queue Competition Easing Strategy for Simultaneous Multi-threading Architecture

JIANG Jianhui, *LIU Yu*, *ZHU Yinan*, *QIAN Jiancheng*

(College of Software Engineering, Tongji University, Shanghai 201804, China)

Abstract: The simultaneous multi-threading (SMT) technique boosts instructions per clock (IPC) by adopting thread level parallelism and instruction level parallelism. However, the competition of key resources between threads do weaken such advancement. Instruction queue (IQ) is proved as one key resource and its competition always results into performance degradation. Typical IQ competition easing strategies include Dwarn, 2OP_Block and Static. This paper presets two IQ utilization parameters to estimate the relationships between IQ usage and system performance. Competition easing capability of typical IQ strategies and their combination are compared. A load dependency chain model is built and analysis of thread characteristics based on the model is given. Then a new IQ competition easing strategy combining with thread characteristics is proposed. The experimental results show that such strategy can achieve total IPC improvement by accelerating high IPC threads.

Key words: simultaneous multi-threading; instruction queue; load dependency chain; competition easing strategy; thread characteristics

同时多线程（simultaneous multithreading，SMT）结构结合了超标量和多线程结构的特点，允许在一个时钟周期内发射多个独立线程的多条指令，同时减少水平浪费和垂直浪费. 水平浪费的产生是由于某个时钟周期里面，一个线程不能够发射足以填满整个发射槽的指令，而垂直浪费的产生是由于某个时钟周期里面一个线程发生了阻塞，造成接下来的几个周期里面发射槽完全不能够被利用. SMT 结构极大地开发了指令级并行（instruction level parallelism，ILP）和线程级并行（thread level parallelism，TLP).

然而，某些关键资源的不合理利用会导致处理器性能下降[1-2]. 本身 ILP 较低的线程如果经常遇到长延迟指令，产生二级 Cache 缺失，会长期占据甚至阻塞关键资源. 文献[3]指出 SPEC 2000 中的 mcf，art 等访存密集型线程因 Dcache 缺失造成的性能损失比重超过或接近 40%.

指令队列（instruction queue，IQ）是 SMT 结构中的关键资源，在多线程共享的同时也被多个线程竞争[4]. 缓解其竞争的一个最直接方法是提供足够大的 IQ，但是增大 IQ 既会导致时钟周期上升，能耗增大. 为此，又提出了一些其他策略，如不改变 IQ 结构，通过设置计数器或选择逻辑来控制 IQ 分配的策

收稿日期：2012-07-23
基金项目：国家自然科学基金(60903033)
第一作者：江建慧(1964—)，男，教授，博士生导师，工学博士，主要研究方向为可信系统与网络、软件可靠性工程、VLSI/Soc 测试与容错. E-mail:jhjiang@tongji.edu.cn
通讯作者：刘　宇(1988—)，男，工学硕士，主要研究方向为处理器微体系结构. E-mail:ruguo.ai2007@163.com

略[2,5-6];设置辅助队列或多级队列,将IQ中的长延迟指令与其他指令分离来疏通IQ的策略[7-9].由于IQ本身的唤醒逻辑和选择逻辑较为复杂,辅助队列在面积、功耗、控制复杂性方面都会有所增加,而权衡性能提升和开销增加是一个较复杂的问题,因此本文将研究的是无需改变IQ结构的缓解竞争的策略.

根据策略所应用的流水线段可将其分为3类[10]:取指段降低选中的线程优先级,调度段推迟选中的指令,发射段采用IQ资源划分.对应的典型策略(也是较早提出的策略)分别是Dwarn[5],2OP_BLOCK[6],Static[2],其余策略多为其改进[11-13].Dwarn试图解决SMT中某个线程发生Dcache缺失,尤其是L2 Cache缺失,长时间占用IQ导致其他线程饥饿的问题.检测到Dcache缺失,立即降低该线程在取指段的优先级.当只有两线程时,发生L2缺失则停止取指.2OP_BLOCK认为过多双操作数都未准备好的指令占用指令队列,会降低其利用效率,该策略从ROB(reorder buffer)头部发射至少一个操作数准备好的指令到IQ,如果遇到双操作数都未准备好的指令,则终止该线程的本次调度.研究发现存储类资源采取静态划分(相应资源分为私有和共享两部分,私有资源按线程数量分为N份,每个线程保有自己的一部分,共享资源先来先得)的方式可以取得较好效果,对IQ的划分也是如此,Static就是基于这种思路.然而,这些研究多为直接评估系统性能,并未讨论IQ利用参数与处理器性能提升的关系.

IQ竞争的一个重要原因是由于某些线程因load指令发生多级Dcache缺失,长期占据IQ,使得其他流动性较好的线程无法获得必要的资源,进而造成流水线阻塞.然而,不同线程对IQ的需求和竞争能力是不同的.某一线程进入IQ的load指令及其后续依赖指令的平均间隔较短,一旦发生load缺失尤其是L2 Dcache缺失,检测到缺失之前大量后续指令可能已经进入,由于一个L2 Dcache会产生几百个时钟周期的延迟,多个二级缺失叠加,很容易阻塞IQ.相反,如果线程的平均间隔较长,load指令即不易发生缺失,在检测到发生缺失之后采取相应措施可能更好.

