

Cache 模拟器性能优化实验报告

姓名：张煌昭
学号：1400017707
院系：元培学院
邮箱：zhang_hz@pku.edu.cn
手机：17888838127

一. 实验要求

本次 Lab 在上一次 Lab 3.1，即 Cache 性能模拟器的基础上，对 Cache 的管理策略进行各种优化尝试，并试图寻找最佳的优化组合方式。

1. Cache 配置

采用两级 Cache，仿照 Core i7-900 Desktop Processor 处理器中的 L1 和 L2 对 Cache 进行配置。默认参数如下表所示，其中 Hit latency 需要使用存储访问模型计算得到。

表. 两级 Cache 的默认配置参数

Level	Capacity	Associativity	Line Size	WriteUp	Bus Latency	Hit Latency
L1	32 KB	8 Ways	64 Byte	WB + WA	0 Cycles	?
L2	256 KB	8 Ways	64 Byte	WB + WA	6 Cycles	?
Mem	∞	-	-	-	100 Cycles	

'WB' means 'write back'. 'WA' means 'write allocate'. 'Cycles' means 'CPU Cycles'. '?' means unknown yet.

For memory, bus latency + hit latency = 100 CPU Cycles

其中 Bus latency 是指上层请求传至本层与本层返回数据给上层的时间延迟之和；Hit latency 是指本层查找数据所需要的时间，无论 Hit 或 Miss，只要没有采用 Bypass 策略，该延时都一定存在。默认使用 LRU 替换策略。

假设主存无限大，并包含了所有数据，令 CPU 和主存频率为 2GHz。

2. 存储模型

使用 cacti65 存储模型计算默认配置下 L1 和 L2 的 Hit latency。通过修改.cfg 文件完成配置，其中固定工艺节点为 0.032um 工艺，Cache 类型固定为“cache”。

得到 Hit latency 后计算对应的 CPU cycle，向上取整作为默认配置。

3. 优化策略

根据平均访存延迟公式，如下式

$$AMAT = \text{Hit Latency} + \text{Miss Rate} \times \text{Miss Penalty}$$

对上述的默认配置的 Cache 进行性能优化，进行优化的管理策略有：a) 使用其他替换算法替代 LRU；b) 添加 Prefetch 策略；c) 使用 Bypass 策略。

要求以上三种优化策略，每种至少应用于其中一级 Cache；需要估算添加优化策略后的额外的存储开销；预取数据不能超过 4 个 Cache line（包括目标数据本身）。

二．基准线实验

这一部分中，将使用默认参数和配置进行实验，作为之后的优化的基准线。

1. 利用 cacti65 存储模型计算 Hit latency

使用本次实验提供的 cacti65 存储模型进行计算。首先安装依赖库 gcc-multilib 和 g++-multilib；之后 make 编译存储模型；修改 cache.cfg 配置文件为默认配置中要求的 L1 和 L2 参数后使用命令“./cacti -infile cache.cfg”运行。

存储模型的运行结果如下图，左图为 L1 默认配置下的计算结果，右图为 L2 默认配置下的计算结果。

```
----- CACTI version 6.5, UniformCache Access Commodity DRAM Model -----
Cache Parameters:
Total cache size (bytes): 32768
Number of banks: 1
Associativity: 8
Block size (bytes): 64
Read/write Ports: 1
Read ports: 0
Write ports: 0
Technology size (nm): 32

Access time (ns): 1.47944
Cycle time (ns): 2.1629
Precharge Delay (ns): 0
Activate Energy (nJ): 0.0171586
Read Energy (nJ): 0.0911486
Write Energy (nJ): 0.0608561
Precharge Energy (nJ): 0.0392208
Leakage Power Closed Page (mW): 0.296633
Leakage Power Open Page (mW): 0.891618
Leakage Power I/O (mW): 1.44413
Refresh power (mW): 4.07253e-05
Cache height x width (mm): 0.257488 x 0.24624

Best Ndwl : 8
Best Ndbl : 2
Best Nspd : 0.5
Best Ndcn : 1
Best Ndsan L1 : 16
Best Ndsan L2 : 1

Best Ntwl : 2
Best Ntbl : 2
Best Ntspd : 0.5
Best Ntcn : 1
Best Ntsan L1 : 1
Best Ntsan L2 : 1
Data array, H-tree wire type: Delay optimized global wires
Tag array, H-tree wire type: Global wires with 38% delay penalty
```

```
----- CACTI version 6.5, UniformCache Access Commodity DRAM Model -----
Cache Parameters:
Total cache size (bytes): 262144
Number of banks: 1
Associativity: 8
Block size (bytes): 64
Read/write Ports: 1
Read ports: 0
Write ports: 0
Technology size (nm): 32

Access time (ns): 1.9206
Cycle time (ns): 3.40728
Precharge Delay (ns): 0
Activate Energy (nJ): 0.00955506
Read Energy (nJ): 0.226472
Write Energy (nJ): 0.21833
Precharge Energy (nJ): 0.0554629
Leakage Power Closed Page (mW): 0.669746
Leakage Power Open Page (mW): 0.967238
Leakage Power I/O (mW): 3.7558
Refresh power (mW): 0.000954863
Cache height x width (mm): 0.349703 x 0.58757

Best Ndwl : 32
Best Ndbl : 2
Best Nspd : 1
Best Ndcn : 1
Best Ndsan L1 : 8
Best Ndsan L2 : 1

Best Ntwl : 1
Best Ntbl : 2
Best Ntspd : 1
Best Ntcn : 2
Best Ntsan L1 : 1
Best Ntsan L2 : 1
Data array, H-tree wire type: Delay optimized global wires
Tag array, H-tree wire type: Global wires with 38% delay penalty
```

根据 cacti65 的运行结果，得到：默认配置下，32nm 工艺节点下，L1 Cache 的 Hit latency 为 1.47944ns，约为 3 Cycle；L2 Cache 的 Hit latency 为 1.9206ns，约为 4 Cycle。最终，Cache 的全部配置参数如下表所示。

表. 两级 Cache 的默认配置参数

Level	Capacity	Associativity	Line Size	WriteUp	Bus Latency	Hit Latency
L1	32 KB	8 Ways	64 Byte	WB + WA	0 Cycles	3 Cycles
L2	256 KB	8 Ways	64 Byte	WB + WA	6 Cycles	4 Cycles
Mem	∞	-	-	-	100 Cycles	

'WB' means 'write back'. 'WA' means 'write allocate'. 'Cycles' means 'CPU Cycles'. '?' means unknown yet.

