5.1.1 已有技术之间的融合

目前,各个领域的研究仍然相对封闭.本综述的目的之一就是对这些领域之间共同的技术予以总结,从而可以促进领域之间技术的融合.已有一些通过借鉴另一领域的技术进行实现的例子:文献[35]是软件实现的乐观MESI 协议;文献[23]提出的路径合成技术借鉴了硬件数值合成技术[79]中的约束可满足性问题构造;Zhang 等人提出了使用硬件事务内存实现数据竞争检测的加速[97].现有一些技术(如文献[62])在硬件上得以高效实现,但如何使用软件实现仍是挑战之一;反之,软件中研究的技术也可能集成到硬件实现中,其中一个例子是将动态锁指派技术[21]实现在硬件层次,将若干连续的缓存线进行绑定,作为原子处理.这一融合对硬件修改的复杂性较低,不仅能够降低访存依赖获取的开销,还可能利用运行时信息(如缓存线之间的空间和线程局部性)实现缓存预取,从而提高多核处理器的执行效率.

5.1.2 锁的动态自适应指派和实现

在软件访存依赖获取技术中,一类新兴的研究是在运行时自适应地调整锁指派,以提高简化性[21];或调整锁实现,以提高高效性[74].此类技术也是未来研究的可行方向.由于程序运行时具有各种局部性,这类新技术可能对访存依赖获取的高效性和简化性带来巨大的提升.然而,这两类研究目前都处于初级阶段:实施动态指派虽然能够提高简化性,但动态指派自身亦带来一定的时间开销,尤其是在动态指派机制复杂性增加后,这类开销更有可能掩盖其带来的益处,如何权衡这些要素,是这类技术未来的研究重点.除了动态锁指派和动态锁实现本身技术的研究以外,其适用的应用领域亦是未来研究的契机之一.特定的应用由于均摊到每一访存事件从而能够接受更高的代价(如软件事务内存[8]),此时,动态锁指派或动态锁实现技术带来的收益(如降低锁的数量和事务回滚的频率)可能远远大于其自身的开销,动态锁技术即可促进应用领域的进步.

推而广之,未来研究还可以实现追踪/合成技术的动态切换.Wu 等人观察到函数有轨迹记录和输出记录两种方式,从而使用网络最小割建模两种记录方式的代价并予以最优选择,从而实现高效记录[98].同理,注意到:路径合成技术[23]在访存事件多、执行路径简单(如生产者-消费者模型)的应用中能大幅提高记录的高效性和简化性;反之,若执行在访存少、路径复杂(如科学计算)的应用中则会记录大量不包含不确定性的分支,其性能甚至不如基于锁的追踪技术.如能在运行时刻根据程序自身特性,自适应地在多个技术之间融合,则能取长补短,实现更高效的访存依赖获取.

5.1.3 记录额外信息以降低合成复杂性

另一个研究方向是如何在运行时用较少的代价得到一些额外信息,从而大幅度降低访存依赖合成的复杂性.现已有一些工作利用了这类信息:Chen 等人利用了时钟信息减少依赖的数量[67];Liu 等人使用了程序中的同步操作来降低约束求解的难度[83].然而,目前还没有一个系统化的方式理解何种信息可以何种方式获取、有怎样的代价、对访存依赖获取有怎样的影响.对这一问题的深入理解,势必将推进访存依赖合成技术实用化的进程.

5.1.4 建模并发程序的执行

目前,一些研究工作[19,83]虽在实际中证实了有效性,但其合成访存依赖的时间复杂性仍未得到证明,我们目前猜想,它们的时间复杂性都是 NP-完全的.另一方面,这类技术(包括一些已经被证明是 NP-完全问题的技术[79])在实践中的表现都是非常优秀的.因此我们有理由相信,现实中并发程序的执行满足一定的特性,从而依赖合成在实际中的复杂性是较低的.框架合成技术[46,76]也隐含地利用了这一假设,且已经证实:在实际应用中,此类技术的开销是可以接受的.

如能准确地建模真实世界中并发程序的执行,则能更好地理解这类技术的优势与不足.基于这一模型,已有技术的优缺点均可在理论模型上予以分析;提出新技术的研究者在进行实例研究的同时,还能从理论上评估其表现(如在何种情况下表现为多项式时间复杂性,或在模型上对比与其他技术的开销).