本文以四线程为例,量化研究了不同策略及策略组合缓解IQ竞争的能力.给出了load依赖链模型,分析了基于load依赖链的基准程序线程特性,并提出了一种结合线程特性的IQ竞争缓解策略.

1 典型IQ竞争缓解策略实验分析

1.1 IQ利用参数

从空间和时间两个方面来考虑IQ的利用情况,分别定义了平均IQ占用数量(average numbers of IQ occupied, a_IQ_oc),即每个周期平均有多少个IQ被使用,以及平均IQ占用周期(average cycles for a IQ slip, a_IQ_slip),即每个IQ从分配到释放平均经历了多少个时钟周期.

以四线程为负载,将IQ使用程度(表示为degree)离散化为1~5个等级,当IQ使用数量介于0~IQ_SIZE/THREAD_NUM * 1-1时,degree=1;而介于IQ_SIZE/THREAD_NUM * 1~IQ_SIZE/THREAD_NUM * 2-1时,degree=2.其他情形与此类似,IQ满载时,degree=5.因此,有

$$a\_IQ\_oc = i * \sum_{1}^{5} p_i, i=1,2,3,4,5,$$

表示degree为i的周期占总体的比例.

$$a\_IQ\_slip = \sum_{1}^{n} T_j/n, j=1\sim n,$$

n表示执行的指令总数,T_j表示第j条指令从分配IQ到释放经历的周期数.

1.2 IQ竞争缓解策略与处理器性能优化

1.2.1 单一策略及其组合

IQ竞争缓解策略研究的关键,在于研究IQ利用参数a_IQ_oc,a_IQ_slip与系统性能IPC的关系,建立IQ性能优化与系统性能优化关系的模型.

不同策略的组合是一种可能的缓解策略.在四线程情况下,Dwarn,2OP_BLOCK和Static等3种IQ竞争缓解策略及其组合如表1所示.

表1 IQ竞争缓解策略及其组合

Tab.1 IQ competition easing strategies and their combination

策略组合	无	Dwarn	2OP_block	static
1(000)	1	0	0	0
2(100)	0	1	0	0
3(010)	0	0	1	0
4(001)	0	0	0	1
5(110)	0	1	1	0
6(101)	0	1	0	1
7(011)	0	0	1	1
8(111)	0	1	1	1

注:"1"表示选取该种策略,"0"表示不选取该种策略.

选取SPEC 2000程序作为基准负载,它们是预编译的Alpha二进制代码.为了降低负载初始化对结果的影响,跳过每个负载前面的初始化代码[14].当

某个线程提交指令数目达到 3 亿条时，则结束本次模拟. 为了提高充分性，选取 ILP 类和 MEM 类中的程序列入 MIX 类，将未选取的程序列入 OTHERS 类. 实验负载将尽量考虑各种负载组合，见表 2.

表 2　SPEC 2000 中选取的实验负载

Tab.2　Workloads selected from SPEC 2000

程序负载组合				
ILP1	apsi/bzip2/gcc/vortex	4ILP0MEM	OTHERS1	equake/art/fma3d/facerec
ILP2	apsi/gcc/gzip/eon		OTHERS2	galgel/gap/lucas/mesa
MIX1	apsi/applu/swim/vpr	1ILP3MEM	OTHERS3	mgrid/perlbmk/wupwise/bzip2
MIX2	gcc/applu/mcf/ammp		OTHERS4	parser/vpr/sixtrack/bzip2
MIX3	bzip2/gcc/swim/vpr	2ILP2MEM		
MIX4	gzip/eon/mcf/ammp			
MIX5	apsi/bzip2/gcc/applu	3ILP1MEM		
MIX6	apsi/gcc/gzip/vpr			
MEM1	applu/swim/vpr/twolf	0ILP4MEM		
MEM2	applu/vpr/mcf/ammp			

模拟器采用 Binghamton 大学的 M-SIM2. 0[12]，其配置参数如表 3 所示. 初始 IQ 的大小为 32.

表 3　M-SIM 的基本配置

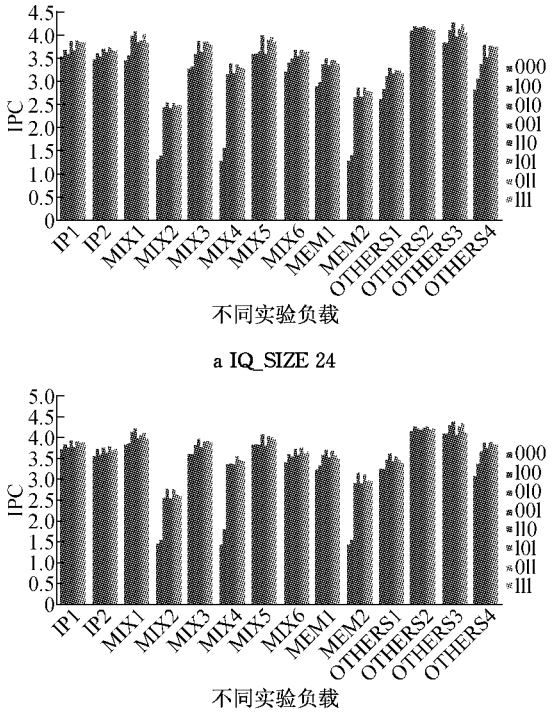
Tab.3　Configuration of simulator M-SIM

参数名	参数值
machine width	8 fetch_width;8 decode_width;8 issue_width;8 commit_width
rf_size	256
ROB_size	96 * 4
LSQ_size	48 * 4
IQ_size	24/40
tlb_miss_lat	30
OTHERS	default

1.2.2　实验结果

图 1 给出了实验的结果，同一种实验负载里的相邻不同条纹依次表示策略 1 至策略8. 图 1a和1b分别配置 IQ_SIZE 为 24 和 40.