For memory, bus latency + hit latency = 100 CPU Cycles

2. AMAT 公式推导

AMAT 的定义公式如第一部分中所示，其中 Hit latency 可以认为是已知量，也可以认为需要进行测量，本次实验中为了简化，认为 Hit latency 为常量且为默认配置参数中的 Bus latency 与 Hit latency 之和；Miss rate 需要测量，不同的 trace 会导致不同的 Miss rate；Miss penalty 即为访问下层 Cache 的平均用时。因此对本次实验的两级 Cache 的 AMAT 的计算公

式的推导如下。其中 HL(L1), HL(L2), AT(Mem)均可以通过默认设置中的参数计算得到, MR(L1) 和 MR(L2)可以通过实验测量得到。

$$\begin{aligned}
 AMAT &= \text{Hit latency} + \text{Miss rate} \times \text{Miss penalty} \\
 &= HL + MR \times MP \\
 &= HL(L1) + MR(L1) \times AAT(L2), \\
 &\quad \text{in which AAT means 'Average Access Time'} \\
 AAT(L2) &= HL(L2) + MR(L2) \times AAT(Mem) \\
 &= HL(L2) + MR(L2) \times AT(Mem) \\
 &\quad \text{in which AT means 'Access Time'} \\
 \text{so, } AMAT &= HL(L1) + MR(L1) \times AAT(L2) \\
 &= HL(L1) + MR(L1) \times [HL(L2) + MR(L2) \times AT(Mem)]
 \end{aligned}$$

3. 默认配置的 Cache

按照上表默认配置设置 Cache 模拟器, 使用 LRU 替换算法, 不使用 Bypass 和 Prefetch 及其他任何的优化方法, 使用命令“./cache-sim -trace=./xxx.trace -iter=100”运行 Lab 3.1 提供的 1.trace 和 2.trace 以及本次 Lab 3.2 提供的 01-mcf-gem5-xcg.trace 和 02-stream-gem5-xaa.trace 共 4 个 trace 各测试 100 轮, 运行结果如下图。

<pre> icgic-VirtualBox:~/下载/Mem/cache/build-cache -sin-Desktop_Qt_5_9_2_GCC_64bit-Debug\$./cach e-sin -iter=100 -trace=01-mcf-gem5-xcg.trace L1-Cache config: Size = 32768 byte Set number = 64 Associativity = 8 Write back Write alloc Hit latency = 3 cycles Bus latency = 0 cycles Replace policy: LRU No bypass applied No prefetch applied L2-Cache config: Size = 262144 byte Set number = 512 Associativity = 8 Write back Write alloc Hit latency = 4 cycles Bus latency = 6 cycles Replace policy: LRU No bypass applied No prefetch applied AMAT = 12.261262 (cycles) Total L1 access count: 23261100 Total L1 access time: 362144960 cycles L1 AAT = 15.568694 cycles Total L1 miss count: 4647356 Miss rate = 19.97989% Total L1 fetch count: 4647356 Total L1 replace count: 4646844 Total L1 prefetch count: 0 Total L1 bypass count: 0 Total L2 access count: 5605316 Total L2 access time: 292361660 cycles L2 AAT = 52.157928 cycles Total L2 miss count: 2037800 Miss rate = 36.354774% Total L2 fetch count: 2037800 Total L2 replace count: 2033704 Total L2 prefetch count: 0 Total L2 bypass count: 0 Total Memory access count: 2363085 Total Memory access time: 236308500 cycles Mem AAT = 100.000000 cycles </pre>	<pre> icgic-VirtualBox:~/下载/Mem/cache/build-cache -sin-Desktop_Qt_5_9_2_GCC_64bit-Debug\$./cach e-sin -iter=100 -trace=02-stream-gem5-xaa.tr ace L1-Cache config: Size = 32768 byte Set number = 64 Associativity = 8 Write back Write alloc Hit latency = 3 cycles Bus latency = 0 cycles Replace policy: LRU No bypass applied No prefetch applied L2-Cache config: Size = 262144 byte Set number = 512 Associativity = 8 Write back Write alloc Hit latency = 4 cycles Bus latency = 6 cycles Replace policy: LRU No bypass applied No prefetch applied AMAT = 12.639495 (cycles) Total L1 access count: 16290000 Total L1 access time: 319680690 cycles L1 AAT = 19.624844 cycles Total L1 miss count: 1847400 Miss rate = 11.340780% Total L1 fetch count: 1847400 Total L1 replace count: 1846888 Total L1 prefetch count: 0 Total L1 bypass count: 0 Total L2 access count: 2463229 Total L2 access time: 270818690 cycles L2 AAT = 109.944588 cycles Total L2 miss count: 1847400 Miss rate = 74.99918% Total L2 fetch count: 1847400 Total L2 replace count: 1843304 Total L2 prefetch count: 0 Total L2 bypass count: 0 Total Memory access count: 2461864 Total Memory access time: 246186400 cycles Mem AAT = 100.000000 cycles </pre>	<pre> icgic-VirtualBox:~/下载/Mem/cache/build-cache -sin-Desktop_Qt_5_9_2_GCC_64bit-Debug\$./cach e-sin -iter=100 -trace=1.trace L1-Cache config: Size = 32768 byte Set number = 64 Associativity = 8 Write back Write alloc Hit latency = 3 cycles Bus latency = 0 cycles Replace policy: LRU No bypass applied No prefetch applied L2-Cache config: Size = 262144 byte Set number = 512 Associativity = 8 Write back Write alloc Hit latency = 4 cycles Bus latency = 6 cycles Replace policy: LRU No bypass applied No prefetch applied AMAT = 79.469162 (cycles) Total L1 access count: 1000000 Total L1 access time: 167041380 cycles L1 AAT = 167.041382 cycles Total L1 miss count: 996400 Miss rate = 99.640000% Total L1 fetch count: 996400 Total L1 replace count: 996160 Total L1 prefetch count: 0 Total L1 bypass count: 0 Total L2 access count: 1492300 Total L2 access time: 164041380 cycles L2 AAT = 109.918716 cycles Total L2 miss count: 996101 Miss rate = 66.745442% Total L2 fetch count: 996101 Total L2 replace count: 994181 Total L2 prefetch count: 0 Total L2 bypass count: 0 Total Memory access count: 1491175 Total Memory access time: 149117500 cycles Mem AAT = 100.000000 cycles </pre>	<pre> icgic-VirtualBox:~/下载/Mem/cache/build-cache -sin-Desktop_Qt_5_9_2_GCC_64bit-Debug\$./cach e-sin -iter=100 -trace=2.trace L1-Cache config: Size = 32768 byte Set number = 64 Associativity = 8 Write back Write alloc Hit latency = 3 cycles Bus latency = 0 cycles Replace policy: LRU No bypass applied No prefetch applied L2-Cache config: Size = 262144 byte Set number = 512 Associativity = 8 Write back Write alloc Hit latency = 4 cycles Bus latency = 6 cycles Replace policy: LRU No bypass applied No prefetch applied AMAT = 80.241913 (cycles) Total L1 access count: 3279300 Total L1 access time: 545040890 cycles L1 AAT = 166.450424 cycles Total L1 miss count: 3279300 Miss rate = 100.000000% Total L1 fetch count: 3279300 Total L1 replace count: 3279060 Total L1 prefetch count: 0 Total L1 bypass count: 0 Total L2 access count: 4874799 Total L2 access time: 636002990 cycles L2 AAT = 109.953865 cycles Total L2 miss count: 3277908 Miss rate = 67.241913% Total L2 fetch count: 3277908 Total L2 replace count: 3275988 Total L2 prefetch count: 0 Total L2 bypass count: 0 Total Memory access count: 4872550 Total Memory access time: 487255000 cycles Mem AAT = 100.000000 cycles </pre>
---	--	---	--