6 总 结

本文讨论了访存依赖获取技术的核心要素:4 个评价指标(即时、准确、高效、简化)、两大类方法(在线合

成、离线获取)和两大类应用(轨迹分析、并发控制).本文提出的框架成功地提取了来自多个领域的并发程序动态分析技术中的共性问题,并能很好地理解这些技术因应用特征不同而在评价指标之间做出的权衡.此外,当前的框架中出现了一些空白,也为未来工作的开展奠定了基础.希望本文能进一步促进并发程序动态分析领域的研究发展,实现自动化的并发程序质量保障.

References:

[1] Hofstee HP. Future microprocessors and off-chip SOP interconnect. IEEE Trans. on Advanced Packaging, 2004,27(2):301−303. [doi: 10.1109/TADVP.2004.830355]

[2] Lu S, Park S, Seo E, Zhou Y. Learning from mistakes: A comprehensive study on real world concurrency bug characteristics. In: Eggers SJ, Larus JR, eds. Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). ACM, 2008. 329−339.

[3] Goodstein ML, Vlachos E, Chen S, Gibbons PB, Kozuch MA, Mowry TC. Butterfly analysis: Adapting dataflow analysis to dynamic parallel monitoring. In: Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2010. 257−270. [doi: 10.1145/1736020.1736050]

[4] Lu S, Tucek J, Qin F, Zhou Y. AVIO: Detecting atomicity violations via access interleaving invariants. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2006. 37−48. [doi: 10.1145/1168857.1168864]

[5] Sen K. Race directed random testing of concurrent programs. In: Proc. of the Conf. on Programming Language Design and Implementation (PLDI). 2008. 11−21. [doi: 10.1145/1375581.1375584]

[6] Leblanc TJ, Mellor-Crummey JM. Debugging parallel programs with instant replay. IEEE Trans. on Computers, 1987,C-36(4):471−482. [doi: 10.1109/TC.1987.1676929]

[7] Herlihy M, Moss JE. Transactional memory: Architectural support for lock-free data structures. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA). 1993. 289−300. [doi: 10.1145/165123.165164]

[8] Shavit N, Touitou D. Software transactional memory. In: Anderson JH, ed. Proc. of the Symp. on Principles of Distributed Computing (PODC). ACM, 1995. 204−213.

[9] Dice D, Shavit N. Understanding tradeoffs in software transactional memory. In: Proc. of the Int'l Symp. on Code Generation and Optimization (CGO). 2007. 21−33. [doi: 10.1109/CGO.2007.38]

[10] Devietti J, Nelson J, Bergan T, Ceze L, Grossman D. RCDC: A relaxed consistency deterministic computer. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2012. 67−78. [doi: 10.1145/1950365.1950376]

[11] Sengupta A, Biswas S, Zhang M, Bond MD, Kulkarni M. Hybrid static-dynamic analysis for statically bounded region serializability. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2015. 561−575. [doi: 10.1145/2694344.2694379]

[12] Devietti J, Lucia B, Ceze L, Oskin M. DMP: Deterministic shared memory multiprocessing. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2009. 85−96. [doi: 10.1145/1508244.1508255]

[13] Cui H, Simsa J, Lin Y, Li H, Blum B, Xu X, Yang J, Gibson GA, Bryant RE. Parrot: A practical runtime for deterministic, stable, and reliable threads. In: Proc. of the Symp. on Operating Systems Principles (SOSP). 2013. 388−405. [doi: 10.1145/2517349.2522735]

[14] Pusukuri KK, Gupta R, Bhuyan AN. Thread tranquilizer: Dynamically reducing performance variation. ACM Trans. on Architecture and Code Optimization, 2012,8(4):46−66. [doi: 10.1145/2086696.2086725]

[15] Gao L, Wang R, Qian DP. Deterministic replay for parallel programs in multi-core processors. Ruan Jian Xue Bao/Journal of Software, 2013,24(6):1390−1402 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4392.htm [doi: 10.3724/SP.J.1001.2013.04392]

[16] Chen Y, Zhang S, Guo Q, Li L, Wu R, Chen T. Deterministic replay: A survey. ACM Computing Surveys (CSUR), 2015,48(2):17−None. [doi: 10.1145/2790077]

[17] Hong S, Kim M. A survey of race bug detection techniques for multithreaded programs. Journal of Software: Testing, Verification and Reliability, 2015,25(3):191−217. [doi: 10.1002/stvr.1564]

[18] Su XH, Yu Z, Wang TT, Ma PJ. A survey on exposing, deteting and avoiding concurrency bugs. Chinese Journal of Computers, 2015,38(11):2215−2233 (in Chinese with English abstract).

