通过比较图 4(a)与图 4(b)的评估结果可见:虽然静态缺失次数的减少对于 WCET 评估值具有直接影响,但是二者间并无成比例变化关系.例如:使用 BBIP 预取机制后,*bs* 的静态指令缺失数的减少(小于 30%)不如 *crc*(约为 50%)显著;但是 *bs* 对应的 WCET 评估值的改进较 *crc* 更显著,分别约为 30%及小于 2.5%.这是因为对于不同的程序,指令访问缺失数与缺失率不同,因此,指令访问性能对于程序运行时间的影响程度不同.当程序具有较高的缺失次数及缺失率时,指令缺失次数变化对于程序执行时间的影响显著.此时,指令缺失次数显著下降将导致 WCET 评估值的显著下降(如程序 *jfdctint*).而对于像 *crc* 这类程序,指令访问数大(约为 21 464)而缺失率小(约为 0.17%),此时,指令缺失次数的减少不能显著影响程序的运行时间,因此 WCET 评估值的改进有限.

从图 4(a)及图 4(b)还可得出,在 3 种不同的处理器配置下,BBIP 预取机制均能够有效降低指令访问缺失数及 WCET 评估值.但是对于一些程序,如 *qsort-exam*,使用乱序执行流水线及 2-level 分支预测器时,指令访问缺失数及 WCET 评估值的改进相对另外两种处理器配置较小.这主要是因为使用 BBIP 预取机制后,错误分支预测导致的指令 Cache 内容污染可能由于基本块后续存储块的提前加载而加剧.对于 3 种不同的处理器配置,在使用 BBIP 预取机制后,测试程序的 WCET 评估值分别平均减小为原来的 81.4%,81.4%及 82.35%.

在通用计算机系统中,Next-N-Line 指令预取技术因实现简单、硬件开销小、指令访问性能提升明显,受到广泛的关注.但是在实时系统中,Next-N-Line 指令预取技术中存在的无效预取造成的 Cache 内容污染,可能影响任务的 WCET 评估值.

接下来比较使用 BBIP 与 Next-N-Line 指令预取技术后程序对应的 WCET 评估值.评估过程使用直接映射、容量分别为 512 及 1 024 字节、行大小为 32 字节的指令 Cache 配置.在 Next-N-Line 指令预取机制的评估过程中,设定预取步长为 2 个 Cache 行,即,每次访问指令 Cache 时预取下两个 Cache 行.如图 5 所示:在绝大部分情况下,BBIP 指令预取技术能够提供比 Next-N-Line 预取技术更小的 WCET 评估值(其中,基准值为使用 Next-N-Line 预取时任务对应的 WCET 评估值,预取步长为 2 个 Cache 行).这是因为程序中存在跳转及循环指令,使用 Next-N-Line 预取的预取行可能不被执行;并且将当前位于指令 Cache 中的有用 Cache 行替换出去,而在 BBIP 指令预取机制中不存在这种无效预取.从图 5 还可看出:当使用 512 字节大小的指令 Cache 时,对于测试程序 *qsort-exam*,使用 Next-N-Line 预取能够提供比 BBIP 预取略小的 WCET 评估值.这是因为在 *qsort-exam* 中,顺序的指令访问较多,因此 Next-N-Line 预取中的无效预取较少;而相反地,BBIP 指令预取技术不能避免基本块第 1 条指令的访问缺失,因此,BBIP 指令预取技术对应的 WCET 评估值略大.平均而言:对于 512 字节及 1 024 字节指令 Cache,相对于 Next-N-Line 预取技术,使用 BBIP 预取技术能够使得 WCET 评估值分别降低约 9.2%及 8.8%.

