Cache预取和替换策略联合优化

刘荟文1)

计算机学院, 计算机科学与技术, 2114019

摘 要 影响Cache性能的因素有很多，其中预取（prefetching）和替换（replacement）算法是最重要的两个因素。本实验综合考虑cache预取和替换之间的关联性，研究设计联合优化算法pref\_srrip，进一步提升应用程序的性能，所得结果显示使用相同预取策略的情况下，pref\_srrip较其他替换策略（SRRIP、SHiP）的Cumulative IPC可提升 2%~9%。

关键词 cache；预取策略；替换策略；next\_line；IP\_stride；SSP；SHiP；SRRIP

# 1 研究动机

Cache在计算机体系结构中有着举足轻重的作用，学术界对于Cache的研究一直是热点。影响Cache性能的因素有很多，其中预取（prefetching）和替换（replacement）算法是最重要的两个因素。尽管研究 Cache预取和替换算法的工作有很多，但是已有工作存在两个问题：

（1）仅考虑 Cache 预取或者仅考虑 Cache 替换，没有考虑两者之间的关联性；

（2）通常只考虑某个层级（例如，L2 或者 L3）的预取或者替换策略，没有考虑多个层级预取和替换策略之间的关联关系。本实验针对上述问题，研究多层级预取和替换联合优化策略，目标是最大化应用程序的性能。

# 2 实验要求

1）本实验使用的champsim模拟器环境提供了在L1D、L2C和LLC多级cache上的next\_line prefetcher、IP\_stride(Instruction Pointer-based/Program counter-based Stride) prefetcher、SSP（Signature Path Prefetcher）等预取方法与SRRIP(Static Re-Reference Interval Predictor)和SHiP(Signature-based Hit Predictor)等LLC替换策略，实验测试组合后的性能表现，寻找到比较优的算法组合。

2）考虑不同cache层级之间的预取策略、LLC cache的预取策略和替换策略之间的关联关系，以已有的预取和替换策略算法为基础，进行修改，设计联合优化算法，进一步提高策略组合的性能。

# 3 经典算法描述

## **3.1 实验环境**

实验测试环境运行在Ubuntu22.04 64位系统下，基于 ChampSim 模拟器设计并实现Cache的预取和替换算法，基于trace462.libquantum-714B.champsimtrace.xz、482.sphinx3-1100B.champsimtrace.xz验证算法的有效性，预热指令数设置为100M，模拟运行指令数设置为100M，以Cumulative IPC(instructions per cycle)作为评估指标。

## **3.2 预取算法**

### 3.2.1 next\_line

当发生缓存缺失时，next\_line 预取策略会立即预取下一行的数据到cache中。非常简单且易于实现，适用于顺序访问，但在非顺序访问模式下，例如随机访问或跳跃式访问，next\_line预取策略可能会导致不必要的预取，浪费缓存和内存带宽。

### 3.2.2 IP\_stride

IP Stride预取策略监视程序计数器（Instruction Pointer）的变化，计算指令地址的步长，如果检测到步长的规律性，据此规律预测未来的指令地址并预取数据到缓存中。在程序按照一定步长连续执行的情况下，IP Stride 预取策略能够有效地预测未来的指令地址。对于一些不具备规律性的访问模式，或者在程序流程中发生跳转等情况时，IP Stride 预取策略可能无法提供良好的性能。

### 3.2.3 SPP

SPP 引入了预取周期的概念，在每个预取周期结束时，触发预取操作，预取下一个顺序地址的数据到缓存中。适用于顺序访存模式，对于随机或不规律的访存模式则无法提供良好的预取效果。

## **3.3 替换策略**

### 3.3.1 SRRIP

SRRIP（静态再参考间隔预测策略）为了阻止再次调用间隔很久的cache块在cache长期存在、污染cache，使用RRPV维护cache的状态。RRPV值表示预测某一cache块被重新调用的次序，RRPV值越小表明该cache块越可能被调用，反之则表示该cache块的下次调用间隔将会很长。初始化时设置所有块的RRPV为maxRRPV - 1，替换时选择第一个RRPV值为maxRRPV - 1的内存块进行替换，并将其值设置为0；没有找到这个内存块则将所有内存块的RRPV值加1，再重复寻找。SRRIP策略能够自动适应工作集中应用程序的大小，并事实性地降低cache缺失率。