[19] Jiang Y, Gu T, Xu C, Ma X, Lu J. CARE: Cache guided deterministic replay for concurrent Java programs. In: Jalote P, Briand LC, van der Hoek A, eds. Proc. of the Int'l Conf. on Software Engineering (ICSE). ACM, 2014. 457−467.

[20] Jiang Y, Li D, Xu C, Ma X, Lu J. Optimistic shared memory dependence tracing. In: Proc. of the Int'l Conf. on Automated Software Enginnering (ASE). 2015. 524−534. [doi: 10.1109/ASE.2015.11]

[21] Jiang Y, Xu C, Li D, Ma X, Lu J. Online shared memory dependence reduction via bisectional coordination. In: Proc. of the Symp. on the Foundations of Software Engineering (FSE). 2016. 822−832. [doi: 10.1145/2950290.2950326]

[22] Lamport L. How to make a multiprocessor computer that correctly executes multiprocess programs. IEEE Trans. on Computers, 1979,C-28(9):690−691. [doi: 10.1109/TC.1979.1675439]

[23] Huang J, Zhang C, Dolby J. CLAP: Recording local executions to reproduce concurrency failures. In: Proc. of the Conf. on Programming Language Design and Implementation (PLDI). 2013. 141−152. [doi: 10.1145/2491956.2462167]

[24] Adve SV, Boehm H. Memory models: A case for rethinking parallel languages and hardware. Communications of the ACM, 2010, 53(8):90−101. [doi: 10.1145/1787234.1787255]

[25] Ronsse M, De Bosschere K. RecPlay: A fully integrated practical record/replay system. ACM Trans. on Computer Systems, 1999, 17(2):133−152. [doi: 10.1145/312203.312214]

[26] Dinning A, Schonberg E. An empirical comparison of monitoring algorithms for access anomaly detection. In: Proc. of the Symp. on Principles and Practice of Parallel Programming (PPoPP). 1990. 1−10. [doi: 10.1145/99163.99165]

[27] Flanagan C, Freund SN. FastTrack: Efficient and precise dynamic race detection. In: Hind M, Diwan A, eds. Proc. of the Conf. on Programming Language Design and Implementation (PLDI). ACM, 2009. 121−133.

[28] Huang J, Liu P, Zhang C. LEAP: Lightweight deterministic multi-processor replay of concurrent Java programs. In: Roman G-C, Sullivan KJ, eds. Proc. of the Symp. on the Foundations of Software Engineering (FSE). ACM, 2010. 385−386.

[29] Yu Y, Rodeheffer T, Chen W. RaceTrack: Efficient detection of data race conditions via adaptive tracking. In: Proc. of the Symp. on Operating Systems Principles (SOSP). 2005. 221−234. [doi: 10.1145/1095809.1095832]

[30] Wilcox JR, Finch P, Flanagan C, Freund SN. Array shadow state compression for precise dynamic race detection. In: Proc. of the Int'l Conf. on Automated Software Enginnering (ASE). 2015. 155−165. [doi: 10.1109/ASE.2015.19]

[31] Huang J, Zhang C. An efficient static trace simplification technique for debugging concurrent programs. In: Proc. of the Int'l Conf. on Static Analysis (SAS). 2011. 163−179. [doi: 10.1007/978-3-642-23702-7_15]

[32] Netzer RHB. Optimal tracing and replay for debugging shared-memory parallel programs. In: Proc. of the Workshop on Parallel and Distributed Debugging (PADD). 1993. 1−11. [doi: 10.1145/174266.174268]

[33] Jalbert N, Sen K. A trace simplification technique for effective debugging of concurrent programs. In: Proc. of the Symp. on the Foundations of Software Engineering (FSE). 2010. 57−66. [doi: 10.1145/1882291.1882302]

[34] Bhansali S, Chen W, de Jong S, Edwards A, Murray R, Drinić M, Mihočka D, Chau J. Framework for instruction-level tracing and analysis of program executions. In: Proc. of the Int'l Conf. on Virtual Execution Environments (VEE). 2006. 154−163. [doi: 10.1145/1134760.1220164]