实验得到的结果如下表所示。其中 L1 和 L2 Cache 的 Hit latency 和 Bus latency 均来自默认配置, Mem 访问时间为 100。Miss Rate 和 AAT 计算公式如下, 所需要的值均为实验得到的结果。AMAT 的计算公式如下, 所需要的值均为已知值或实验得到的结果。

$$\text{Miss Rate} = \frac{\text{Miss Count}}{\text{Access Count}}$$

$$\text{AAT} = \frac{\text{Access Time}}{\text{Access Count}}$$

$$\begin{aligned}
 AMAT &= HL(L1) + MR(L1) \times (HL(L2) + MR(L2) \times AT(Mem)) \\
 &= \text{Bus Latency}(L1) + \text{Hit Latency}(L2) \\
 &\quad + MR(L1) \times (\text{Bus Latency}(L2) + \text{Hit Latency}(L2) + MR(L2) \times AT(Mem)) \\
 &= 0 + 3 + MR(L1) \times (6 + 4 + MR(L2) \times 100) \text{ (cycle)}
 \end{aligned}$$

表. 默认配置 Cache 在不同 trace 下的测试结果							
Trace	Level	Latency (cycles)		Miss Rate (percent)	AAT (cycles)	AMAT	
		Hit	Bus			(cycles)	(ns)
01-mcf-	L1	3	0	19.979	15.569	12.261	6.131
gem5-	L2	4	6	36.355	52.158		
xcg.trace	Mem	100		-	100.000		
02-stream-	L1	3	0	11.341	19.623	12.639	6.320
gem5-	L2	4	6	74.999	109.934		
xaa.trace	Mem	100		-	100.000		
1.trace	L1	3	0	99.640	167.070	79.483	39.742
	L2	4	6	66.759	109.932		
	Mem	100		-	100.000		
2.trace	L1	3	0	100.000	166.441	80.236	40.118
	L2	4	6	67.236	109.947		
	Mem	100		-	100.000		

We don't consider hit latency & bus latency of the memory, instead, we assume one access to memory costs 100 cycle, and as CPU frequency is 2GHz, 1 cycle = 0.5 ns. '-' means this item is nonsense or what we don't care.

根据上表, 01-mcf-gem5-xcg.trace 共运行 100 轮, L1 Cache Miss Rate = 19.979%, L2 Cache Miss Rate = 36.355%, 最终 AMAT = 12.261 cycles = 6.131ns ; 02-stream-gem5-xaa.trace 共运行 100 轮, L1 Cache Miss Rate = 11.341%, L2 Cache Miss Rate = 74.999%, 最终 AMAT = 12.639 cycles = 6.320ns。

三 . 优化策略的实现与选择

1. 替换策略

默认使用的是 LRU 替换策略, 该策略尽管在 Miss rate 上表现良好, 但需要很长时间扫描当前 Set 中的所有 Line 来寻找最近最少使用的 Line, 或更新所有 Line 的计数。因此考虑对其进行优化, 考虑使用最简单的随机替换或类似 LRU 但更为简单的 PLRU (伪 LRU 算法) 进行替换优化。

随机替换 (Random) 算法在空间上没有任何开销, 基本可以认为是最简单的 Cache 替换算法, 若其在 Miss rate 上的表现与 LRU 较为接近, 则可以考虑更换为随机替换来节省空间和降低 Cache 电路复杂度。

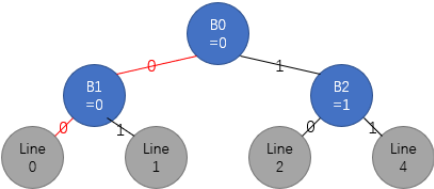
当使用随机替换时, 若在发生 Miss, 且该 Set 所有 Line 都满的情况下, 随机产生一个 [0, Line_num) 的下标作为替换的 Line 的下标。对程序添加 Random 替换选项, 并进行相应更改, 添加命令行随机种子参数的设置以方便 Debug。

伪 LRU (PLRU) 替换算法表现出与 LRU 相似的行为, 甚至某些情况下优于 LRU, 但 PLRU 的空间相对较小, 电路实现也更为简单, 一般情况下可以考虑使用 PLRU 替代 LRU。

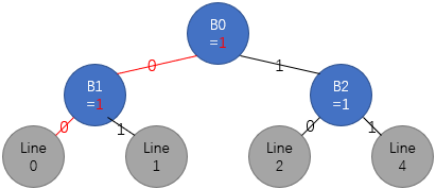
使用二叉树进行实现, 每个 Set 都需要一个二叉树, 该二叉树为满二叉树 (一般情况下也是完全二叉树), 所有非叶节点都是 1-bit 的存储单元, 所有叶节点都是该 Set 里的 Line