### 3.3.2 SHiP

SHiP是基于签名学习的缓存替换策略，结合采样器（Sampler）和SHCT（Sampler History Counter Table）以学习签名的重引用行为,据此进行缓存替换决策。SHCT表项值代表着签名的重引用行为。如若为0，说明这个缓存行大概率不会被使用；若值为正，说明相应的签名很可能被命中。发生命中时，增加SHCT表中该缓存行对应的签名的值；当某缓存行要被替换出去且在插入后没有被重引用过，减少SHCT中对应的值。在算法执行过程中，如果发生cache缺失，通过要插入的缓存行的签名在SHCP表中找到相应的值，替换策略可以选择是否要替换该行。相关研究表明，在LLC中使用SHiP策略对缓存的性能有很大的提升。

# 4 实验结果

根据提供的多种预取策略和替换策略进行分组，所有分组测试结果见文末附录表格1。

收集并计算所得数据，在两个数据集上总体表现相对较有的算法组合有：

1. L1D Prefetcher: Next\_line

L2C Prefetcher: IP\_stride

LLC Prefetcher: Next\_line

LLC Replacement: SHiP

1. L1D Prefetcher: No

L2C Prefetcher: SPP

LLC Prefetcher: Next\_line

LLC Replacement: SHiP

1. L1D Prefetcher: Next\_line

L2C Prefetcher: SPP

LLC Prefetcher: IP\_stride

LLC Replacement: SHiP

1. L1D Prefetcher: Next\_line

L2C Prefetcher: IP\_stride

LLC Prefetcher: Next\_line

LLC Replacement: SRRIP

1. L1D Prefetcher: Next\_line

L2C Prefetcher: SPP

LLC Prefetcher: IP\_stride

LLC Replacement: SRRIP

分析实验结果，发现单一的预取算法与替换策略相组合时并不能很好地发挥缓存的性能优势，选择在不同层级的cache上使用不同的预取策略，缓存性能则能得到较为显著的提升。同时对比分析可得，SHiP、SRRIP等替换策略的对性能的影响与Next\_line、IP\_stride、SPP等预取策略产生的影响相互独立，与本次实验背景所讨论的“现有的工作缺乏对cache预取和替换策略之间关联性的探讨”情形一致。

# 5 联合优化算法设计

## **5.1 设计思路**

使用Next\_line预取策略时，预取器总预取正在使用的数据块的下一内存块，在极端某些情况下，会使得cache中存在大量并不会被用到的无效块缓存，影响性能表现。考虑设计修改替换策略，使替换发生时，优先选择将“已预取进入cache、但从未被使用的“数据块换出，以减少“将来有更大概率被访存的块被换出cache”的情况发生。最终选择基于SRRIP算法进行修改，设计出与预取策略相关联的pref\_SRRIP替换策略。

1. 定义由LLC\_pref和LLC\_repl共享的二维数组my\_priority，用以记录cache中每一个数据块的预取优先级，初始化为0。
2. 每次预取成功后，将cache放入位置对应的my\_priority数组值修改为1。
3. 每次hit命中后，将命中的cache块位置的对应my\_priority数组值修改为0。
4. SRRIP在遍历cache中RRPV值时，选择RRPV值为maxRRPV - 1的候选块中对应my\_priority数组值为1的块换出；若不存在满足条件的数据块，直接选候选块中的第一个换出。

当一个已被预取的内存块在接下来的数次内存访问和替换中都没有命中，就可以认为这个内存块的预取对性能的提升“无效”。推测在未来相当长的一段时间内，其较拥有相同换出优先级（RRPV）但曾经被访问过的数据块、有更大概率未来不会被访问到，因此选择将其换出，以提升性能表现。