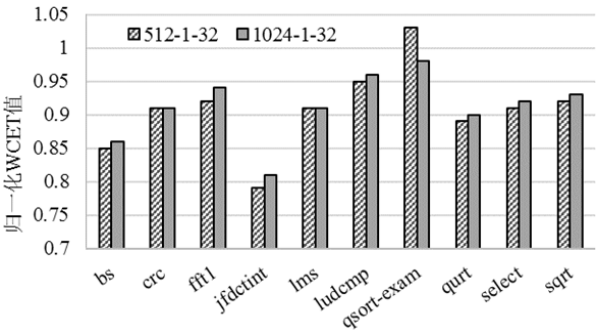

Fig.5 Normalized WCET estimates of BBIP prefetching with Next-N-Line prefetching as baseline

图 5 BBIP 预取相对于 Next-N-Line 指令预取的归一化 WCET 评估值

需要强调的是:在 Next-N-Line 预取方法 WCET 评估过程中,我们假设每次对指令 Cache 发起指令访问时均执行预取.基于该项假设,使用 Next-N-Line 预取后,Cache 访问缺失仅可能存在于某些基本块的第 1 条指令访问处,其甚至可以避免一些长度为 1 个存储块的基本块的指令访问缺失,而这在 BBIP 中无法保证.在真实硬件系统,这种访问时预取的实现方式通常会带来较大的访问时间及能耗开销,很少使用.因此,如果使用缺失时预取

的实现方案,则 BBIP 相对于 Next-N-Line 指令预取的 WCET 评估性能优势更为明显.另外,本文使用的测试程序规模较小,Next-N-Line 预取导致的 Cache 污染问题没有凸显.可以预见:随着实时任务程序规模的增加,Cache 污染问题的突出,BBIP 的优势将会更为明显.

6 结 论

为了简化 WCET 分析过程中指令 Cache 访问时间评估,减小实时任务 WCET 评估值,本文提出了一种基于程序基本块的指令预取方法(BBIP).在该预取方法中,每次指令访问发生缺失时,根据系统记录的基本块信息(如基本块起始地址及长度)对指令所在基本块的后续存储块执行指令预取.与现有应用于实时系统的指令预取技术相比,基于基本块的指令预取方法硬件实现开销小,且可有效避免无效预取及由此产生的指令 Cache 污染.另外,该预取方法能够极大地简化实时任务 WCET 分析,降低 WCET 评估值,从而改进实时系统任务可调度性分析.此外,与一些面向实时环境的新型 Cache 硬件结构相比,BBIP 预取方法能够以可选附属特性的形式存在于现有指令 Cache 结构中,适用性良好.

实时基准测试程序评估结果表明:BBIP 预取方法能够有效降低平均执行情况下的指令访问缺失数及任务执行时间;同时,在 WCET 分析中,具备 BBIP 预取功能的指令 Cache 能够将一部分原先静态判定为缺失的指令转化为访问命中,从而减少了静态指令访问缺失数,降低 WCET 评估值,有利于实时系统的可调度性分析.

致谢 在此,感谢新加坡国立大学发布了开源静态 WCET 评估工具 Chronos,并对其进行持续的支持与维护.此外,我们向对本文的工作给予支持和建议的同行,尤其是北京航空航天大学计算机学院的龙翔教授及其他老师和同学表示感谢.

References:

[1] Clifton C, Leavens GT, Chambers C, Millstein T. MultiJava: Modular open classes and symmetric multiple dispatch for Java. In: Rosson MB, Lea D, eds. Proc. of the 2000 ACM SIGPLAN Conf. on Object-Oriented Programming Systems, Languages & Applications. SIGPLAN Notices, 2000,35(10):130−145. [doi: 10.1145/353171.353181]

[2] Smith AJ. Sequential program prefetching in memory hierarchies. Computer, 1978,11(12):7−21. [doi: 10.1109/C-M.1978.218016]

[3] Smith AJ. Cache memories. ACM Computing Surveys (CSUR), 1982,14(3):473−530. [doi: 10.1145/356887.356892]

[4] Smith JE, Hsu WC. Prefetching in supercomputer instruction caches. In: Werner R, ed. Proc. of the 1992 ACM/IEEE Conf. on Supercomputing. IEEE Computer Society Press, 1992. 588−597. [doi: 10.1109/SUPERC.1992.236645]

[5] Pierce J, Mudge T. Wrong-Path instruction prefetching. In: Beaty S, Melvin S, eds. Proc. of the 29th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO-29). IEEE, 1996. 165−175. [doi: 10.1109/MICRO.1996.566459]

[6] Joseph D, Grunwald D. Prefetching using Markov predictors. In: Pleszkun AR, Mudge T, eds. Proc. of the 24th Annual Int'l Symp. on Computer architecture. ACM Press, 1997,25(2):252−263. [doi: 10.1145/264107.264207]