在 IQ 容量较大和较小的情况下，采用 IQ 竞争缓解策略对于系统性能都有明显提升. 其中，Static 与 2OP_BLOCK 显著提升 IPC，尤其是负载 MIX2，MIX4，MEM1，MEM2，OTHERS4 的 IPC 增长接近 50%. 对 ILP1 和 ILP2 的提升较小，此类计算密集型线程较少发生 Dcache 缺失，因此导致 load 指令长时间占用 IQ 资源，后续依赖指令阻塞 IQ 的情况较少. MEM1 和 MEM2 负载下的提升明显，说明该种线程组合下使系统性能受到 IQ 阻塞的严重制约，这是由访存密集型线程本身的特性，即频繁访问 Dcache 并多级缺失造成的. MIX2 和 MIX4 中混有如 mcf 等访存密集的线程，因此也在设置 IQ 竞争缓解策略之后获得了较大提升. 进一步，Dwarn，2OP_BLOCK，Static 这 3 种策略的组合提升系统性能的效果不佳. 如在某些线程组合中，策略 5 比策略 3 只获得了微弱的性能提升，策略 7 和策略 8 都没有超过只选用 Static 的情况.



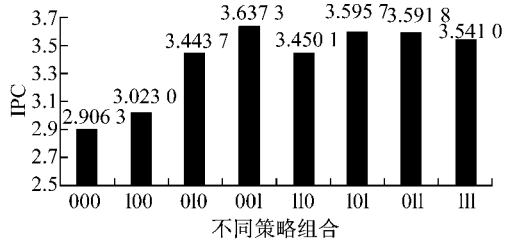
a IQ_SIZE 24

b IQ_SIZE 40

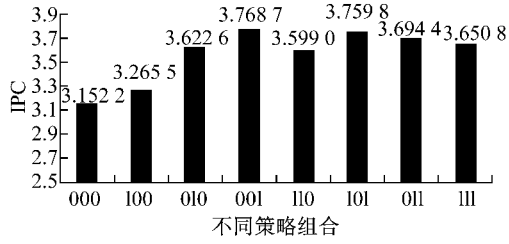
图 1　不同 IQ 竞争缓解策略的 IPC

Fig.1　IPC for IQ competition easing strategies by different workloads

图 2 给出了平均 IPC 值. 策略 2 带来的提升不大，Dwarn 在取指段降低某些线程的优先级，如果线程在流水线中的流动速度相近，那么频繁地切换优先级不是很好的选择，甚至造成性能下降，如图 1 中 MIX5 所示. 策略 3 作用明显，2OP_BLOCK 禁止双操作数未准备好的指令获得 IQ 资源，减少了因 load 指令缺失而大量后续依赖指令阻塞队列的情况. 策略 4 效果最好，Static 能够保证各个线程不因 IQ 完全被其他线程占用而停止. 策略组合带来的 IPC 提升没有超过单个策略的效果.



a IQ_SIZE 24



b IQ_SIZE 40

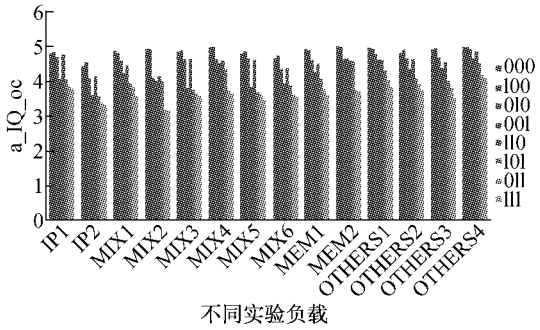
图2 不同IQ竞争缓解策略的平均IPC

Fig.2 Average IPC for IQ competition easing strategies

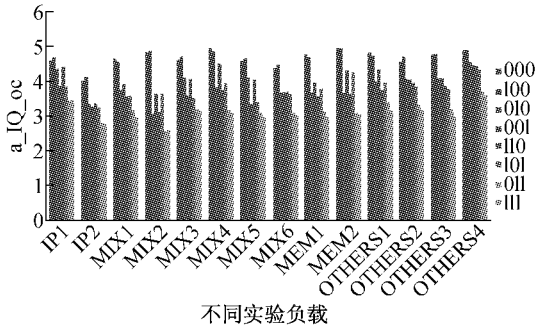
1.3 实验结果分析

1.3.1 a_IQ_oc分析

设a_IQ_oc的上限为5,越接近上限,则IQ满载的比例越高,如图3所示.在不采用IQ竞争缓解策略的情况下,运行各个实验负载的IQ都接近满载,设置不同的IQ资源也无法避免线程间的竞争所导致的阻塞.