## **5.2 结果分析**

根据提供的预取策略和设计的替换策略进行分组，并与先前使用经典算法的分组所得数据进行比较分析，所有分组测试结果可见文末附录表格2。

观察所得数据可知，当预取策略设置相同时，考虑了预取策略关联关系的Pref\_SRRIP在不同traces上的表现（Cumulative IP）较SHiP和SRRIP均有提升，提升范围在 2%~ 9%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **L1-pref** | **L2-pref** | **L3-pref** | **L3-repl** | **IPC-462** | **IPC-482** | **IPC-AVG** |
| no | no | no | ship | 0.50861 | 0.69799 | 0.6033 |
| next\_line | no | no | ship | 0.55513 | 1.06186 | 0.808495 |
| no | next\_line | no | ship | 0.54739 | 1.04346 | 0.795425 |
| no | no | next\_line | ship | 0.51445 | 0.85659 | 0.68552 |
| no | ip\_stride | no | ship | 0.70383 | 1.12101 | 0.91242 |
| no | no | ip\_stride | ship | 0.66159 | 0.88023 | 0.77091 |
| next\_line | next\_line | next\_line | ship | 0.6444 | 1.1546 | 0.8995 |
| next\_line | ip\_stride | next\_line | ship | 0.69077 | 1.22446 | 0.957615 |
| next\_line | next\_line | ip\_stride | ship | 0.67272 | 1.20168 | 0.9372 |
| no | spp\_dev | no | ship | 0.82631 | 1.03685 | 0.93158 |
| no | spp\_dev | next\_line | ship | 0.83288 | 1.09431 | 0.963595 |
| no | spp\_dev | ip\_stride | ship | 0.83411 | 1.05092 | 0.942515 |
| next\_line | spp\_dev | no | ship | 0.60517 | 1.20538 | 0.905275 |
| next\_line | spp\_dev | next\_line | ship | 0.64727 | 1.03625 | 0.84176 |
| next\_line | spp\_dev | ip\_stride | ship | 0.69659 | 1.23443 | 0.96551 |
| no | no | no | srrip | 0.49873 | 0.60392 | 0.551325 |
| next\_line | no | no | srrip | 0.54333 | 0.97908 | 0.761205 |
| no | next\_line | no | srrip | 0.53716 | 0.96036 | 0.74876 |
| no | no | next\_line | srrip | 0.50901 | 0.87178 | 0.690395 |
| no | ip\_stride | no | srrip | 0.69243 | 1.02737 | 0.8599 |
| no | no | ip\_stride | srrip | 0.65005 | 0.89159 | 0.77082 |
| next\_line | next\_line | next\_line | srrip | 0.63579 | 1.17974 | 0.907765 |
| next\_line | ip\_stride | next\_line | srrip | 0.67658 | 1.23575 | 0.956165 |
| next\_line | next\_line | ip\_stride | srrip | 0.66473 | 1.213 | 0.938865 |
| no | spp\_dev | no | srrip | 0.83533 | 1.0452 | 0.940265 |
| next\_line | spp\_dev | next\_line | srrip | 0.64839 | 1.2374 | 0.942895 |
| next\_line | spp\_dev | ip\_stride | srrip | 0.69664 | 1.23966 | 0.96815 |
| next\_line | spp\_dev | no | srrip | 0.60444 | 1.2137 | 0.90907 |

表 1

备注：红字分组为性能表现相对较好的分组

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **L1-pref** | **L2-pref** | **L3-pref** | **L3-repl** | **IPC-462** | **IPC-482** | **IPC-AVG** |
| no | no | next\_line | pref\_srrip | 0.53767 | 0.92763 | 0.73265 |
| no | no | next\_line | srrip | 0.50901 | 0.87178 | 0.690395 |
| no | no | next\_line | ship | 0.51445 | 0.85659 | 0.68552 |
| no | spp\_dev | next\_line | pref\_srrip | 0.85682 | 1.11264 | 0.98473 |
| no | spp\_dev | next\_line | srrip | 0.70683 | 1.09692 | 0.901875 |
| no | spp\_dev | next\_line | ship | 0.83288 | 1.09431 | 0.963595 |
| no | ip\_stride | next\_line | pref\_srrip | 0.76864 | 1.11264 | 0.94064 |
| no | ip\_stride | next\_line | srrip | 0.71034 | 1.12937 | 0.919855 |
| no | ip\_stride | next\_line | ship | 0.72315 | 1.12087 | 0.92201 |
| next\_line | ip\_stride | next\_line | pref\_srrip | 0.73542 | 1.22828 | 0.98185 |
| next\_line | ip\_stride | next\_line | srrip | 0.67658 | 1.23575 | 0.956165 |
| next\_line | ip\_stride | next\_line | ship | 0.69077 | 1.22446 | 0.957615 |

表 2