计算机体系结构课程实验报告

Cache预取和替换

学号：2112193　姓名：王可一 专业：物联网工程 学院：网络空间安全学院

Github:0BZ0/Champism (github.com)

一、研究动机

CPU的计算速度极快，然而内存的访问速度相对较慢。这导致CPU每次从内存读取数据时都会产生大量等待时间，从而降低整体性能。为了解决这一问题，引入了多级缓存的概念，以在CPU和内存之间建立数据缓存层。最常用的数据被暂时保存在靠近CPU的高速缓存（Cache）中，以便CPU能够快速访问。不同级别的缓存具有不同的容量和访问速度，通常来说，L1缓存最小、速度最快，L2缓存次之，L3缓存最大但速度相对较慢。此外，L1和L2是CPU私有的，而L3是所有CPU共享的。多级缓存的设计旨在实现更高的命中率，即CPU能够更频繁地从高速缓存中获取所需数据，从而减少对内存的访问次数，提高整体性能。

缓存的设计主要基于局部性原理，以提升计算机的整体性能。由于缓存的性能仅次于寄存器，而CPU与内存之间的速度差异显著，主要体现在存取速度的数量级差异上。因此，通过尽可能多地让CPU访问缓存，同时减少CPU直接访问主存的次数，可以显著提高计算机的性能。

设置正确的缓存预取和替换策略对于提升CPU访问内存速度至关重要。通过合理的缓存管理，计算机的效率得到显著提升，整体性能也得到明显改善。因此，研究和优化缓存系统的策略成为提高计算机性能的关键方向。

二、相关工作

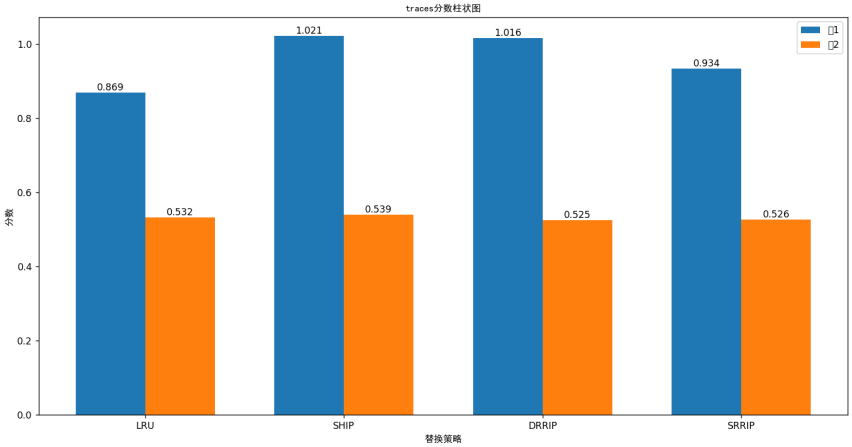
研究多层级预取和替换联合优化策略，目标是最大化应用程序的性能。

通过git克隆下载champsim内容可知，所给LLC替换策略为lru、ship、drrip、srrip；LLC的预取策略为next_line、ip_stride；L1D的预取策略为next_line；L2C的预取策略为ip_stride、next_line。由此可对当前所给的已知策略进行排列组合，并利用所给traces对所得组合进行相关数据测量。

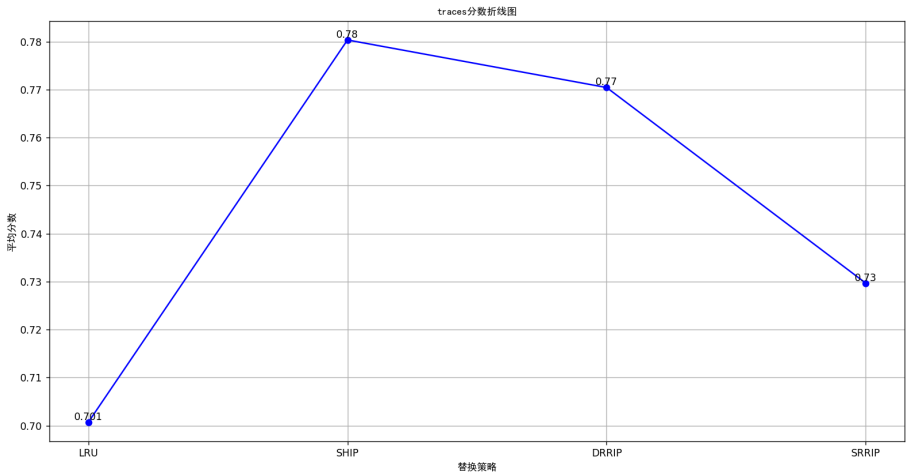
1、LLC替换策略分析

首先对LLC的替换策略进行分析，共有4种组合为：LRU、SHIP、DRRIP、SRRIP，利用相同预取策略进行比较分析。

对于所有组合更改build_champsim.sh文件后，可直接使用语句./build_champsim.sh next_line 进行构建相关执行文件，后通过run_champsim.sh文件进行IPC比较并通过score.py脚本进行分数计算进行比较。