a IQ_SIZE 24



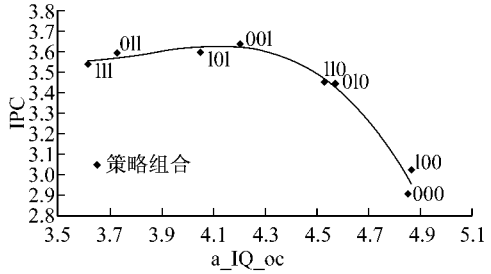
b IQ_SIZE 40

图3 不同IQ竞争缓解策略的a_IQ_oc数据

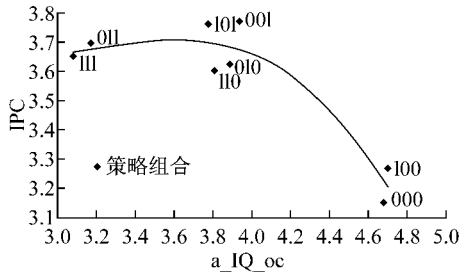
Fig.3 a_IQ_oc for IQ competition easing strategies by different workloads

Dwarn对于a_IQ_oc影响不大,说明线程调度的方法并不能有效缓解其竞争,该策略对于系统性能的提升或降低更多依赖于线程本身特性. 2OP_BLOCK和Static都较好地降低了IQ的空间占用,有效缓解其竞争,因此,后者对IQ空间上的缓解能力更好,但采用较大IQ资源时,前者对某些组合缓解能力更优.在ILP类线程组合负载下,只有Static及其混合策略能够较好地缓解IQ资源空间上的竞争.策略5是策略2和策略3的混合,其对a_IQ_oc的影响是这两种策略单独作用的综合,策略6与此类似.策略7与策略8包含2OP_BLOCK与Static的共同作用,在流水线调度阶段采用指令调度与资源划分的方法能够极大地缓解IQ竞争,这两种策略作用下的a_IQ_oc得到进一步的降低.

图4列出了a_IQ_oc与IPC的比较,随着缓解策略数量的增多,IQ的竞争逐步得到缓解,但IPC没有持续提升,后者呈先升后降变化,这与现实是相符合的,因为一味禁止线程获得充足的IQ资源有可能导致性能降低.此外,2OP_BLOCK,Static及策略6情况下得到的a_IQ_oc是较好的,这也解释了策略7和8并不能获得更好的IPC的原因.



a IQ_SIZE 24



b IQ_SIZE 40

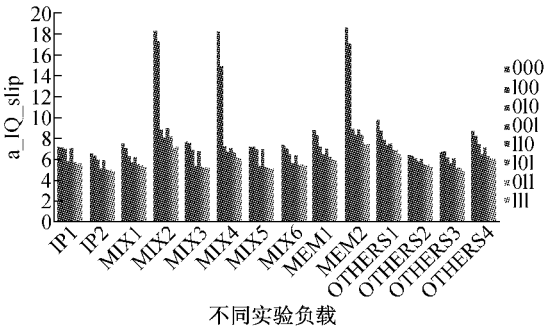
图4 不同IQ竞争缓解策略的平均a_IQ_oc

Fig.4 Average a_IQ_oc for IQ competition easing strategies

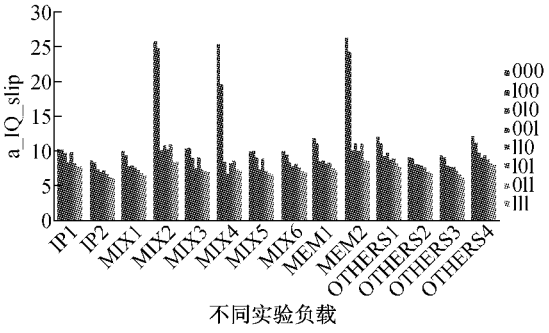
1.3.2 a_IQ_slip分析

图5表明,IQ竞争缓解策略明显降低了该资源的平均占用周期. Dwarn对于a_IQ_oc影响不大,但或多或少降低了a_IQ_slip,表明该策略通过改变线

程取指优先级的方法，加速了 IQ 资源在线程间的切换. 2OP_BLOCK 延缓双操作数为准备好的指令进入 IQ，Static 变相地延后分配 IQ 资源，两者都较好地降低了 IQ 时间上的占用，综合来看，后者对 IQ 时间上的占用缓解能力更好，但采用较大 IQ 资源时，前者对某些组合缓解能力更优. 在 ILP 类线程组合负载下，只有 Static 及其混合策略能够较好地缓解 IQ 资源时间上的竞争. 策略 5 是策略 2 和策略 3 的混合，其对 a_IQ_slip 的影响是这两种策略单独作用的综合，策略 6 与此类似. 策略 7 与策略 8 包含 2OP_BLOCK 与 Static 的共同作用，与 a_IQ_oc 类似，这两种策略作用下的 a_IQ_slip 得到进一步的降低.



