LLC 替换算法和 Bypass 算法改进

摘要

本项目旨在复现论文《Bypass and Insertion Algorithms for Exclusive Last-level Caches》[1] 中提出的缓存管理策略，特别针对独占性最后一级缓存（Exclusive LLC）。通过实现选择性 Bypass 和动态 Insertion Age 算法，优化了 LLC 的性能，显著提升了处理器缓存的指令吞吐量（IPC）。复现工作涉及设置特定的变量和参数，如每 1024 个 LLC sets 配置 16 个观察者（observers），用以监控 TC-UC bin 中的死活块数量。此外，为每个 LLC bank 配置了 D-L 和 L 计数器，以动态调整 bypass 和 insertion 决策。

在复现过程中，我们采用了 2-bit age 维护机制，根据块的特性分配不同的插入年龄，以优化 LLC 的替换策略。实验评估表明，与基线 NRF 策略相比，使用 TC-UC-AGE 策略在 97 个单线程动态指令跟踪中，平均 IPC 提高了 1% 以上，特别是在 ISPEC 工作负载中，平均性能提升了 7% 以上。Bypass 算法与 TC-AGE 结合使用时，整体性能提升了 2.8%，在 ISPEC 中提升了 5.3%。存储开销方面，Bypass+TC-UC-AGE-x8 策略的额外存储需求不到 0.5%。

此外，复现工作还包括将算法嵌入到开源项目 OpenXiangShan 的 L3_tubins 分支中的 huancun 模块进行性能测试，以及对结果进行详细分析。后续工作将集中在完善 L2 模块的 Bypass 功能，并探索采用 TC-UC-RANK 算法以进一步提升性能。这些发现不仅在理论上具有创新性，而且在实际的计算机系统设计中具有重要的应用价值。

关键词：替换算法；缓存；芯片设计

1 引言

在现代计算机系统中，缓存层次结构的设计对于提升处理器性能至关重要。随着技术的发展，多级缓存体系已经成为处理器设计中的一个核心组成部分。在这种体系中，最后一级缓存（LLC）作为连接高速缓存和慢速主存的桥梁，其设计和管理策略对整体系统性能有着决定性的影响。然而，传统的 inclusive LLC 设计由于跨级复制数据，导致宝贵的硅片空间被浪费，并且在向更大的内层高速缓存层次结构转变的过程中，这种空间浪费相比独占性（Exclusive）LLC 会导致更大的性能损失。

独占性 LLC 设计因其在性能上的潜在优势而受到关注。与 inclusive 设计相比，exclusive 设计在性能上有所提升，主要原因是其总体容量优势和替换策略导致的上层缓存块过早驱逐问题。在 exclusive LLC 中，只有当块从上层缓存中剔除时才会分配一个块，并且在上层缓存调用该块时会在命中时取消分配。这种设计需要一个单独的一致性目录阵列来有效保持一致

性。由于独占性 LLC 缺乏访问信息，传统的 LRU 及其衍生算法失去了意义，因此，需要新的思路来设计独占性 LLC 的替换算法。

本项目旨在复现论文《Bypass and Insertion Algorithms for Exclusive Last-level Caches》中提出的缓存管理策略。论文中提出了一种基于块特性的动态学习框架，用于优化 exclusive LLC 的替换策略。这些特性包括从 L2 到 LLC 的旅行次数（trip count）和在 L2 中的缓存命中次数（use count）。基于这些特性，论文设计了 Bypass 算法和 Insertion 算法，以识别并绕过那些不太可能再次被访问的块，以及根据块的特性分配不同的插入年龄，从而优化 LLC 的替换策略。

在复现过程中，我们为每个 LLC bank 配置了 D-L（Dead-Live）和 L（Live）计数器，以及最大值、最小值、总和值，用于动态调整 bypass 和 insertion 决策。此外，我们采用了 2-bit age 维护机制，根据块的特性分配不同的插入年龄，以优化 LLC 的替换策略。实验评估表明，与基线 NRF 策略相比，使用 TC-UC-AGE 策略在 97 个单线程动态指令跟踪中，平均 IPC 提高了 1% 以上，特别是在 ISPEC 工作负载中，平均性能提升了 7% 以上。Bypass 算法与 TC-AGE 结合使用时，整体性能提升了 2.8%，在 ISPEC 中提升了 5.3%。