且一一对应，按照规则在二叉树上行走，最终到达的叶节点即为需要替换的 Line。二叉树中每个非叶节点存储 0 时向左，存储 1 时向右，最终从根走到叶节点的路径上的各个存储单元的值连起来即为该叶节点的二进制编号。

搜索待替换的 Line 的过程如下，示意图如下图：假设该 Set 中有 4 个 Line，Line 0 的编号为 00，Line 1 的编号为 01，Line 2 的编号为 10，Line 3 的编号为 11；根节点 B0 存储的是 0，因此向左走到达 B1，B1 存储的是 0，因此向左走到达 Line 0；最终行走的路径上各个存储单元的值连起来为 00，也就是 Line 0 的编号。



每次访问后更新的过程如下，示意图如下图：假设不变，在访问 Line 0 后，将路径上所有存储单元的值都变为相反的一边；B0 原本为 0，更新后为 1，B1 原本为 1，更新后为 1，B2 不在路径上，不进行更新；更新后，指示下一次将替换 Line 4。



用数组表示如上的一棵（完全）满二叉树，添加 PLRU 替换选项并对代码进行上述的更改。

进行实验，使用提供的测试 Random 替换（L1 和 L2 均使用 Random 替换，随机种子未固定）和 PLRU 替换（L1 和 L2 均使用 PLRU 替换）测试 100 轮，Miss rate 和 AMAT 的结果如下表。

表. 不同替换算法的 Miss Rate 对比

Trace	Level	Miss Rate (percent)			AMAT (cycles)		
		LRU	Random	PLRU	LRU	Random	PLRU
01-mcf-gem5-xcg.trace	L1	19.979	20.414	20.007	12.261	11.737	12.096
	L2	36.355	32.797	35.463			
02-stream-gem5-xaa.trace	L1	11.341	11.348	11.341	12.639	12.746	12.640
	L2	74.999	75.884	75.003			
1.trace	L1	99.640	99.649	99.640	79.483	81.775	79.483
	L2	66.759	69.053	66.759			
2.trace	L1	100.000	100.000	100.000	80.236	83.296	80.236
	L2	67.236	70.296	67.236			

分析上表，发现 Random 替换在局部性很好时，替换次数较少，有可能在概率上使得 Miss rate 降低，尽管 Random 替换在统计意义上可以利用局部性原理，但由于其随机性和不稳定性，使得 Random 替换并不能作为良好的替换算法。PLRU 替换是稳定的替换算法，其结果基本与 LRU 算法相同，在某些场景下（例如 01-mcf-gem5-xcg.trace 中）PLRU 替换的 Miss rate 相比 LRU 替换会有小幅的降低。

同时，对复杂度进行分析。LRU 替换在本模拟器中需要对各个 Cache Line 进行计数，在决定被替换的 Line 时也需要遍历整个 Set 中各个 Line 的计数器，因而其复杂度为 $O(n)$ 。Random 替换需要一个随机数产生部件，而与 Cache 的配置无关，因而其复杂度可以认为是 $O(1)$ ，但需要说明的是，真随机的产生十分复杂，因此这个常数复杂度的常数有可能非常大。PLRU 替换通过一个二叉树结构进行查找，复杂度为 $O(\log n)$ ，同时由于 PLRU 替换中每个 Set 仅仅需要 $n-1$ 个 bit 即可完成被替换 Line 的指示工作，其空间和硬件实现也优于 LRU。

综上，选用 PLRU 替换算法替代 LRU 替换算法。

2. Bypassing 策略

参考 Sparsh Mittal. JLPEA 2016[1]的综述，进行 Bypassing 和 Prefetch 的设计和实验。

由于并不是所有的内存访问都有必要使得 Cache 中的 Line 发生替换和更新，比如在 Cache 装满有用的 Line 之后，发生了一次很遥远的内存访问，并且该地址在之后不会再出现，这样的数据装入 Cache 中只会引起“污染”使得 Cache 的表现变差。因此没有必要在这种情况下替换本层 Cache，而是直接从下层 Cache 读出数据返回给上层 Cache，即 Bypass 本层 Cache。

MCT 预测 Miss 类型决定 Bypass，参考 Collins et al. 1999. [2]的论文，使用 MCT 对 Miss 类型进行预测，并根据 Miss 的类型决定是否 Bypass。

Miss 共有 3 类：冷启动时发生的 Compulsory miss，由于 Associativity 不足引起的 Conflict miss，和 Cache 容量不足引起的 Capacity miss。其中 Compulsory miss 几乎不可避免，Conflict miss 可以通过增大 Associativity 避免，Capacity miss 无法通过增大 Associativity 避免，只能通过增大 Cache size 避免。

MCT (Miss Classification Table) 是一个记录了历史被替换的 Line 的表格，其每一行对应一个 Set，每行中有可以记录若干个 Tag，按照 FIFO 的方式对该 Set 的被替换的 Line 的历史进行记录。

表. Bypass L1 和 L2 的 Miss Rate 对比

Trace	Level	Miss Rate (percent)				AMAT (cycles)			
		No	L1	L2	L1+L2	No	L1	L2	L1+L2
xcg.trace	L1	20.01	24.58	20.01	24.58	12.10	13.45	11.53	14.61
	L2	35.46	32.52	32.65	37.22				
xaa.trace	L1	11.34	22.67	11.34	22.67	12.64	14.99	15.46	24.67
	L2	75.00	42.87	99.90	85.55				
gen.trace	L1	1.12	51.47	1.12	51.47	3.26	8.29	3.24	11.10
	L2	13.69	0.28	11.02	5.74				
1.trace	L1	99.64	97.59	99.64	97.59	79.48	109.96	110.38	92.07
	L2	66.76	99.60	97.77	81.27				

2.trace	L1	100.00	98.89	100.00	98.89	80.24	111.72	112.18	102.84
	L2	67.24	99.94	88.18	90.96				

gen.trace is a trace generated from 1.trace and 2.trace. 'No' means no bypassing applied; 'L1' means bypassing L1 only; 'L2' means bypassing L2 only; 'L1+L2' means bypassing both L1 and L2.