[7] Luk CK, Mowry TC. Cooperative prefetching: Compiler and hardware support for effective instruction prefetching in modern processors. In: Bondi J, Smith J, eds. Proc. of the 31st Annual ACM/IEEE Int'l Symp. on Microarchitecture (MICRO-31). IEEE Computer Society Press, 1998. 182−194. [doi: 10.1109/MICRO.1998.742780]

[8] Xia C, Torrellas J. Instruction prefetching of systems codes with layout optimized for reduced cache misses. In: Baer JL, ed. Proc. of the 23rd Annual Int'l Symp. on Computer Architecture. 1996,24(2):271−282. [doi: 10.1109/ISCA.1996.10019]

[9] Reinman G, Calder B, Austin T. Fetch directed instruction prefetching. In: Ronen R, Farrens M, Spillinger I, eds. Proc. of the 32nd Annual Int'l Symp. on Microarchitecture (MICRO-32). IEEE, 1999. 16−27. [doi: 10.1109/MICRO.1999.809439]

[10] Zhang Y, Haga S, Barua R. Execution history guided instruction prefetching. In: Ebcioglu K, Pingali K, Nicolau A, eds. Proc. of the 16th Int'l Conf. on Supercomputing. ACM Press, 2002. 199−208. [doi: 10.1145/514191.514220]

[11] Aamodt TM, Chow P, Hammarlund P, Wang H, Shen JP. Hardware support for prescient instruction prefetch. In: Tirado F, Zapata EL, eds. Proc. of the IEEE Int'l Symp. on High Performance Computer Architecture. IEEE Computer Society, 2004. 84. [doi: 10.1109/HPCA. 2004.10028]

[12] Yan J, Zhang W. WCET analysis of instruction caches with prefetching. In: Pande S, Li ZY, eds. Proc. of the 2007 ACM SIGPLAN/SIGBED Conf. on Languages, Compilers, and Tools for Embedded Systems. ACM SIGPLAN Notices, 2007,42(7): 175−184. [doi: 10.1145/1254766.1254801]

[13] Ding Y, Yan J, Zhang W. Optimizing instruction prefetching to improve worst-case performance for real-time applications. Journal of Computing Science and Engineering, 2009,3(1):59−71. [doi: 10.5626/JCSE.2009.3.1.059]

[14] Lee M, Min SL, Kim CS. A worst case timing analysis technique for instruction prefetch buffers. Microprocessing and Microprogramming, 1994,40(10):681−684. [doi: 10.1016/0165-6074(94)90017-5]

[15] Schoeberl M. A time predictable instruction cache for a Java processor. In: Zahir T, Angelo C, eds. Proc. of the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops. Berlin, Heidelberg: Springer-Verlag, 2004. 371−382. [doi: 10.1007/978-3-540-30470-8_52]

[16] Li XF, Roychoudhury A, Mitra T. Modeling out-of-order processors for WCET analysis. Journal of Real-time Systems, 2006,34(3): 195−227. [doi: 10.1007/s11241-006-9205-5]

[17] Austin T, Larson E, Ernst D. SimpleScalar: An infrastructure for computer system modeling. Computer, 2002,35(2):59−67. [doi: 10.1109/2.982917]

[18] Arnold R, Mueller F, Whalley D, Harmon M. Bounding worst-case instruction cache performance. In: Juan S, Rico P, eds. Proc. of the IEEE Real-Time Systems Symp. 1994. 172−181. [doi: 10.1109/REAL.1994.342718]

[19] Homepage of SNU real-time benchmark suite. 2007. http://archi.snu.ac.kr/realtime/benchmark/

王恩东(1964－),男,山东济南人,教授,CCF高级会员,主要研究领域为计算机系统结构,混合存储系统,可扩展系统互连协议分布对象计算.

倪璠(1984－),男,博士,工程师,主要研究领域为计算机系统结构,实时系统,操作系统.

陈继承(1976－),男,博士,高级工程师,CCF会员,主要研究领域为计算机体系结构,缓存一致性,集成电路设计.

王洪伟(1982－),男,博士,工程师,主要研究领域为计算机体系结构,高性能计算,人工智能.

唐士斌(1986－),男,博士,工程师,主要研究领域为计算机体系结构,缓存一致性,片上存储系统.