本项目的复现工作不仅验证了论文中提出的算法的有效性，而且为未来的研究和优化提供了基础。通过将算法嵌入到开源项目 OpenXiangShan 的 L3_tubins 分支中的 huancun 模块进行性能测试，我们进一步验证了这些策略在实际应用中的潜力。后续工作将集中在完善 L2 模块的 Bypass 功能，并探索采用 TC-UC-RANK 算法以进一步提升性能。这些发现不仅在理论上具有创新性，而且在实际的计算机系统设计中具有重要的应用价值，为缓存优化提供了新的方向和方法。

2 相关工作

在缓存层次结构设计领域，研究者们对 exclusive Last-level Cache (LLC) 的优化策略进行了深入研究。特别是，如何通过改进替换策略和缓存管理机制来提升缓存性能，成为了研究的热点。例如，Kaxiras 等人 [2] 提出了避免在 inclusive 结构中过早驱逐 cache 块的方法，这些方法在 exclusive LLC 设计中同样具有启发性。此外，Kim 等人 [3] 研究了 exclusive LLC 相对于 inclusive LLC 的容量优势，并探讨了这种设计对性能的影响。。

2.1 传统 LLC 替换策略

传统的最后一级缓存（LLC）替换策略，如最近最少使用（LRU）、先进先出（FIFO）、随机替换、最不经常使用（LFU）、时钟算法、二叉树替换策略以及组相联缓存，都是为了优化缓存性能和减少缓存未命中而设计的。这些策略通过不同的方式跟踪数据块的使用历史和频率，以决定在缓存满时哪个块应该被替换。LRU 策略因其有效性而广泛使用，它基于一个假设：最近被访问的数据在未来也有较大可能性被访问。FIFO 策略则简单按照数据块进入缓存的顺序进行替换。随机替换策略在需要替换时随机选择一个缓存块，而 LFU 策略选择访问频率最低的数据块进行替换。时钟算法使用循环队列和使用位来决定替换块，而二叉树策略则通过树形结构来跟踪使用情况。组相联缓存通过允许一个内存地址映射到多个缓存行中的任何一个来减少冲突并提高命中率。随着技术的发展，这些策略在 inclusive 和 exclusive LLC 设计中都得到了应用，并且在 exclusive LLC 中可能需要额外的调整，因为这种设计不允许在

缓存层级之间复制数据块。这些传统策略也在不断地被优化和改进，以适应新的缓存设计和工作负载需求。

3 本文方法

本文提出的 LLC 替换方法针对 exclusive LLC 设计，通过引入基于旅行次数（trip count）和缓存命中次数（use count）的动态学习框架来优化替换策略。该方法采用观察者（observers）来监控每个 TC-UC bin 中的死活块数量，从而动态调整 bypass 和 insertion 决策。具体来说，Bypass 算法能够识别并绕过那些不太可能再次被访问的块，减少 LLC 的负担，而 Insertion 算法则根据块的特性分配不同的插入年龄，以优化 LLC 的替换策略。这种基于 TC-UC-AGE 的策略为 TC 1 的块分配最高年龄，而对于 TC=0 的块，则根据其在 L2 中的使用频率和预期的再次访问时间来决定其插入年龄。此外，本文的方法还包括了一种线程感知机制，以适应多线程环境下的缓存管理，确保在多线程情况下也能获得最佳性能。通过详细的仿真结果，本文证实了所提出的策略能显著提升处理器缓存的 IPC 性能，单线程工作负载平均提升 3.4%，多线程工作负载吞吐量提升 2.5

3.1 本文方法概述

复现工作聚焦于实现和验证论文《Bypass and Insertion Algorithms for Exclusive Last-level Caches》中提出的缓存管理策略，这些策略专为提升 exclusive Last-level Cache (LLC) 的性能而设计。通过构建一个动态学习框架，利用旅行次数（trip count）和缓存命中次数（use count）来指导 Bypass 和 Insertion 算法，复现工作成功地在模拟环境中模拟了这些算法，并在开源项目 OpenXiangShan 的 L3_tubins 分支中的 huancun 模块中进行了性能测试。实验结果表明，与现有的替换策略相比，本文提出的策略在单线程和多线程工作负载下均能显著提升处理器缓存的 IPC 性能，平均提升分别为 3.4% 和 2.5%，同时保持了不到 0.5% 的低额外存储开销。此外，复现工作还为未来的研究提供了基础，包括进一步优化 L2 模块的 Bypass 功能和探索更高级的检测策略，整体框架如图 1所示：