a IQ_SIZE 24



b IQ_SIZE 40

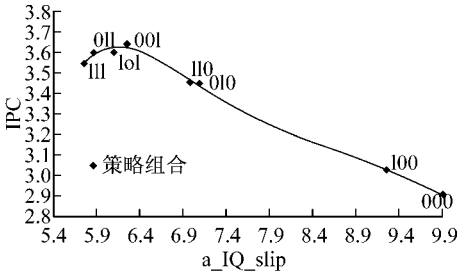
图 5　不同 IQ 竞争缓解策略的 a_IQ_slip 数据

Fig.5　a_IQ_slip for IQ competition easing strategies by different workloads

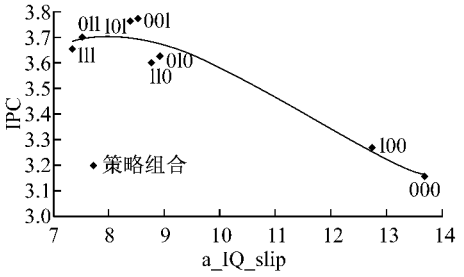
图 6 列出了 a_IQ_slip 与 IPC 的比较，随着缓解策略数量的增加，IQ 的竞争逐步得到缓解，但相应的 IPC 没有持续提升，后者仍呈先升后降变化. 2OP_BLOCK，Static 及策略 6 情况下得到的 a_IQ_slip 是较好的，策略 7 和 8 并不能获得更好的 IPC.

研究结果表明：

(1)Static，2OP_BLOCK 对 IQ 竞争缓解作用明显，其他策略次之. Static 将 IQ 资源按线程数完全划分，这种不灵活的方式指出一种可能，即对共享资源的使用，保障线程得到最小资源的能力比使其具有获得最大资源的能力更重要.



a IQ_SIZE 24



b IQ_SIZE 40

图 6　不同 IQ 竞争缓解策略的平均 a_IQ_slip

Fig.6　Average a_IQ_slip for IQ Competition Easing Strategies

(2)不同流水线段的典型 IQ 缓解策略的组合进一步降低了 IQ 的竞争，但是过多地禁止线程获得 IQ 资源无法提升系统性能，甚至造成下降，如 2OP_BLOCK 和 Static 共同作用.

(3)对比 a_IQ_oc，a_IQ_slip 与 IPC 的关系发现，为不同线程组合配置单一策略的方式，受到线程组合本身的限制，而不同线程对 IQ 的需求和竞争能力是不同的，是否可以根据线程特性配置缓解策略，需要进一步的研究.

2　结合线程特性的指令队列竞争缓解策略

2.1　基于 load 依赖链的线程特性

2.1.1　load 依赖链

load 依赖链指的是在 1 个线程中，以 load 指令开始，后续指令依赖前序指令结果作为源操作数的有限指令序列. ①每条依赖链以 1 条 load 指令开始，以该链最后 1 条指令提交结束；②若前序指令在 1 条 load 链中，后续依赖指令也加入该依赖链；③指令若依赖两条不同的 load 链，取最近的 1 条加入.

符合 load 链定义的两个模型如图 7 所示，字母 a～k 表示指令的调度顺序. 模型 1 中的 a 是 1 条 load 指令，表示 1 条 load 链的开始，假设该链在指令 k 处结束，则该 load 链包含 4 条指令，3 个有效间隔；模型 2 中的 a 和 i 各为 1 条 load 指令，以 a 开始的

load链假设在指令g处结束,则包含4条指令,3个有效间隔,以i开始的load链假设在指令k处结束,则包含2条指令,1个有效间隔.

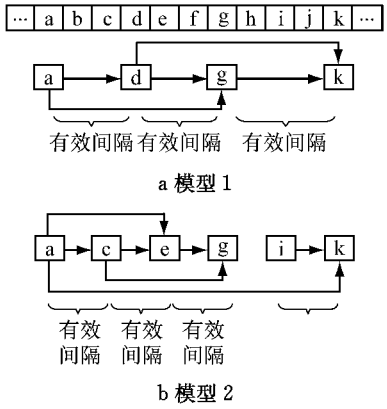


图7 load依赖链模型

Fig.7 load dependency chain models

2.1.2 线程特性指标

基于load依赖链的线程特性指标体系包含如下指标:

(1)load依赖链数量(表示为chains):load指令的数量.值越大,则线程特性越强.不同线程chains的值基本在同一级别.为了观察方便,同除以10^7.

(2)依赖链中的平均指令间隔长度(表示为a_len_load_interval).较小的间隔容易造成IQ阻塞,计算方式为

$$\text{a\_len\_load\_interval}=\left(\sum_{i=0}^{n}\sum_{j=0}^{a_i}b_j\right)\Big/\left(\sum_{i=0}^{n}a_i\right)$$

式中:a_i是第i条load依赖链的间隔数量(不小于0的整数);b_j是某条load依赖链中第j个间隔的长度(周期数);n是所有的load依赖链条数.

(3)依赖链中的平均指令间隔个数(表示为a_num_load_interval).较长的依赖链容易造成IQ阻塞,计算方式为

$$\text{a\_num\_load\_interval}=\sum_{i=1}^{n}a_i/n$$

式中:a_i是第i条load依赖链的间隔数量(不小于0的整数);n是所有的load依赖链条数.

(4)load指令发生L2 Dcache缺失的程度(表示为CacheMissDegree).发生缺失数量的波动较大,从十万条级别至千万条级别,该值主要用于区别缺失程度,并不考察精确值,据此,设置缺失数量十万级、百万级和千万级的线程,该值分别为1,2或4,10.

总的线程特性计算方法如下:Degree = CacheMissDegree * chains * a_num_load_interval/a_len_load_interval,Degree越大,则线程竞争IQ的能力越强.