[35] Bond MD, Kulkarni M, Cao M, Zhang M, Fathi Salmi M, Biswas S, Sengupta A, Huang J. OCTET: Capturing and controlling cross-thread dependences efficiently. In: Proc. of the Int'l Conf. on Object Oriented Programming Systems Languages and Applications (OOPSLA). 2013. 693−712. [doi: 10.1145/2509136.2509519]

[36] Bacon DF, Goldstein SC. Hardware-Assisted replay of multiprocessor programs. In: Proc. of the Workshop on Parallel and Distributed Debugging (PADD). 1991. 194−206. [doi: 10.1145/122759.122777]

[37] Attiya H, Guerraoui R, Hendler D, Kuznetsov P, Michael MM, Vechev M. Laws of order: Expensive synchronization in concurrent algorithms cannot be eliminated. In: Proc. of the Symp. on Principles of Programming Languages (POPL). 2011. 487−498. [doi: 10.1145/1926385.1926442]

[38] Xu M, Bodik R, Hill MD. A flight data recorder for enabling full-system multiprocessor deterministic replay. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA). 2003. 122−135. [doi: 10.1145/859618.859633]

[39] von Praun C, Gross TR. Object race detection. In: Proc. of the Int'l Conf. on Object Oriented Programming Systems Languages and Applications (OOPSLA). 2001. 70−82. [doi: 10.1145/504282.504288]

[40] Dunlap GW, Lucchetti DG, Fetterman MA, Chen PM. Execution replay of multiprocessor virtual machines. In: Proc. of the Int'l Conf. on Virtual Execution Environments (VEE). 2008. 121−130. [doi: 10.1145/1346256.1346273]

[41] Choi J, Gupta M, Serrano M, Sreedhar VC, Midkiff S. Escape analysis for Java. In: Proc. of the Int'l Conf. on Object Oriented Programming Systems Languages and Applications (OOPSLA). 1999. 1−19. [doi: 10.1145/320384.320386]

[42] Voung JW, Jhala R, Lerner S. RELAY: Static race detection on millions of lines of code. In: Crnkovic I, Bertolino A, eds. Proc. of the Joint Meeting of the European Software Engineering Conf. and the Symp. on the Foundations of Software Engineering (ESEC/FSE). ACM, 2007. 205−214.

[43] Zamfir C, Candea G. Execution synthesis: A technique for automated software debugging. In: Proc. of the European Conf. on Computer Systems (EuroSys). 2010. 321−334. [doi: 10.1145/1755913.1755946]

[44] Huang J, Meredith PO, Rosu G. Maximal sound predictive race detection with control flow abstraction. In: Proc. of the Conf. on Programming Language Design and Implementation (PLDI). 2014. 337−348. [doi: 10.1145/2594291.2594315]

[45] Nieuwenhuis R, Oliveras A, Tinelli C. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL. Journal of the ACM, 2006,53(6):937−977. [doi: 10.1145/1217856.1217859]

[46] Park S, Zhou Y, Xiong W, Yin Z, Kaushik R, Lee KH, Lu S. PRES: Probabilistic replay with execution sketching on multiprocessors. In: Proc. of the Symp. on Operating Systems Principles (SOSP). 2009. 177−192. [doi: 10.1145/1629575.1629593]

[47] Zhou J, Xiao X, Zhang C. Stride: Search-Based deterministic replay in polynomial time via bounded linkage. In: Proc. of the Int'l Conf. on Software Engineering (ICSE). 2012. 892−902. [doi: 10.1109/ICSE.2012.6227130]

[48] Savage S, Burrows M, Nelson G, Sobalvarro P, Anderson T. Eraser: A dynamic data race detector for multithreaded programs. ACM Trans. on Computer Systems, 1997,15(4):391−411. [doi: 10.1145/265924.265927]

[49] Serbanuta TF, Chen F, Rosu G. Maximal causal models for sequentially consistent systems. In: Proc. of the Int'l Conf. on Runtime Verification (RV). 2012. 136−150. [doi: 10.1007/978-3-642-35632-2_16]

[50] Smaragdakis Y, Evans J, Sadowski C, Yi J, Flanagan C. Sound predictive race detection in polynomial time. In: Proc. of the Symp. on Principles of Programming Languages (POPL). 2012. 387−400. [doi: 10.1145/2103656.2103702]