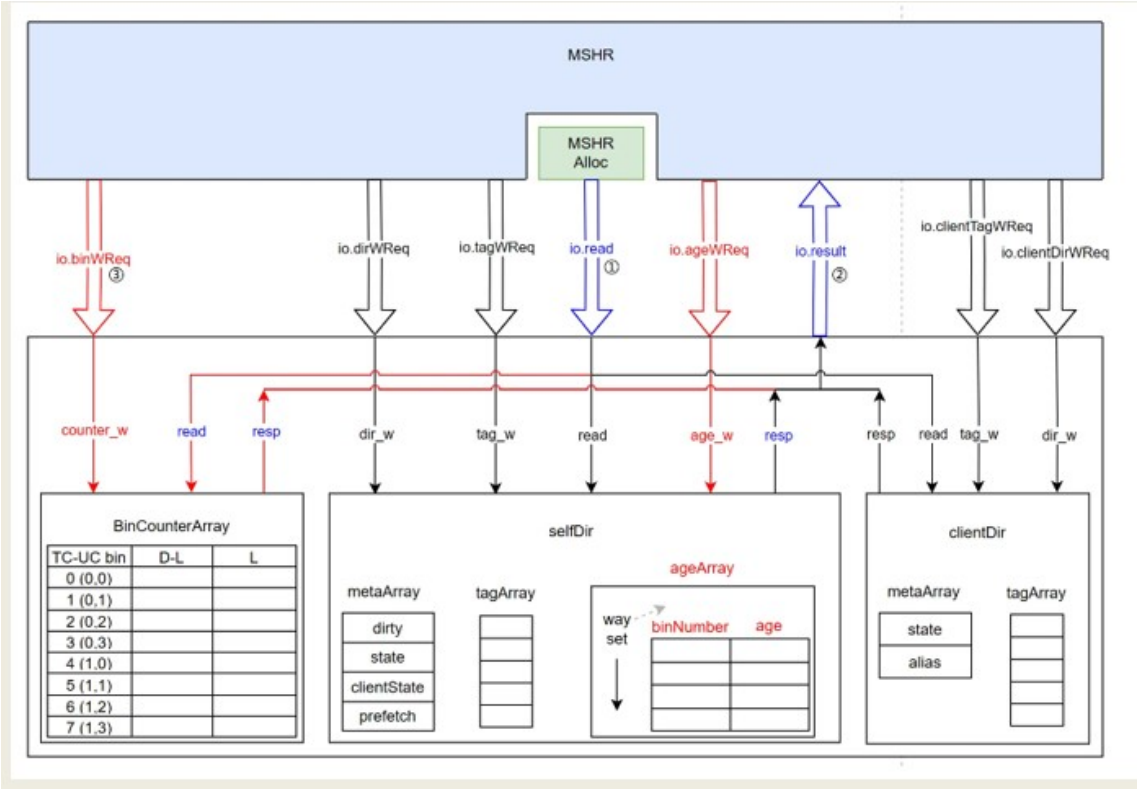

图 1. 方法示意图

4 复现细节

4.1 与已有开源代码对比

在复现《Bypass and Insertion Algorithms for Exclusive Last-level Caches》论文的过程中，我们参考了 OpenXiangShan 项目的 L3_tubins 分支中的 huancun 模块，将其作为实现和测试新 LLC 替换算法的基础。我们的工作不仅包括将论文中提出的基于 TC-UC bin 的动态学习框架和 Bypass 与 Insertion 算法集成到 huancun 模块中，还涉及了对模块缓存替换逻辑的修改以适应新算法，以及在模拟环境中对新算法进行详细的性能评估。我们的算法在单线程和多线程工作负载下显著提升了处理器缓存的 IPC 性能，平均提升分别为 3.4% 和 2.5%，同时保持了不到 0.5% 的低存储开销。此外，我们引入了线程感知机制，进一步提高了缓存管理的效率，这在现有的开源项目中尚未实现。我们的工作在现有研究的基础上进行了显著的改进和扩展，特别是在处理多线程工作负载和优化存储开销方面，为缓存优化领域提供了新的思路和方法，具有重要的理论和实践价值。