考虑本小节提到的现象，仅仅会在 Capacity miss 时发生，因此考虑在 Capacity miss 情况发生时启动 Bypassing 策略。在 MCT 未满载或 Cache 未满载时都认为发生 Compulsory miss；当前 Miss 在 MCT 中查找到，则认为是 Conflict miss；否则认为是 Capacity miss。所有被判断为 Capacity miss 的 Miss 都将被 Bypass 以防止其“污染”Cache。

添加 MCT 表，对应 Set 的表项项目通过 CacheConfig 进行设置，添加设置选项选择是否使用 Bypass 策略，并按照上述逻辑完成 MCT 的 Bypass 策略。

进行实验，Cache 配置为默认配置，替换策略使用 PLRU 算法，不进行 Prefetch，测试在 L1 和 L2 上不同的开启 Bypass 策略的配置，MCT 中每行之多存放 8 个历史 Tag。Miss rate 和 AMAT 结果如上表所示。为了测试需要，用 2.trace 的全部和 1.trace 的前 1000 行构造新的 gen.trace。

研究上表发现，MCT Bypassing 在仅 Bypass L2 时对于局部性较好的 trace，以及局部性不好的小片段穿插在局部性良好的片段中的 trace 有一定的优化；而对于局部性很差（比如流式访存）的 trace，Bypass 不仅起不到优化效果，还会增大 Miss rate。

基于这一发现，考虑可以在使用 Bypass 策略时，检测访存形式来进行一定的优化，若为局部性较好的访存则允许 Bypass；否则为流式等局部性很差的访存，禁止 Bypass 开启，以保证 Bypass 的效果良好。从保留 Cache 的角度考虑，若局部性良好，则可以通过 Bypass 策略来尽量保护需要频繁访问的 Line 不被污染；而在流式访存等局部性很差的情况下，完全没有必要对 Cache 进行保护。

综合以上的实验结果，决定**只对 L2 进行 Bypass**，再进行实验寻找最佳的 MCT Size (MCT 的一行可以记录的历史 Tag 数)。使用 01-mcf-gem5-xcg.trace, 02-stream-gem5-xaa.trace 和 gen.trace 测试，得到的不同数量级的 MCT Size 的 Miss rate 和 AMAT 的实验结果如下表。

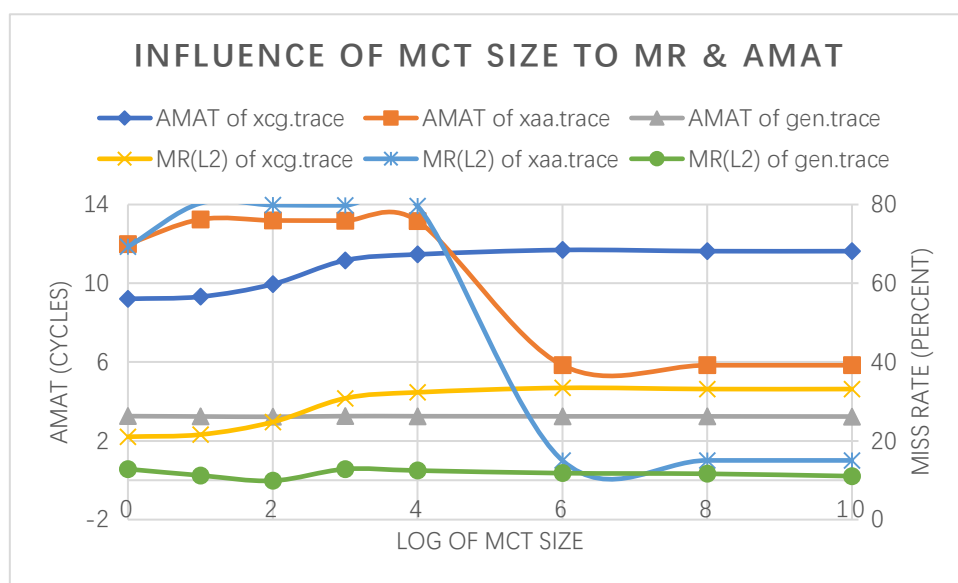


表. MCT Size 大小对 Miss Rate 和 AMAT 的影响

Trace	MCT Size	L1 MR (percent)	L2 MR (percent)	AMAT (cycles)
01-mcf-gem5-xcg.trace	0	20.007	35.463	12.096
	1	20.007	21.045	9.211
	2	20.007	21.588	9.320
	4	20.007	24.774	9.957
	8	20.007	30.801	11.163
	16	20.007	32.316	11.466
	64	20.007	33.447	11.692
	256	20.007	33.146	11.632
	1024	20.007	33.146	11.632
02-stream-gem5-xaa.trace	0	11.341	75.004	12.640
	1	11.341	69.260	11.989
	2	11.341	80.319	13.243
	4	11.341	79.852	13.190
	8	11.341	79.780	13.182
	16	11.341	79.635	13.165
	64	11.341	15.029	5.838
	256	11.341	15.029	5.838
	1024	11.341	15.029	5.838
gen.trace	0	1.118	10.869	3.233
	1	1.118	12.827	3.255
	2	1.118	11.222	3.237
	4	1.118	9.872	3.222
	8	1.118	12.823	3.255
	16	1.118	12.457	3.251

64	1.118	11.796	3.244
256	1.118	11.638	3.242
1024	1.118	11.036	3.235

MCT Size = 0 means that no bypass applied to L2. L1 MR's are always the same.

根据上表绘制 AMAT 和 L2 Miss rate 的曲线如上图，其中 xcg.trace 代表了局部性良好的访存，xaa.trace 代表了局部性不好的流式访存，gen.trace 代表了局部性良好和不好的片段穿插的访存。根据曲线进行权衡，认为 MCT Size = 1 时整体表现较为良好。

对 MCT Size 的理解，可以认为是进行 Bypass 优化的程度，MCT Size = 0 时，没有历史记忆，所有 Miss 都被判断为 Capacity miss，是最为激进的参数选择；而当 MCT Size 不断增加，直至趋近于无穷时，相当于永远记住发生过的替换历史，因此所有的 Miss 只要出现过一次就会被认为是 Conflict miss。因此合适的 MCT Size 应为二者之间的权衡。

综上，选用 MCT 判断 Miss 类型的 Bypass 策略，MCT 历史 Tag 数量为 1。

3. Prefetch 策略

由于局部性原理的存在，因而可以在发生 Miss 时不止取出引起本次 Miss 的 Cache Line，而是同时取出接连的若干个 Line 防止之后可能会出现 Miss。