[51] Dolby J, Hammer C, Marino D, Tip F, Vaziri M, Vitek J. A data-centric approach to synchronization. ACM Trans. on Programming Languages and Systems, 2012,34(1):4:1−4:48. [doi: 10.1145/2160910.2160913]

[52] Harris T, Cristal A, Unsal OS, Ayguade E, Gagliardi F, Smith B, Valero M. Transactional memory: An overview. IEEE Micro, 2007,27(3):8−29. [doi: 10.1109/MM.2007.63]

[53] Yoo RM, Hughes CJ, Lai K, Rajwar R. Performance evaluation of Intel(R) transactional synchronization extensions for high-performance computing. In: Proc. of the Int'l Conf. for High Performance Computing (SC). 2013. 1−11. [doi: 10.1145/2503210.2503232]

[54] Zhang M, Huang J, Cao M, Bond MD. Low-Overhead software transactional memory with progress guarantees and strong semantics. In: Proc. of the Symp. on Principles and Practice of Parallel Programming (PPoPP). 2015. 97−108. [doi: 10.1145/2688500.2688510]

[55] Min SL, Choi J. An efficient cache-based access anomaly detection scheme. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 1991. 235−244. [doi: 10.1145/106972.106996]

[56] Ananian CS, Asanovic K, Kuszmaul BC, Leiserson CE, Lie S. Unbounded transactional memory. IEEE Micro, 2006,26(1):59−69. [doi: 10.1109/MM.2006.26]

[57] Lucia B, Ceze L, Strauss K, Qadeer S, Boehm H. Conflict exceptions: Simplifying concurrent language semantics with precise hardware exceptions for data-races. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA). 2010. 210−221. [doi: 10.1145/1815961.1815987]

[58] Hower DR, Hill MD. Rerun: Exploiting episodes for lightweight memory race recording. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA). 2008. 265−276. [doi: 10.1109/ISCA.2008.26]

[59] Montesinos P, Ceze L, Torrellas J. DeLorean: Recording and deterministically replaying shared-memory multiprocessor execution efficiently. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA). 2008. 289−300. [doi: 10.1109/ISCA.2008.36]

[60] Yang Z, Yang M, Zang B, Xu L, Chen H. ORDER: Object centric deterministic replay for Java. In: Nieh J, Waldspurger CA, eds. Proc. of the USENIX Annual Technical Conf. (ATC). USENIX Association, 2011. 1−14.

[61] Ceze L, Tuck J, Torrellas J, Cascaval C. Bulk disambiguation of speculative threads in multiprocessors. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA). 2006. 227−238. [doi: 10.1109/ISCA.2006.13]

[62] Ceze L, Tuck J, Montesinos P, Torrellas J. BulkSC: Bulk enforcement of sequential consistency. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA). 2007. 278−289. [doi: 10.1145/1250662.1250697]

[63] Elmas T, Qadeer S, Tasiran S. Goldilocks: A race and transaction-aware Java runtime. In: Proc. of the Conf. on Programming Language Design and Implementation (PLDI). 2007. 245−255. [doi: 10.1145/1250734.1250762]

[64] Cherem S, Chilimbi T, Gulwani S. Inferring locks for atomic sections. In: Proc. of the Conf. on Programming Language Design and Implementation (PLDI). 2008. 304−315. [doi: 10.1145/1375581.1375619]

[65] Lee D, Chen PM, Flinn J, Narayanasamy S. Chimera: Hybrid program analysis for determinism. In: Proc. of the Conf. on Programming Language Design and Implementation (PLDI). 2012. 463−474. [doi: 10.1145/2254064.2254119]

[66] Xu M, Hill MD, Bodik R. A regulated transitive reduction (RTR) for longer memory race recording. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2006. 49−60. [doi: 10.1145/1168857.1168865]

[67] Chen Y, Hu W, Chen T, Wu R. LReplay: A pending period based deterministic replay scheme. In: Proc. of the Int'l Symp. on Computer Architecture (ISCA). 2010. 187−197. [doi: 10.1145/1815961.1815985]

[68] Berger ED, Yang T, Liu T, Novark G. Grace: Safe multithreaded programming for C/C++. In: Proc. of the Int'l Conf. on Object Oriented Programming Systems Languages and Applications (OOPSLA). 2009. 81−96. [doi: 10.1145/1640089.1640096]

[69] Liu T, Curtsinger C, Berger ED. Dthreads: Efficient deterministic multithreading. In: Proc. of the Symp. on Operating Systems Principles (SOSP). 2011. 327−336. [doi: 10.1145/2043556.2043587]

[70] Bergan T, Hunt N, Ceze L, Gribble SD. Deterministic process groups in dOS. In: Arpaci-Dusseau RH, Chen B, eds. Proc. of the Conf. on Operating Systems Design and Implementation (OSDI). USENIX Association, 2010. 177−191.