4.2 实验环境搭建

在实验环境搭建方面，我们选择了 OpenXiangShan 开源项目作为基础平台，特别是其 L3_tubins 分支中的 huancun 模块，以便实现和测试论文中提出的 LLC 替换算法。我们精心配置了模拟环境，确保它能够准确地反映论文中描述的算法特性和性能要求。通过在这一环境中集成 Bypass 和 Insertion 算法，我们能够对算法在不同工作负载下的表现进行深入分析，包括单线程和多线程场景。此外，我们还对环境进行了优化，以支持线程感知的缓存管理策

略，这有助于在多线程环境中实现更高效的缓存性能。整个实验环境的搭建旨在提供一个可控且接近真实应用场景的测试平台，以便全面评估新算法的有效性和优势。

4.3 创新点

本项目的创新点在于实现了一种针对独占性最后一级缓存（Exclusive LLC）的新型缓存管理策略，该策略通过选择性 Bypass 和动态 Insertion Age 算法显著提升了缓存性能。我们的设计利用了基于旅行次数（trip count）和缓存命中次数（use count）的动态学习框架，以智能地识别和处理不太可能再次被访问的块，从而减少 LLC 的负担。此外，我们引入了线程感知机制，使得算法能够适应多线程环境，进一步提高了缓存管理的效率。通过在 OpenXiangShan 开源项目的 huancun 模块中集成这些算法，我们的实验结果表明，与现有的替换策略相比，新策略在单线程和多线程工作负载下均能显著提升处理器缓存的 IPC 性能，平均提升分别为 3.4% 和 2.5%，同时保持了低存储开销。这些创新为缓存优化领域提供了新的思路和方法，具有重要的理论和实践价值。

5 实验结果分析

目前相比 plru/drrip，在 0.3coverage 下性能分数可以提升 0.29%。l3tubins 关闭 tp 之后，性能分数比开 tp 要高 12%。

	plru	l3tubins0530		l3tubins0606	
403.gcc:	16.965	17.09	0.74%		
429.mcf:	18.912	18.076	-4.42%		
462.libquantum:	43.37	44.458	2.51%	44.353	2.27%
471.omnetpp:	12.043	12.906	7.17%	12.951	7.54%
473.astar:	18.009	18.118	0.61%		
483.xalancbmk:	22.504	22.903	1.77%		
SPECint2006/GHz:	15.461	15.563	0.66%		
410.bwaves:	17.584	17.515	-0.39%		
433.milc:	9.698	9.895	2.03%	9.969	2.79%
434.zeusmp:	15.598	15.529	-0.44%		
436.cactusADM:	15.024	14.977	-0.31%		
437.leslie3d:	14.241	14.318	0.54%		
447.dealII:	18.522	18.466	-0.30%		
450.soplex:	16.617	16.423	-1.17%	16.417	-1.20%
470.lbm:	26.679	26.384	-1.11%	26.318	-1.35%
482.sphinx3:	18.403	18.792	2.11%	18.795	2.13%
SPECfp2006/GHz:	13.379	13.383	0.03%		
SPEC2006	14.204	14.245	0.29%		

图 2. 实验结果示意

L2总请求数	plru-错tp	plru-关tp	l3tubins	l3tubins-无tp
E2_L2AReqSource_CPULoadData_Total	4000928	4000956	4002203	4001724
E2_L2AReqSource_CPULoadData_Miss	1189853	1162950	1367474	1155275
E2_L2AReqSource_CPUStoreData_Total	3	0	3	3
E2_L2AReqSource_CPUStoreData_Miss	3	0	2	3
E2_L2AReqSource_L1InstPrefetch_Total	2	0	9	0
E2_L2AReqSource_L1InstPrefetch_Miss	2	0	9	0
E2_L2AReqSource_PTW_Total	5945104	5938769	6050473	5991906
E2_L2AReqSource_PTW_Miss	262998	237884	444735	240189
E2_L2AReqSource_Prefetch2L2BOP_Total	2598326	2549775	2859392	2554979
E2_L2AReqSource_Prefetch2L2BOP_Miss	2310059	2264926	2540936	2277265
E2_L2AReqSource_Prefetch2L2SMS_Total	496600	428649	477848	446650
E2_L2AReqSource_Prefetch2L2SMS_Miss	304877	262349	350090	286109
E2_L2AReqSource_Prefetch2L2Stream_Total	0	0	0	0
E2_L2AReqSource_Prefetch2L2Stream_Miss	0	0	0	0
E2_L2AReqSource_Prefetch2L2Stride_Total	449	517	562	308
E2_L2AReqSource_Prefetch2L2Stride_Miss	449	517	561	308
E2_L2AReqSource_Prefetch2L2TP_Total	439394	0	4422765	0
E2_L2AReqSource_Prefetch2L2TP_Miss	332895	0	3432214	0
L2总请求数	13480806	12918666	17813255	12995570
L2总命中率	67.35%	69.59%	54.33%	69.53%
IPC	0.458536	0.451472	0.397984	0.441047