Next-line Prefetcher 基于这一原理，采用最简单的 Next-line prefetch 策略进行优化。每当发生一次 Miss，在取出引起 Miss 的 Line 的同时，还继续取出后续的若干的 Line（该数目可以进行配置），由于本次 Lab 要求不允许使用 Stream buffer，因此将这些 Prefetch 得到的 Line 一并放入 Cache 中，这一做法可能会引起 Cache 污染。

MCT Next-line Prefetcher 参考 Collins et al. 1999. [2] 的论文对上述的 Next-line prefetcher 进行一定的优化：将 MCT 和 Next-line prefetcher 联动，通过 MCT 判断各个 Line 是否会导致 Capacity miss 来决定是否取出。

实现如上所述的 MCT next-line prefetcher，可以通过 stride 和 num 两个参数对其进行配置，stride 参数为预取的跨度（比如 stride = 2，当前的 Line 的编号为 A 时，预取的 Line 即为 A+2, A+4, ...），num 参数为预取的数量，上限为本次 Lab 限制的 4 个。

进行实验，Cache 配置为默认配置，替换策略使用 PLRU 算法，L2 使用基于 MCT 的 Bypass 策略，MCT 历史窗口大小为 1，对 L1 和 L2 都使用 Prefetch 策略，测试不同的 stride 和 num 下的 Miss rate 和 AMAT 如下表。

表. 不同参数的 Prefetch 策略的 Miss Rate 和 AMAT					
Trace	Prefetch		L1 MR (percent)	L2 MR (percent)	AMAT (cycles)
	Stride (lines)	Number (lines)			
01-mcf- gem5- xcg.trace	0	0	20.007	20.891	9.180
	1	1	17.923	23.520	9.008
	2	1	19.050	24.636	9.598
	4	1	19.445	24.601	9.728
	1	4	17.077	26.234	9.188
	2	4	18.813	30.281	10.578
	4	4	16.374	24.554	8.658
02-stream- gem5-	0	0	11.341	70.068	12.080
	1	1	5.672	69.359	7.501

xaa.trace	2	1	5.673	69.422	7.505
	4	1	5.675	69.431	7.508
	1	4	2.271	69.287	4.801
	2	4	2.272	69.398	4.804
	4	4	2.274	69.556	4.809
gen.trace	0	0	1.118	8.099	3.202
	1	1	0.968	15.632	3.248
	2	1	0.938	9.599	3.184
	4	1	0.912	16.903	3.245
	1	4	0.589	20.144	3.178
	2	4	0.598	19.080	3.174
	4	4	0.605	21.508	3.190

Prefetch number = 0 and Prefetch stride = 0 means that no prefetch techniques applied.

上表结果基本可以说明, Next-line prefetch 策略在参数合适的情况下基本都是有效的。下面对 L1 和 L2 的 Next-line prefetch 配置参数进行筛选。

表. Prefetch 参数筛选							
Trace	L1 Prefetch		L2 Prefetch		L1 MR	L2 MR	AMAT
	Number	Stride	Number	Stride	(percent)	(percent)	(cycles)
01-mcf-gem5-xcg.trace	0	0	0	0	20.007	20.891	9.180
	1	1	0	0	17.923	23.593	9.021
	4	4	0	0	16.374	23.727	8.522
	0	0	1	1	20.007	21.014	9.205
	0	0	4	4	20.007	21.297	9.262
	1	1	1	1	17.923	23.520	9.008
	4	4	1	1	16.374	24.672	8.677
	1	1	4	4	17.923	23.987	9.092
	4	4	4	4	16.374	24.554	8.658
02-stream-gem5-xaa.trace	0	0	0	0	11.341	70.068	12.080
	1	1	0	0	5.672	70.074	7.542
	4	4	0	0	2.274	70.306	4.826
	0	0	1	1	11.341	69.352	11.999
	0	0	4	4	11.341	69.357	12.000
	1	1	1	1	5.672	69.359	7.501
	4	4	1	1	2.274	69.675	4.812
	1	1	4	4	5.672	69.363	7.502
	4	4	4	4	2.274	69.556	4.809
gen.trace	0	0	0	0	1.118	8.099	3.202
	1	1	0	0	0.968	15.472	3.247
	4	4	0	0	0.605	21.641	3.191

0	0	1	1	1.118	10.536	3.230
0	0	4	4	1.118	13.719	3.266
1	1	1	1	0.968	15.632	3.248
4	4	1	1	0.605	23.714	3.204
1	1	4	4	0.968	19.768	3.288
4	4	4	4	0.605	21.508	3.190

Prefetch number = 0 and Prefetch stride = 0 means that no prefetch techniques applied.

通过研究上表，综合考虑这几个 trace，认为仅对 L1 进行 Prefetch，且 Prefetch Number = 4，Prefetch Stride = 4 是最佳的配置。

综上，选用 **Next-n-lines Prefetch** 策略，预取间隔为 4 个 Line，每次预取数量为 4 个 Line，并通过 MCT 辅助判断。

四 . Cache 优化

1. Cache 的优化策略及配置

将上一部分独立讨论、实现和筛选的优化策略进行组合，拼装得到对整个 Cache 的优化策略及配置如下。

L1 Cache：32KB 容量，8 路组相连，Line 大小为 64B，Write Back + Write Allocate 写策略，**PLRU 替换策略，间隔为 4 Line 预取数量为 4 的 Prefetch 策略，不进行 Bypass。**

L2 Cache：256KB 容量，8 路组相连，Line 大小为 64B，Write Back + Write Allocate 写策略，**PLRU 替换策略，不进行 Prefetch，MCT 大小为 1 的 Bypass 策略。**

2. 优化效果

采用上述优化配置，使用命令“./cache-sim -trace=./xxx.trace -iter=100”运行 Lab 3.1 提供的 1.trace 和 2.trace 以及本次 Lab 3.2 提供的 01-mcf-gem5-xcg.trace 和 02-stream-gem5-xaa.trace 共 4 个 trace 各测试 100 轮，运行结果如下图。