2.1.3 基准程序线程特性

选取SPEC 2000中的25个程序作为基准负载(crafty在M-SIM2.0上不能正确运行,故略去),得到的原始线程特性指标如表4所示.

表4 SPEC 2000线程特性指标计算结果

Tab.4 Thread characteristic index results for SPEC2000

程序名	指标				
	inst_commit	DL2miss_num	load_chains	a_num_l_inter	a_len_l_inter
applu	300 000 000	5 682 857	81 040 935	2.059 5	16.508 4
swim	300 000 000	5 781 728	64 671 653	1.923 5	22.793 1
mcf	300 000 000	28 591 525	178 016 648	1.019 6	4.628 4
gcc	300 000 000	5 989 147	169 558 055	0.496 5	15.245 4
⋮	⋮	⋮	⋮	⋮	⋮
fma3d	300 000 000	7 707 630	80 606 635	1.914 7	6.941 2
lucas	300 000 000	5 030 999	34 992 045	1.142 9	3.018 6

按Degree可以得到程序排名,rank值从高到低,代表IQ竞争能力从强到弱.最高的mcf为39.21,最低的vortex为1.07.策略设计和实验验证应尽量考虑各种强弱线程的情况.

2.2 结合线程特性的缓解策略

2.2.1 基本原理

本文提出一种根据线程特性配置静态IQ竞争缓解策略的方法,为竞争能力强的线程配置缓解能力强的策略,为竞争能力弱的线程配置缓解能力弱的策略或不配置策略.

考察已有策略发现,Static直接作用于IQ分配,是一种缓解能力和缓解效果都比较好的策略,且硬件开销很小(用计数器实现),因此选其为缓解能力强的策略;OOOD作用于IQ调度阶段,降低了2OP_BLOCK对于TLP的限制,因此选其为缓解能力弱的策略.Dwarn作用于流水线第一级,是一种间接的缓解IQ竞争策略,选取其用于策略的组合.

根据单个线程上作用策略的数量,给出两种具体的配置方法:一是单混和策略,为竞争能力强的线程配置Static,弱的线程不配置策略或配置较弱的OOOD;二是多混合策略,在单混合策略的基础上面,为每个线程加入Dwarn.

2.2.2　配置方法

对于四线程情况，如果它们同为竞争激烈或竞争不激烈，那么采用统一的配置方法，如 Static. 如果同时有竞争激烈（较强）和不激烈（较弱）的线程，则根据线程组合情况（一弱三强、两弱两强、三弱一强）进行配置. 对于一弱三强组合，可考虑不限制弱的线程，期望弱线程的快速流动能够加快资源的切换. 对于两弱两强组合，除了为强线程配置 Static，也为弱的线程配置较弱的 OOOD，以防止弱线程之间的过度竞争. 对于三弱一强组合，则为强线程配置 Static，弱线程配置 OOOD. 表 5 给出了一个策略提升 IPC 的例子，其中，S 代表 Static，O 代表 OOOD，D 代表 Dwarn.

表 5　策略配置实例

Tab.5　Strategy configuration examples

线程	Rank	单策略	单混和策略	多混合策略
bzip2	18	S	O	D+O
apsi	23	S	O	D+O
mcf	1	S	S	D+S
lucas	9	S	S	D+S

2.2.3　可行性分析

结合线程特性采用相应策略，能够更好地利用 IQ 资源，加速原本 IQ 竞争能力较弱的线程的流动速度，而原本 IQ 竞争较强线程的受限情况变化较小. 实验发现，在某些负载组合下采用本文策略，所有负载的吞吐率都上升了.

新策略可复用现有机制，只需附加简单的控制逻辑，因此硬件开销较小. 该策略提前获取线程特性，因此适用于某些专用系统，如长期运行固定负载的服务器.

3　实验及其结果分析

3.1　基准程序选取

在四线程情况下，实验负载将尽量考虑强弱不同线程特性的负载组合，见表 6. 所有强线程配置 Static. 而对于弱线程，在编号 1～4 负载中，不配置；在其余编号负载中，配置 OOOD. 不考虑四线程的特性同为强或弱的情况.

3.2　实验结果分析

本文研究的目的是优化处理器性能，因此主要考虑的指标是处理器 IPC 和线程 IPC.

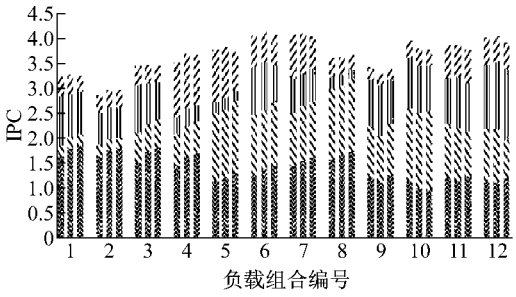
3.2.1　处理器 IPC

图 8 给出了不同缓解策略下的处理器 IPC. 图 8a 和 8b 中 IQ 大小分别取 24 和 40. 图中从左至右给出 1 号至 12 号负载结果，每种负载结果依次为单策略（Static）、单混和策略，以及多混合策略，4 种花纹分别对应 4 个线程的 IPC.