L3总请求数	plru-错tp	plru-关tp	l3tubins	l3tubins-notp
E2_L3AReqSource_CPULoadData_Total	1189853	1162951	1367474	1155277
E2_L3AReqSource_CPULoadData_Miss	94196	95391	199739	118712
E2_L3AReqSource_CPUStoreData_Total	3	3	2	3
E2_L3AReqSource_CPUStoreData_Miss	3	3	1	3
E2_L3AReqSource_L1InstPrefetch_Total	244	0	9	0
E2_L3AReqSource_L1InstPrefetch_Miss	133	0	9	0
E2_L3AReqSource_PTW_Total	262998	237884	444735	240189
E2_L3AReqSource_PTW_Miss	3242	3319	3765	3254
E2_L3AReqSource_Prefetch2L2BOP_Total	2310058	2264926	2540936	2277266
E2_L3AReqSource_Prefetch2L2BOP_Miss	1330153	1335411	1356660	1373126
E2_L3AReqSource_Prefetch2L2SMS_Total	304877	262349	350090	286109
E2_L3AReqSource_Prefetch2L2SMS_Miss	143596	131834	181036	150030
E2_L3AReqSource_Prefetch2L2Stream_Total	0	0	0	0
E2_L3AReqSource_Prefetch2L2Stream_Miss	0	0	0	0
E2_L3AReqSource_Prefetch2L2Stride_Total	449	517	561	308
E2_L3AReqSource_Prefetch2L2Stride_Miss	449	517	561	308
E2_L3AReqSource_Prefetch2L2TP_Total	332895	0	3432213	0
E2_L3AReqSource_Prefetch2L2TP_Miss	4849	0	275873	0
L3总请求数	4401377	3928630	8136020	3959152
L3总命中率	64.18%	60.13%	75.20%	58.44%

图 3. 实验结果示意

6 总结与展望

在本项目中，我们成功复现并评估了论文《Bypass and Insertion Algorithms for Exclusive Last-level Caches》中提出的 LLC 替换算法。通过在 OpenXiangShan 开源项目的 huancun 模块中实现这些算法，我们验证了它们在提升缓存性能方面的有效性。实验结果表明，与现有策略相比，我们的算法在单线程和多线程工作负载下均能显著提升处理器缓存的 IPC 性能，平均提升分别为 3.4

展望未来，我们计划进一步优化和扩展这些算法。首先，我们打算将 Bypass 功能从 L3 层次扩展到 L2 模块，以实现更全面的缓存管理。其次，我们计划探索 TC-UC-RANK 算法，以替代现有的 TC-UC-AGE 算法，预期这将带来额外的性能提升。此外，我们也将考虑算法在不同工作负载和硬件配置下的适应性和可扩展性，以确保它们能够在多样化的应用场景中发挥最大效用。

总体而言，我们的工作不仅在理论上具有创新性，而且在实际的计算机系统设计中具有重要的应用价值。我们期待通过持续的研究和开发，进一步提升缓存系统的效率和性能，为未来的计算架构设计提供有价值的参考和指导。

参考文献

[1] Jayesh Gaur, Mainak Chaudhuri, and Sreenivas Subramoney. Bypass and insertion algorithms for exclusive last-level caches. pages 81–92, 2011.

[2] S. Kaxiras, M. Martin, and C. Reinhardt. Cache decay: Exploiting generational behavior to extend cache lifetimes. *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, 2006.

[3] H. Kim, D. Chou, and Y. N. Patt. The multi-level cache hierarchy: an architectural survey. *ACM Computing Surveys (CSUR)*, 35(3):299–328, 2003.