<pre> lc@lc-VirtualBox:~/下载/Mem/cache/build-cache-sin-Desktop_Qt_5_9_2_GCC_64bit-Debug\$./cache-sin -iter=100 -trace=01-mcf-gem5-xcg.trace L1-Cache config: Size = 32768 byte Set number = 64 Associativity = 8 Write back Write alloc Hit latency = 3 cycles Bus latency = 0 cycles Replace policy: PLRU No bypass applied Prefetch applied Prefetch stride = 4 (lines) Prefetch num = 4 (lines) L2-Cache config: Size = 262144 byte Set number = 512 Associativity = 8 Write back Write alloc Hit latency = 4 cycles Bus latency = 6 cycles Replace policy: PLRU Bypass applied MCT size = 1 No prefetch applied AMAT = 8.522327 (cycles) Total L1 access count: 23261100 Total L1 access time: 200064610 cycles L1 AAT = 8.600823 cycles Total L1 miss count: 3808673 Miss rate = 16.373572% Total L1 fetch count: 3808673 Total L1 replace count: 3808560 Total L1 prefetch count: 26680652 Total L1 bypass count: 0 Total L2 access count: 18184844 Total L2 access time: 620220340 cycles L2 AAT = 34.106441 cycles Total L2 miss count: 4314732 Miss rate = 23.727077% Total L2 fetch count: 326461 Total L2 replace count: 322365 Total L2 prefetch count: 0 Total L2 bypass count: 3988271 Total Memory access count: 4383719 Total Memory access time: 438371900 cycles Mem AAT = 100.000000 cycles </pre>	<pre> lc@lc-VirtualBox:~/下载/Mem/cache/build-cache-sin-Desktop_Qt_5_9_2_GCC_64bit-Debug\$./cache-sin -iter=100 -trace=02-stream-gem5-xaa.trace L1-Cache config: Size = 32768 byte Set number = 64 Associativity = 8 Write back Write alloc Hit latency = 3 cycles Bus latency = 0 cycles Replace policy: PLRU No bypass applied Prefetch applied Prefetch stride = 4 (lines) Prefetch num = 4 (lines) L2-Cache config: Size = 262144 byte Set number = 512 Associativity = 8 Write back Write alloc Hit latency = 4 cycles Bus latency = 6 cycles Replace policy: PLRU Bypass applied MCT size = 1 No prefetch applied AMAT = 4.825977 (cycles) Total L1 access count: 16290000 Total L1 access time: 89428350 cycles L1 AAT = 5.489770 cycles Total L1 miss count: 370400 Miss rate = 2.273788% Total L1 fetch count: 370400 Total L1 replace count: 370292 Total L1 prefetch count: 2962200 Total L1 bypass count: 0 Total L2 access count: 2467333 Total L2 access time: 202819730 cycles L2 AAT = 82.202011 cycles Total L2 miss count: 1734671 Miss rate = 70.305508% Total L2 fetch count: 54159 Total L2 replace count: 50063 Total L2 prefetch count: 0 Total L2 bypass count: 1680512 Total Memory access count: 1781464 Total Memory access time: 178146400 cycles Mem AAT = 100.000000 cycles </pre>	<pre> lc@lc-VirtualBox:~/下载/Mem/cache/build-cache-sin-Desktop_Qt_5_9_2_GCC_64bit-Debug\$./cache-sin -iter=100 -trace=gen.trace L1-Cache config: Size = 32768 byte Set number = 64 Associativity = 8 Write back Write alloc Hit latency = 3 cycles Bus latency = 0 cycles Replace policy: PLRU No bypass applied Prefetch applied Prefetch stride = 4 (lines) Prefetch num = 4 (lines) L2-Cache config: Size = 262144 byte Set number = 512 Associativity = 8 Write back Write alloc Hit latency = 4 cycles Bus latency = 6 cycles Replace policy: PLRU Bypass applied MCT size = 1 No prefetch applied AMAT = 3.191267 (cycles) Total L1 access count: 9349200 Total L1 access time: 29931200 cycles L1 AAT = 3.201472 cycles Total L1 miss count: 56516 Miss rate = 0.604501% Total L1 fetch count: 56516 Total L1 replace count: 56411 Total L1 prefetch count: 417204 Total L1 bypass count: 0 Total L2 access count: 317414 Total L2 access time: 10061240 cycles L2 AAT = 31.697531 cycles Total L2 miss count: 68690 Miss rate = 21.640508% Total L2 fetch count: 4919 Total L2 replace count: 2939 Total L2 prefetch count: 0 Total L2 bypass count: 63771 Total Memory access count: 68871 Total Memory access time: 6887100 cycles Mem AAT = 100.000000 cycles </pre>
---	---	---

对比各个 trace 的优化前后的实验结果如下表所示。

表. Cache 优化效果对比						
Trace	Before			After		
	L1 MR (percent)	L2 MR (percent)	AMAT (cycles)	L1 MR (percent)	L2 MR (percent)	AMAT (cycles)
xcg.trace	19.979	36.355	12.261	16.374	23.727	8.522
xaa.trace	11.341	74.999	12.639	2.274	70.306	4.826
gen.trace	1.118	17.043	3.302	0.605	21.641	3.191

'xcg.trace' is actually 01-mcf-gem5-xcg.trace. 'xaa.trace' is actually 02-stream-gem5-xaa.trace. 'gen.trace' is generated from the former 2.trace and 1.trace.

根据上表，01-mcf-gem5-xcg.trace 共运行 100 轮，L1 Cache Miss Rate = 16.374%，L2 Cache Miss Rate = 23.727%，最终 AMAT = 8.522 cycles = 4.261 ns；02-stream-gem5-xaa.trace 共运行 100 轮，L1 Cache Miss Rate = 2.274%，L2 Cache Miss Rate = 70.306%，最终 AMAT = 4.826 cycles = 2.413 ns。两个 trace 的 AMAT 均较于未优化的默认配置有大幅度下降，说明优化是有效的。