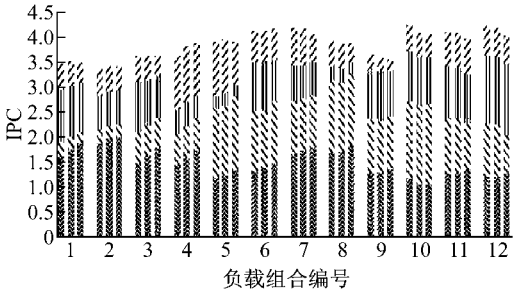
表 6　实验负载

Tab.6　Experimental workloads

编号	T1	T2	T3	T4	组合
1	vortex	mcf	fma3d	art	1 弱 3 强
2	mesa	mcf	ammp	art	
3	vortex	parser	fma3d	art	
4	mesa	paerser	art	lucas	
5	eon	apsi	mcf	lucas	2 弱 2 强
6	mesa	apsi	lucas	parser	
7	vortex	eon	equake	mgrid	
8	vortex	mesa	mcf	art	
9	eon	perlbmk	facerec	mcf	3 弱 1 强
10	swim	bzip2	galgel	art	
11	eon	swim	facerec	parser	
12	eon	swim	mesa	ammp	



a IQ_SIZE 24



b IQ_SIZE 40

图 8　单策略、单混合策略、多混合策略的 IPC 结果

Fig.8　IPC of single, single mixture and multi-mixture strategies

在图 8a 中，单混合策略大幅提升了较弱线程的 IPC，进而整体吞吐率得到了提升，其中 2 号和 4 号负载提升明显；多混合策略是在单混合策略基础上，取指段加入 Dwarn，因此这种线程的吞吐率是 3 个策略中最高的. 同时，给予弱线程过多的资源占用机会并不能带来整体性能的线性提升，3 号和 4 号负载的多混合策略整体性能反而不及单混合策略. 5 号至 8 号负载是 2 个竞争较强线程和 2 个竞争较弱线程的组合，单混和策略相较单策略有更好的整体吞吐率，但提升幅度没有 2 号和 4 号负载明显，因为弱线

程之间的竞争会削弱这种优势;而多混合策略在此类负载情况下,存在进一步提升的可能,如8号负载,但也有其整体IPC不及单策略的情况,如5,6,7号负载.9号至12号负载是1个竞争较强线程和3个竞争较弱线程的组合,随着弱线程间的竞争进一步增大,采用线程特性配置策略的方法受到了一定的限制,在大部分情况下,单混合策略和多混合策略都没能超过单策略的整体IPC,因此在这种情况下,采用单一策略更好.

当IQ增大时,线程间竞争对IQ资源使用的影响会变小,表现为采用新策略所带来的性能提升幅度减小,如图8b所示.对1号至4号负载,3号负载的单混和策略效果相较IQ值较小时降低,4号负载提升仍然明显.5,6号负载略有提升,而7,8号负载性能降低.对9号至12号负载,大部分情况下,单混合策略和多混合策略都没能超过单策略的整体IPC.

总体来看,当IQ容量较小时,线程对该资源竞争激烈,采用新策略的效果更好.

3.2.2 线程IPC

表7比较了IQ 24下12组负载的IPC,包括单混和策略与单策略的对比,多混合策略与单策略的对比,正值代表提升,负值代表降低.对1号至4号负载,较弱线程得到了大幅提升,较强线程受到一定抑制,但由于较强线程本身占整体的比例较小,其降低对于整体影响不大.4号负载的单混策略取得了较为理想的结果,减轻较弱线程的IQ分配限制,带动了流水线整体的流动,各个线程性能都增加了,整体IPC提升超过5%.对5号至8号负载,单混策略都是正面的效果,但不及前面4个负载,而多混策略降低了性能,表现为较弱线程的提升无法消除较强线程降低的影响,因为较弱线程之间也存在竞争.对9号至12号负载,随着弱线程间的竞争进一步增大,采用线程特性配置策略的方法受到了一定的限制,因此该类负载可以采用单策略.

表7 IQ 24负载的IPC比较

Tab.7 IPC comparison under IQ size of 24 %

指标	1号		2号		3号		4号	
	单混	多混	单混	多混	单混	多混	单混	多混
总IPC	1.03	0.91	3.63	3.73	0.84	0.23	5.42	4.62
IPC1	10.42	14.65	8.48	9.18	14.30	20.83	11.87	16.28
IPC2	−2.39	−9.71	−1.81	−1.76	−5.81	−6.93	0.76	2.71
IPC3	−12.66	−14.69	−3.69	−3.63	−14.78	−21.29	2.24	−4.84
IPC4	−1.07	−10.44	−1.62	−3.97	−1.43	−14.61	0.48	−6.73
指标	5号		6号		7号		8号	
	单混	多混	单混	多混	单混	多混	单混	多混
总IPC	1.29	−0.98	1.39	0.31	0.85	−0.47	0.90	2.16
IPC1	6.01	15.36	7.82	14.96	2.48	2.89	4.25	9.96
IPC2	0.17	6.31	−1.49	4.72	8.53	13.46	−1.63	1.40
IPC3	0.34	−11.66	0.63	−21.22	−5.93	−4.86	−1.64	−10.11
IPC4	−2.07	−25.62	−4.55	−4.31	−8.11	−23.45	−1.80	−20.02
指标	9号		10号		11号		12号	
	单混	多混	单混	多混	单混	多混	单混	多混
总IPC	−3.61	−1.13	−4.05	−4.80	−0.45	−2.78	0.71	−2.45
IPC1	−7.23	4.31	−12.13	−13.39	−3.58	3.29	0.33	8.10
IPC2	−9.78	−1.04	3.39	5.92	−3.82	−18.23	−0.20	−28.99
IPC3	7.32	−3.94	−5.55	−9.59	9.91	7.00	4.00	10.15
IPC4	−1.74	−17.02	−5.12	−8.56	−3.76	−1.90	−4.51	−3.41

从比较IQ 40下12组负载的IPC来看,IQ 40下新策略的效果没有IQ 24的好,因为大IQ能够更好地容忍对该资源的竞争[15].