[71] Patil H, Pereira C, Stallcup M, Lueck G, Cownie J. PinPlay: A framework for deterministic replay and reproducible analysis of parallel programs. In: Proc. of the Int'l Symp. on Code Generation and Optimization (CGO). 2010. 2−11. [doi: 10.1145/1772954.1772958]

[72] Russell K, Detlefs D. Eliminating synchronization-related atomic operations with biased locking and bulk rebiasing. In: Proc. of the Int'l Conf. on Object Oriented Programming Systems Languages and Applications (OOPSLA). 2006. 263−272. [doi: 10.1145/1167473.1167496]

[73] Chen Y, Chen H. Scalable deterministic replay in a parallel full-system emulator. In: Proc. of the Symp. on Principles and Practice of Parallel Programming (PPoPP). 2013. 207−218. [doi: 10.1145/2442516.2442537]

[74] Cao M, Zhang M, Sengupta A, Bond MD. Drinking from both glasses: Combining pessimistic and optimistic tracking of cross-thread dependences. In: Proc. of the Symp. on Principles and Practice of Parallel Programming (PPoPP). 2016. 20:1−20:13. [doi: 10.1145/2851141.2851143]

[75] Honarmand N, Torrellas J. RelaxReplay: Record and replay for relaxed-consistency multiprocessors. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2014. 223−238. [doi: 10.1145/2541940.2541979]

[76] Altekar G, Stoica I. ODR: Output-Deterministic replay for multicore debugging. In: Proc. of the Symp. on Operating Systems Principles (SOSP). 2009. 193−206. [doi: 10.1145/1629575.1629594]

[77] Netzer RH, Miller BP. On the complexity of event ordering for shared-memory parallel program executions. In: Wahhabi BW, ed. Proc. of the Int'l Conf. on Parallel Processing (ICPP). Pennsylvania State University Press, 1990. 93−97.

[78] Gibbons PB, Korach E. Testing shared memories. SIAM Journal on Computing, 1997,26(4):1208−1244. [doi: 10.1137/S0097539794279614]

[79] Lee D, Said M, Narayanasamy S, Yang Z, Pereira C. Offline symbolic analysis for multi-processor execution replay. In: Albonesi DH, Martonosi M, August DI, Martínez JF, eds. Proc. of the Int'l Symp. on Microarchitecture (MICRO). ACM, 2009. 564−575.

[80] Narayanasamy S, Pereira C, Calder B. Recording shared memory dependencies using strata. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2006. 229−240. [doi: 10.1145/1168857.1168886]

[81] Yuan X, Wu C, Wang Z, Li J, Yew P, Huang J, Feng X, Lan Y, Chen Y, Guan Y. ReCBuLC: Reproducing concurrency bugs using local clocks. In: Proc. of the Int'l Conf. on Software Engineering (ICSE). 2015. 824−834. [doi: 10.1109/ICSE.2015.94]

[82] Lee D, Said M, Narayanasamy S, Yang Z. Offline symbolic analysis to infer total store order. In: Proc. of the 2011 Int'l Symp. on High Performance Computer Architecture (HPCA). 2011. 357−358. [doi: 10.1109/HPCA.2011.5749743]

[83] Liu P, Zhang X, Tripp O, Zheng Y. Light: Replay via tightly bounded recording. In: Proc. of the Conf. on Programming Language Design and Implementation (PLDI). 2015. 55−64. [doi: 10.1145/2737924.2738001]

[84] Lamport L. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 1978,21(7):558−565. [doi: 10.1145/359545.359563]

[85] Huang J, Luo Q, Rosu G. GPredict: Generic predictive concurrency analysis. In: Proc. of the Int'l Conf. on Software Engineering (ICSE). 2015. 847−857. [doi: 10.1109/ICSE.2015.96]

