

实验报告

一. 实验题目:

高速缓存的性能分析与优化

二. 实验内容:

模拟一级高速缓存, 尝试调整参数, 收集数据, 比较不同容量和不同相联度 Cache 的 3C 失效率。

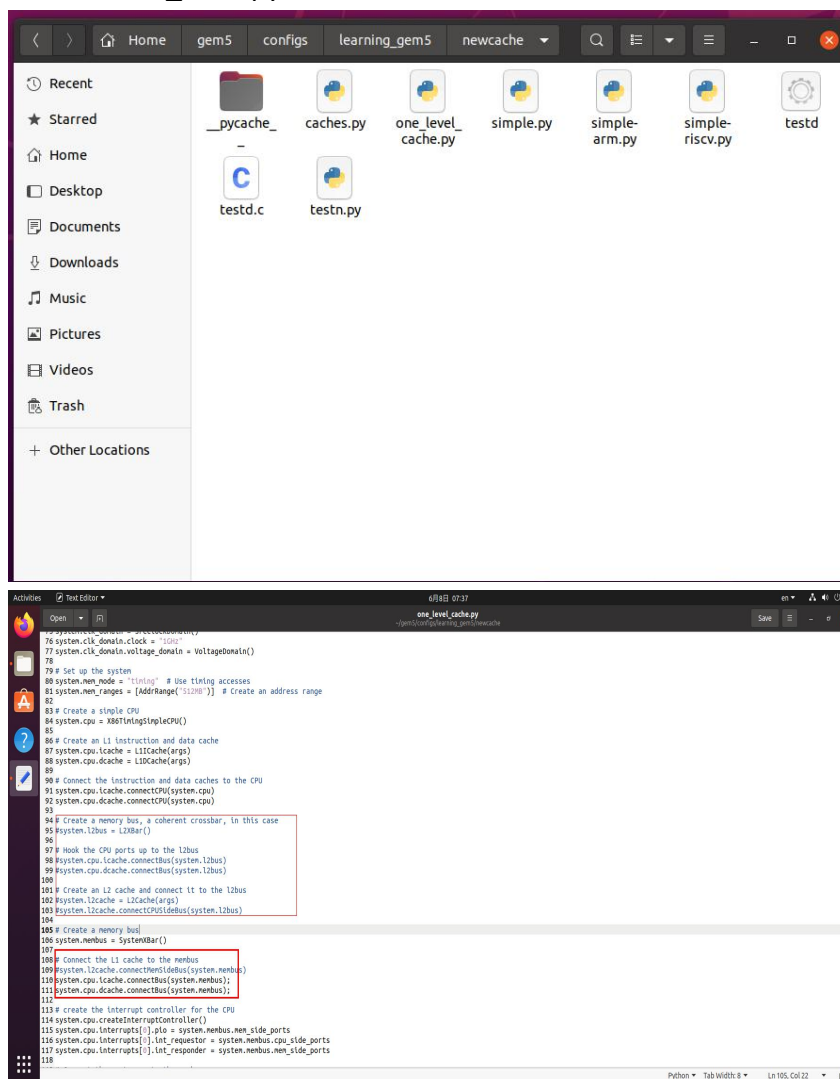
三. 实验环境:

GEM5 (也可自行选择其他工具仿真, 或者自己设计仿真)

四. 实验设计

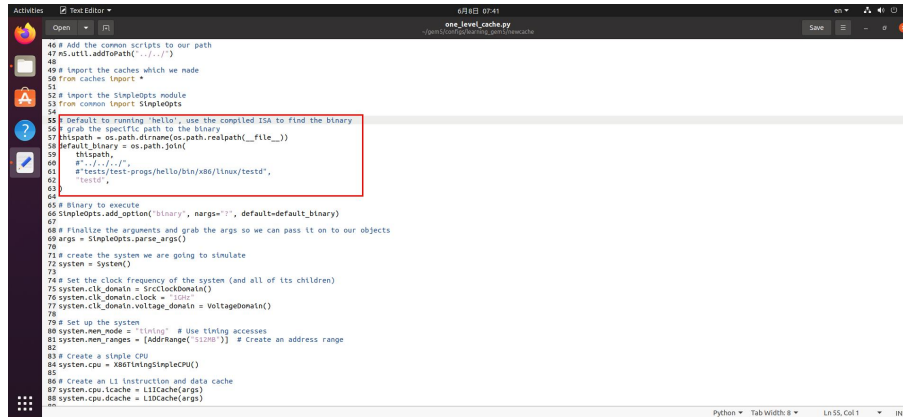
1. 一级 cache 的模拟

我们直接在 gem5 现成的二级 cache 架构进行修改, 文件地址为 gem5/configs/learning_gem5/part1, 将 part1 文件中的内容直接复制到一新建文件下, 对该目录下的二级 cache 架构进行修改, 从而避免修改源文件, 将该目录下的 two_level.py 文件内容进行修改, 将二级 cache 改为一级, 具体如下图



