天津大学

计算机组成与体系结构实践

两级 Cache 仿真器

学　　院　智能与计算学部

专　　业　计算机科学与技术

学　　号　3020202184

姓　　名　刘锦帆

2023 年 10 月 31 日

目录

1 实验目标

- 设计一个可灵活配置的两级 Cache 存储体系仿真器。
- 基于该仿真器和 SPEC 标准测试程序对 Cache 存储体系的性能进行分析。

2 实验内容

- 对实验内容（一）中单级 Cache 仿真模型进行扩展，设计一个灵活可配置的两级 Cache 存储体系仿真仿真器。
- 使用具有标准格式的访存地址流文件作为输入（该地址流已由 SPEC 标准测试程序产生），并将最终两级 Cache 中的存储内容和性能分析结果以标准格式输出到结果文件中。

3 实验结果

本实验使用 C++ 构建，提供了 Makefile，通过 `make` 指令运行，`make clean` 会删除`.o`, `.diff` 和 `.txt` 文件，如有错误，会通过 diff 指令进行输出。

执行结果如下：

```
~/Coding/CPP/TJU-2023-Computer-Organization/Proj1-2/Proj1-2/src
> make test
rm -f *.o sim_cache
rm -f *.txt *.diff
g++ -O3 -m32 -Wall   -c main.cc
g++ -O3 -m32 -Wall   -c cache.cc
g++ -o sim_cache -O3 -m32 -Wall   main.o cache.o -lm
----------DONE WITH SIM_CACHE----------
--------------------TEST START--------------------
./sim_cache 32 2048 4096 4 8 0 gcc_trace.txt > 6B.txt && diff -iw ../validation/Validation6_PartB.txt 6B.txt
./sim_cache 16 1024 8192 8 4 0 go_trace.txt > 7B.txt && diff -iw ../validation/Validation7_PartB.txt 7B.txt
./sim_cache 32 1024 0 8 0 256 perl_trace.txt > 8B.txt && diff -iw ../validation/Validation8_PartB.txt 8B.txt
./sim_cache 128 1024 4096 2 4 1024 gcc_trace.txt > 9B.txt && diff -iw ../validation/Validation9_PartB.txt 9B.txt
./sim_cache 64 8192 16384 2 4 1024 perl_trace.txt > 10B.txt && diff -iw ../validation/Validation10_PartB.txt 10B.txt
--------------------TEST DONE--------------------
```

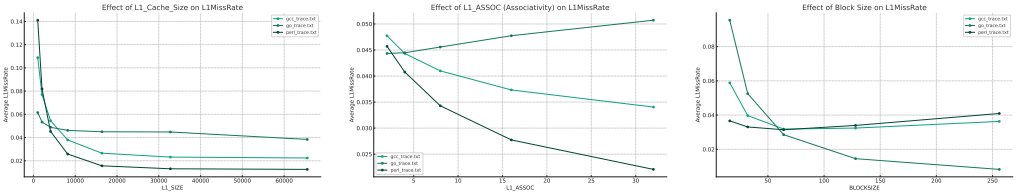
图 1: Result

4 实验分析

4.1 Miss Rate

如图 2 所示：

- 当 L1 缓存大小增加时，L1MissRate 平均值呈现下降趋势。这是符合预期的，因为更大的缓存大小通常意味着更低的缺失率。
- L1 Associativity 值与 L1MissRate 之间的关系似乎并不明确。这可能是因为其他参数的变化对缺失率产生了影响，使得 associativity 的影响不太明显。通常，增加 associativity 可以减少缺失率，但如果其他参数不合适，这种效果可能会受到抵消。



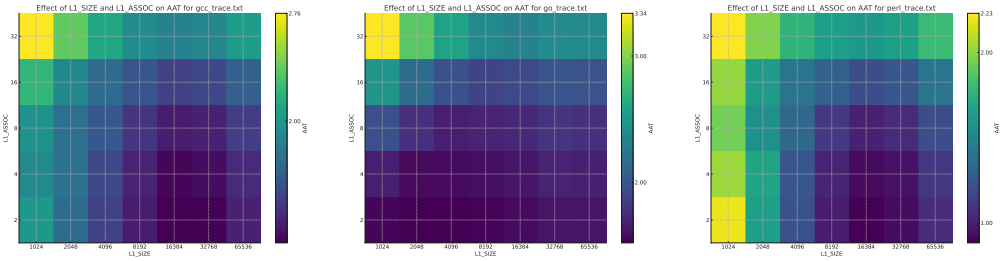
(a) L1 Cache Size (b) Associativity (c) Block Size