[86] Huang J, Zhang C. Persuasive prediction of concurrency access anomalies. In: Proc. of the Int'l Symp. on Software Testing and Analysis (ISSTA). 2011. 144−154. [doi: 10.1145/2001420.2001438]

[87] Liu T, Berger ED. SHERIFF: Precise detection and automatic mitigation of false sharing. In: Proc. of the Int'l Conf. on Object Oriented Programming Systems Languages and Applications (OOPSLA). 2011. 3−18. [doi: 10.1145/2048066.2048070]

[88] Musuvathi M, Qadeer S, Ball T, Basler G, Nainar PA, Neamtiu I. Finding and reproducing Heisenbugs in concurrent programs. In: Draves R, van Renesse R, eds. Proc. of the Conf. on Operating Systems Design and Implementation (OSDI). USENIX Association, 2008. 267−280.

[89] Park S, Lu S, Zhou Y. CTrigger: Exposing atomicity violation bugs from their hiding places. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2009. 25−36. [doi: 10.1145/1508244.1508249]

[90] Cai Y, Cao L. Effective and precise dynamic detection of hidden races for Java programs. In: Proc. of the Joint Meeting of the European Software Engineering Conf. and the Symp. on the Foundations of Software Engineering (ESEC/FSE). 2015. 450−461. [doi: 10.1145/2786805.2786839]

[91] Dice D, Shavit N. TLRW: Return of the read-write lock. In: Proc. of the Symp. on Parallelism in Algorithms and Architectures (SPAA). 2010. 284−293. [doi: 10.1145/1810479.1810531]

[92] Dalessandro L, Spear MF, Scott ML. NOrec: Streamlining STM by abolishing ownership records. In: Proc. of the Symp. on Principles and Practice of Parallel Programming (PPoPP). 2010. 67−78. [doi: 10.1145/1693453.1693464]

[93] Biswas S, Zhang M, Bond MD, Lucia B. Valor: Efficient, software-only region conflict exceptions. In: Aldrich J, Eugster P, eds. Proc. of the Int'l Conf. on Object Oriented Programming Systems Languages and Applications (OOPSLA). USENIX Association, 2015. 241−259.

[94] Bergan T, Anderson O, Devietti J, Ceze L, Grossman D. CoreDet: A compiler and runtime system for deterministic multithreaded execution. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2010. 53−64. [doi: 10.1145/1736020.1736029]

[95] Cui H, Wu J, Tsai C, Yang J. Stable deterministic multithreading through schedule memoization. In: Arpaci-Dusseau RH, Chen B, eds. Proc. of the Conf. on Operating Systems Design and Implementation (OSDI). USENIX Association, 2010. 207−221.

[96] Cui H, Wu J, Gallagher J, Guo H, Yang J. Efficient deterministic multithreading through schedule relaxation. In: Proc. of the Symp. on Operating Systems Principles (SOSP). 2011. 337−351. [doi: 10.1145/2043556.2043588]

[97] Zhang T, Lee D, Jung C. TxRace: Efficient data race detection using commodity hardware transactional memory. In: Proc. of the Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2016. 159−173. [doi: 10.1145/2872362.2872384]

[98] Wu M, Long F, Wang X, Xu Z, Lin H, Liu X, Guo Z, Guo H, Zhou L, Zhang Z. Language-based replay via data flow cut. In: Proc. of the Symp. on the Foundations of Software Engineering (FSE). 2010. 197−206. [doi: 10.1145/1882291.1882322]

附中文参考文献:

[15] 高岚,王锐,钱德沛.多核处理器并行程序的确定性重放研究.软件学报,2013,24(6):1390−1402. http://www.jos.org.cn/1000-9825/4392.htm [doi: 10.3724/SP.J.1001.2013.04392]

[18] 苏小红,禹振,王甜甜,马培军.并发缺陷暴露、检测与规避研究综述.计算机学报,2015,38(11):2215−2233.

蒋炎岩(1988－),男,江苏南京人,学士,主要研究领域为软件测试与分析.

马晓星(1975－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为自适应软件系统,软件工程.

许畅(1977－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为大数据软件工程,软件测试与分析,自适应与嵌入式系统.

吕建(1960－),男,博士,教授,博士生导师,CCF 会士,主要研究领域为大数据软件工程,软件测试与分析,自适应与嵌入式系统.