4 结论

本文的研究表明,Static,2OP_BLOCK对于IQ竞争缓解作用明显,但为不同线程组合配置单一策略的方式,受到线程组合本身的限制.

本文所提出的结合线程特性的IQ竞争缓解策略能够加速执行原本IPC较高的线程,通过提升此类线程的性能,使整体IPC进一步增加,而部分线程较快结束也有利于新线程的进入和流水线资源的再分配.对于较小IQ的作用效果明显,相比Static最多获得7%性能提升.新策略的局限在于通过限制竞争较强的线程获得整体性能提升,期望负载中有竞

争较强的线程，因此，随着 IQ 容量增大，系统能够更好地容忍线程间的竞争，新策略的作用效果减小，当有多个竞争较弱线程的时候，也会出现效果降低的情况.

参考文献：

[1] Knijnenburg P M W, Ramirez A, Latorre F, et al. Branch classification to control instruction fetch in simultaneous multithreaded architectures [C] // Proceedings of the International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems. Kauai: IEEE Computer Society, 2002: 67-76.

[2] Raasch S E, Reinhardt S K. The impact of resource partitioning on SMT processors [C]// Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques. New Orleans: IEEE Computer Society/ACM/IFIP, 2003: 15-25.

[3] Joshua J Y, Ajay J, Resit S, et al. Analyzing the processor bottlenecks in SPEC CPU 2000 [C/CD]// Proceedings of the 2006 SPEC Benchmark Workshop. Austin: SPEC, 2006.

[4] 印杰. 同时多线程处理器容错技术研究[D]. 上海：同济大学电子与信息工程学院，2011.
YIN Jie. Research on fault-tolerant techniques for simultaneous multithreading processor [D]. Shanghai: College of Electronics and Information Engineering of Tongji University, 2011.

[5] Cazorla F J, Ramirez A, Valero M. Dcache warn: an i-fetch policy to increase SMT efficiency [C]// Proceedings of the 18th International Parallel and Distributed Processing Symposium. Santa Fe: IEEE Computer Society, 2004: 74-83.

[6] Sharkey J J, Ponomarev D V. Efficient instruction schedulers for SMT processors [C]// Proceedings of the 12th International Symposium on High-Performance Computer Architecture. Austin: IEEE Computer Society, 2006:288-298.

[7] Wang H, Sangireddy R, Baldawa S. Optimizing instruction scheduling through combined in-order and O-O-O execution in SMT processors [J]. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(3): 389.

[8] Brekelbaum E, Rupley J, Wilkerson C, et al. Hierarchical scheduling windows [C]// Proceedings of the 35th International Conference on Microarchitecture. Istanbul: IEEE Computer Society/ACM, 2002:27-36.

[9] Cristal A, Ortega D, Llosa J, et al. Out-of-order commit processors [C] // Proceedings of the 10th International Symposium on High Performance Computing. Madrid: IEEE Computer Society, 2004:48-59.

[10] 印杰，江建慧. 缓解同时多线程结构中线程对关键资源的竞争[J]. 计算机科学，2010，37(3)：256.
YIN Jie, JIANG Jianhui. Easing the competition for key resource among threads in SMT [J]. Computer Science, 2010, 37(3): 256.

[11] Tullsen D M, Eggers S J, Emer J S, et al. Exploiting choice: instruction fetch and issue on an implementable simultaneous multithreading processor [C] // Proceedings of 23nd Annual International Symposium on Computer Architecture. Philadephia: ACM/IEEE Computer Society, 1996:191-202.

[12] 张彩霞，张民选. Dwarn+：一种改进的同时多线程取指策略[J]. 小型微型计算机系统，2007，28(7)：1720.
ZHANG Caixia, ZHANG Minxuan. Dwarn +: An improved fetch strategy for simultaneous multithreading architecture [J]. Journal of Chinese Computer Systems, 2007, 28(7): 1720.

[13] Sharkey J J, Ponomarev D V. Balancing ILP and TLP in SMT architectures through out-of-order instruction dispatch [C]// Proceedings of the 2006 International Conference on Parallel Processing. Columbus: IEEE Computer Society, 2006: 329-336.

[14] Sherwood T, Perelman E, Hamerly G, et al. Automatically characterizing large scale program behavior [C]// Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems. San Jose: ACM, 2002: 47-57.

[15] 刘宇. 同时多线程处理器指令队列竞争的研究[D]. 上海：同济大学电子与信息工程学院，2013.
LIU Yu. Research on instruction queue competition of simultaneous multithreading processors [D]. Shanghai: College of Electronics and Information Engineering of Tongji University, 2013.