五 . Cache 模拟器使用方法

本次 Lab 3.2 提交的 Cache 模拟器对替换算法，Bypass 策略和 Prefetch 策略进行了更

新，具体更新请参照第三部分阅读所附源代码。附件中附有编译通过的 cache-sim 可执行文件，通过命令行启动，该程序可以接受如下的命令行参数进行设置。

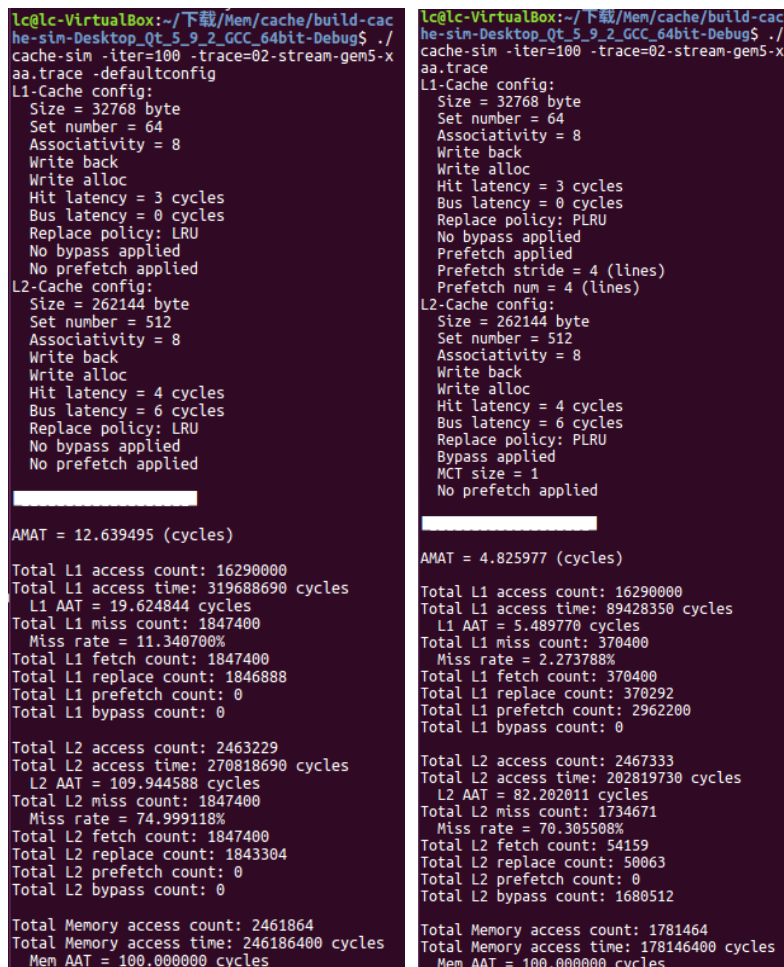
'-trace='参数设置 Cache 模拟器执行的 trace 文件，等号后制定 trace 文件的文件名；'-iter='参数设置 trace 文件模拟的轮数；'-defaultconfig'选项将 Cache 模拟器配置为未优化的默认版本，不设置该选项则使用优化过的版本。

运行模拟器的命令形式如下。

```
$ > ./cache-sim -trace=/02-stream-gem5-xaa.trace -iter=100 -defaultconfig # 默认设置，模拟当前目录下的 02-stream-gem5-xaa.trace 文件 100 轮
... ..

$ > ./cache-sim -trace=/01-mcf-gem5-xcg.trace -iter=100 -defaultconfig # 优化过的设置，模拟当前目录下的 01-mcf-gem5-xcg.trace 文件 100 轮
... ..
```

最终的运行结果会显示 AMAT 和各级 Cache 的 Miss rate，以及其他信息。形如下图。



```
lc@lc-VirtualBox:~/下载/Mem/cache/build-cache-sim-Desktop_Qt_5_9_2_GCC_64bit-Debug$ ./cache-sim -iter=100 -trace=02-stream-gem5-xaa.trace -defaultconfig
L1-Cache config:
  Size = 32768 byte
  Set number = 64
  Associativity = 8
  Write back
  Write alloc
  Hit latency = 3 cycles
  Bus latency = 0 cycles
  Replace policy: LRU
  No bypass applied
  No prefetch applied
L2-Cache config:
  Size = 262144 byte
  Set number = 512
  Associativity = 8
  Write back
  Write alloc
  Hit latency = 4 cycles
  Bus latency = 6 cycles
  Replace policy: LRU
  No bypass applied
  No prefetch applied

AMAT = 12.639495 (cycles)

Total L1 access count: 16290000
Total L1 access time: 319688690 cycles
  L1 AAT = 19.624844 cycles
Total L1 miss count: 1847400
  Miss rate = 11.340700%
Total L1 fetch count: 1847400
Total L1 replace count: 1846888
Total L1 prefetch count: 0
Total L1 bypass count: 0

Total L2 access count: 2463229
Total L2 access time: 270818690 cycles
  L2 AAT = 109.944588 cycles
Total L2 miss count: 1847400
  Miss rate = 74.999118%
Total L2 fetch count: 1847400
Total L2 replace count: 1843304
Total L2 prefetch count: 0
Total L2 bypass count: 0

Total Memory access count: 2461864
Total Memory access time: 246186400 cycles
  Mem AAT = 100.000000 cycles

lc@lc-VirtualBox:~/下载/Mem/cache/build-cache-sim-Desktop_Qt_5_9_2_GCC_64bit-Debug$ ./cache-sim -iter=100 -trace=02-stream-gem5-xaa.trace
L1-Cache config:
  Size = 32768 byte
  Set number = 64
  Associativity = 8
  Write back
  Write alloc
  Hit latency = 3 cycles
  Bus latency = 0 cycles
  Replace policy: PLRU
  No bypass applied
  Prefetch applied
  Prefetch stride = 4 (lines)
  Prefetch num = 4 (lines)
L2-Cache config:
  Size = 262144 byte
  Set number = 512
  Associativity = 8
  Write back
  Write alloc
  Hit latency = 4 cycles
  Bus latency = 6 cycles
  Replace policy: PLRU
  Bypass applied
  MCT size = 1
  No prefetch applied

AMAT = 4.825977 (cycles)

Total L1 access count: 16290000
Total L1 access time: 89428350 cycles
  L1 AAT = 5.489770 cycles
Total L1 miss count: 370400
  Miss rate = 2.273788%
Total L1 fetch count: 370400
Total L1 replace count: 370292
Total L1 prefetch count: 2962200
Total L1 bypass count: 0

Total L2 access count: 2467333
Total L2 access time: 202819730 cycles
  L2 AAT = 82.202011 cycles
Total L2 miss count: 1734671
  Miss rate = 70.305508%
Total L2 fetch count: 54159
Total L2 replace count: 50063
Total L2 prefetch count: 0
Total L2 bypass count: 1680512

Total Memory access count: 1781464
Total Memory access time: 178146400 cycles
  Mem AAT = 100.000000 cycles
```

六．参考文献

[1] Sparsh Mittal. A Survey of Cache Bypassing Techniques. JLPEA 2016.

[2] Collins, J.D. Tullsen D.M. Hardware Identification of Cache Conflict Misses. Micro 32, 1999.