图 2: Miss Rate

- 对于给定的数据，块大小似乎并没有对 L1MissRate 产生显著的影响。这可能是因为块大小的选择需要与其他参数，如缓存大小和 associativity，相匹配才能获得最佳性能。

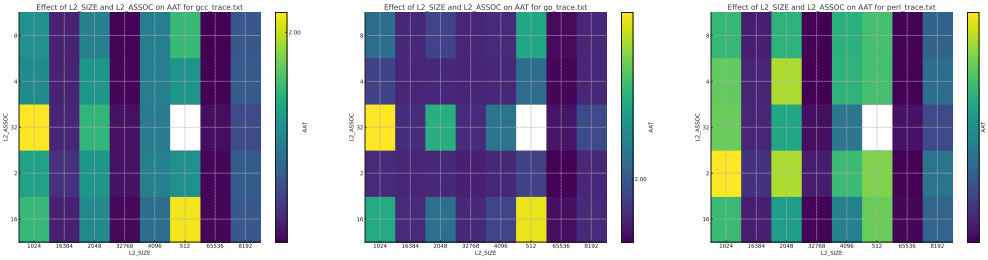
4.2 AAT

如图 3 和 4 所示：



(a) Gcc (b) Go (c) Perl

图 3: AAT - L1 Cache

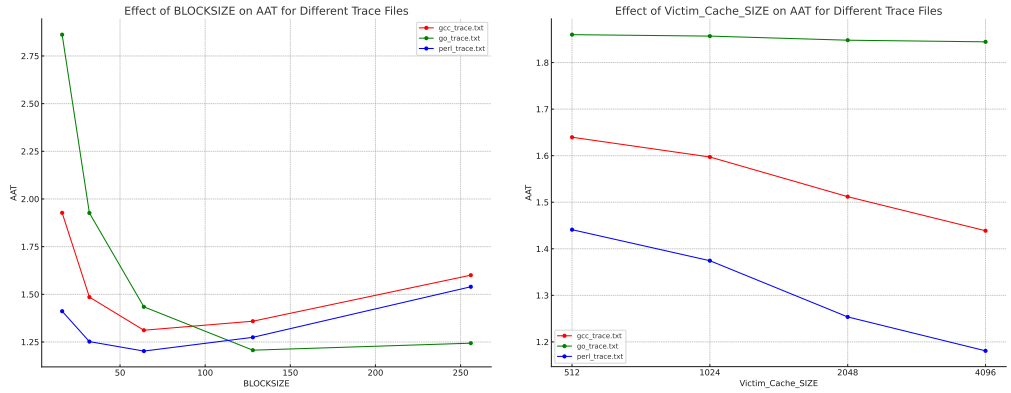


(a) Gcc (b) Go (c) Perl

图 4: AAT - L2 Cache

- 增大 L1 Cache Size 或 L1 ASSOC 通常可以降低 AAT。这可能是因为增大这些参数可以增加缓存的容量和关联性，从而降低缺失率和增加缓存命中率。
- 与 L1 缓存类似，增大 L2 SIZE 或 L2 ASSOC 也可以降低 AAT。但是，L2 缓存的影响可能不如 L1 缓存明显，因为 L2 缓存通常在 L1 缓存缺失时才被访问。

如图 5 所示：



(a) Block Size　　(b) Victim Size

图 5: AAT

- 对于所有三个 trace file，随着 BLOCKSIZE 的增加，AAT 似乎都有所减少，尤其在较小的 BLOCKSIZE 值时下降得较为明显。这可能是因为增大 BLOCKSIZE 可以提高空间局部性，从而增加缓存命中率。但是，BLOCKSIZE 过大可能会导致缓存行里的一些数据没有被利用，从而浪费缓存空间。对于 `go_trace.txt`，在 BLOCKSIZE 较大时，AAT 的减少不如其他两个明显，这可能是因为这个工作负载与其他两个有所不同。
- 对于所有三个 trace file，当 Victim Cache SIZE 为 512 时，AAT 值都是最高的。这可能是因为该大小不适合这些特定的工作负载，或者这个大小的 Victim Cache 在逐出数据时可能不够有效。随着 Victim Cache SIZE 的增加（从 512 开始），AAT 明显降低。这表明，增大 Victim Cache 的大小可以提高其效率，从而降低 AAT。当 Victim Cache SIZE 达到一定值后，例如 32 或 64，AAT 的下降趋势放缓，表明在这些值之后，进一步增加 Victim Cache 的大小可能不会带来显著的性能提升。

4.3 最优参数组合

表 1: 最优参数组合

Trace	L1	L2	blocksize	victim size	Best AAT
gcc	8192, 2	65536, 8	128	4096	0.8182
go	2048, 2	65536, 2	128	2048	0.8466
perl	8192, 2	65536, 2	64	4096	0.6714