traces整体分数分析图



traces平均分数分析图

IPC数据相差不大且没有分数得出明显，直接通过分数分析图可知，LLC的替换策略中SHIP远胜于其他策略，故后续的替换策略均为SHIP，进行后续预取策略选择。

2、预取策略分析

（1）L1D预取策略分析

在替换策略为SHIP的基础上进行预取策略的更改，先对L1D预取进行分析，分为两种情况：no 和 next_line。

traces分数/预取策略	no	next_line
482.sphinx3-1100B	0.685370	1.049410
462.libquantum-714B	0.505410	0.546040
平均	0.595390	0.797725

由表格可明显得出L1D采用next_line的预取策略效果更好，后续分析中对L1D采用next_line。

（2）L2C预取策略分析

L2C的预取策略为三种情况：no、next_line、ip_stride。

traces分数/预取策略	no	next_line	ip_stride
482.sphinx3-1100B	1.049410	1.122300	1.217410

traces分数/预取策略	no	next_line	ip_stride
462.libquantum-714B	0.546040	0.560870	0.657060
平均	0.797725	0.841585	0.937235

由数据可得，L2C采用ip_stride的预取结果更好。

（3）LLC预取策略分析

LLC的预取策略和L2C的相同，为三种情况：no、next_line、ip_stride。在LLC预取策略为ip_stride时，build_champsim报错，生成失败，后发现是因参数重复定义问题进行修改，后对其进行静态处理，成功建立。但在后续run_champsim中，报错，未发现可修改处，故未采用该组合。

traces分数/预取策略	no	next_line	ip_stride
482.sphinx3-1100B	1.217410	1.176450	——
462.libquantum-714B	0.657060	0.681300	——
平均	0.937235	0.928875	——

由此可得未采用预取策略可有更好的结果。

三、方法描述

因为上诉LLC预取策略实现结果甚至不如未实现，故对LLC预取策略进行学习和设计。

1、Spatial Prefetcher

借助已知算法完成，所给只有next_line算法能够成功运行，故以next_line算法为运行基础，借助spatial空间预取分析，实现二者结合进行数据预测。通过循环来找出可能被访问的数据块。

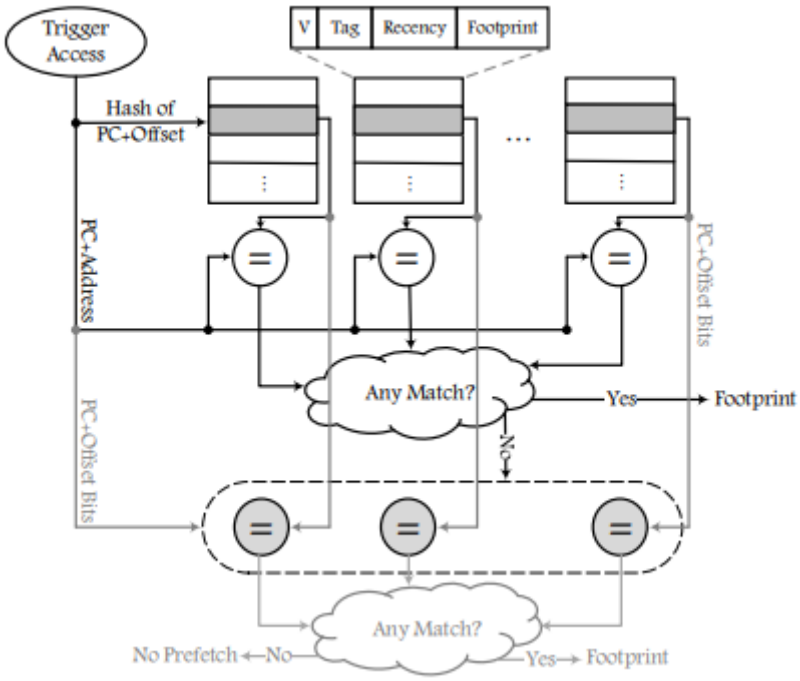
```
uint32_t CACHE::llc_prefetcher_operate(uint64_t addr, uint64_t ip, uint8_t cache_hit, uint8_t type, uint32_t metadata_in)
{
    // 检查是否发生缓存命中
    if (!cache_hit) {
        // 空间预取：预取下一个可能被访问的数据块
        for (int i = 1; i <= PREFETCH_DISTANCE / BLOCK_SIZE; ++i) {
            uint64_t pf_addr = (addr + i * BLOCK_SIZE) & ~(BLOCK_SIZE - 1);
            prefetch_line(ip, addr, pf_addr, FILL_LLC, 0);
        }
    }

    // 返回原始的 metadata
    return metadata_in;
}
```

spatial预取主要算法

2、Bingo Spatial Data Prefetcher

通过查找学习，借助Bingo算法完成实验。该算法记录了PC+address和PC+offset两种event；因为最长event中包含了最短event的信息，History table的实现用最短event （hash后）作为index，最长event作为tag；当要写入时，根据PC+offset确认set，再根据相应的替换算法来确认way；命中时PC+address的优先级更高，所以更准；当命中最短PC+offset但没命中PC+address时，有可能会出现multi-hit的情况，Bingo会根据每个hit的entry进行prefetch（这个实现的性能最好），以此实现更好的性能。



Bingo主要运作流程

四、实验结果

1、Spatial Prefetcher

根据以上收集信息，可得L1D最优选择为next_line，L2C最优选择为ip_stride，LLC的替换策略为SHIP，由上诉方法得出LLC的预取策略为spatial预取——空间预取。

```
ChampSim is successfully built
Branch Predictor: perceptron
L1D Prefetcher: next_line
L2C Prefetcher: ip_stride
LLC Prefetcher: spatial
LLC Replacement: ship
Cores: 1
Binary: bin/perceptron-next_line-ip_stride-spatial-ship-1core
```

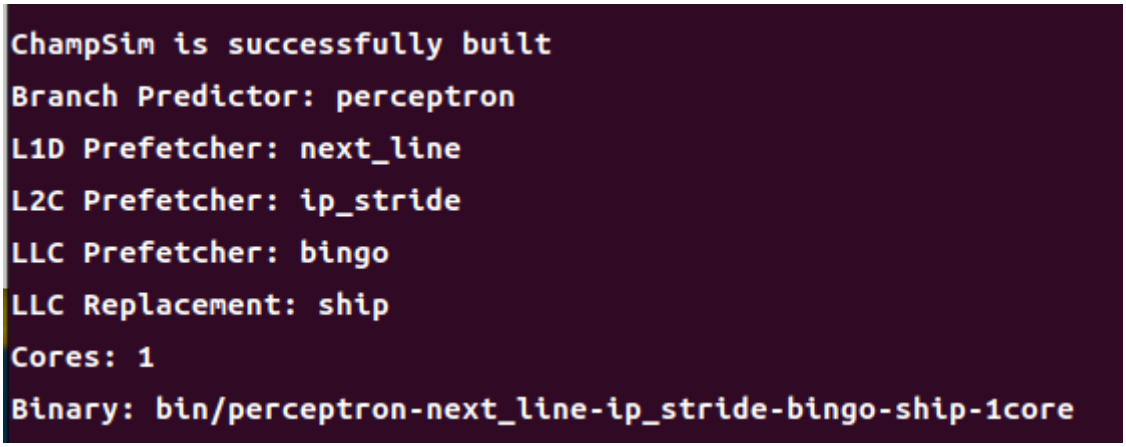
champsim成功生成

traces分数/预取距离	256	512	1024	2048	4096
482.sphinx3-1100B	1.201290	1.204350	1.196680	1.175960	1.175100
462.libquantum-714B	0.783830	0.798510	0.817050	0.858040	0.857660
平均	0.992560	1.001430	1.006865	1.017000	1.016380

由此可得，在预取距离为2048时，此时预取效果最好。

2、Bingo Spatial Data Prefetcher

同上诉策略，将LLC预取策略改为Bingo策略。此外，Bingo策略也有部分优化，下面对全部预取策略进行分析。其中Bingo_POA为PC Offset > Address；Bingo_APO为Address > PC Offset。



Bingo成功生成

traces分数/Bingo策略	Bingo	Bingo_Plus	Bingo_thresh100	Bingo_POA	Bingo_APO
482.sphinx3-1100B	1.213120	1.210990	1.211050	1.214710	1.211860
462.libquantum-714B	0.656840	0.657000	0.657270	0.656500	0.656940
平均	0.934980	0.933995	0.934160	0.935605	0.934400

故应用Bingo策略，实现效果比next_line好，但并未达到预期。

参考文献

【1】Pejman Lotfi-Kamran,Hamid Sarbazi-Azad.Chapter Two - Spatial prefetching.Advances in Computers.Volume 125, 2022, Pages 19-29。

【2】Mohammad Bakhshalipour,Mehran Shakerinava, Pejman Lotfi-Kamran,Hamid Sarbazi-Azad.Bingo Spatial Data Prefetcher.2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)