2. 加入所需要的测试程序 (我的是数据量为 200 的冒泡排序)

将加载好的二进制文件加载在如下位置

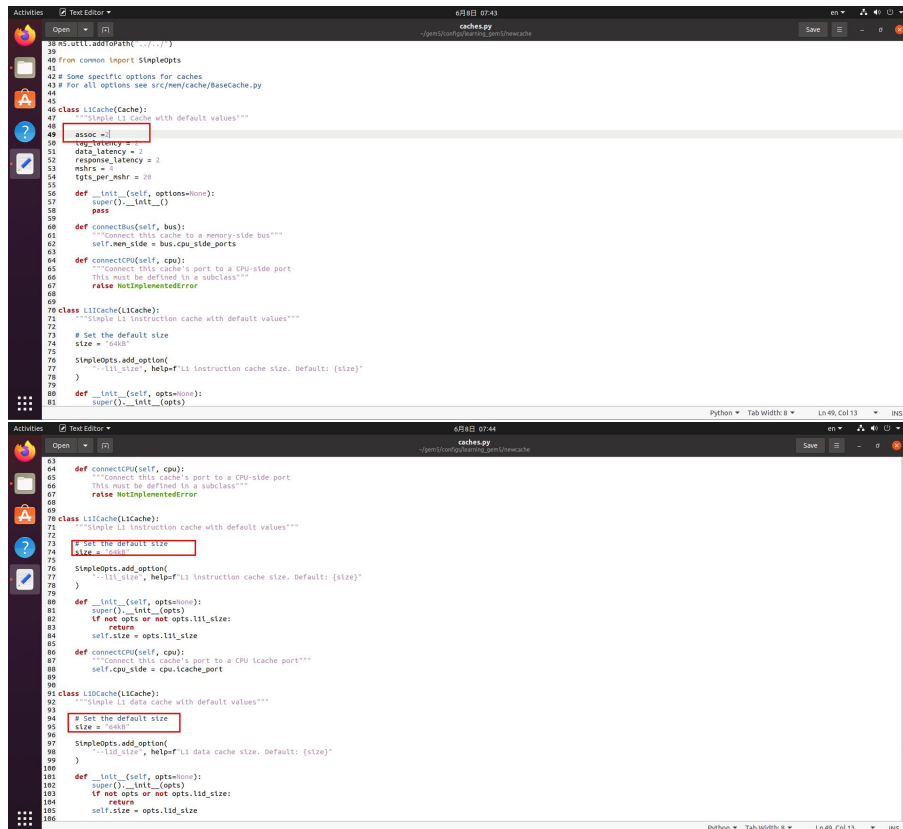


```
46 # Add the common scripts to our path
47 sys.util.add_path(../..)
48
49 # Import the caches which we made
50 from caches import *
51
52 # Import the SimpleOpts module
53 from common import SimpleOpts
54
55 # Default to running 'hello', use the compiled ISA to find the binary
56 # grab the specific path to the binary
57 hispath = os.path.dirname(os.path.realpath(__file__))
58 default_binary = os.path.join(
59     hispath,
60     # "../..",
61     # "tests/test-progs/hello/bin/x86/linux/testd",
62     "testd",
63 )
64
65 # Binary to default
66 SimpleOpts.add_option("binary", nargs="?", default=default_binary)
67
68 # Finalize the arguments and grab the args so we can pass it on to our objects
69 args = SimpleOpts.parse_args()
70
71 # create the system we are going to simulate
72 system = System()
73
74 # Set the clock frequency of the system (and all of its children)
75 system.clk_domain = srcClockDomain()
76 system.clk_domain.clock = "cpu0"
77 system.clk_domain.voltage_domain = VoltageDomain()
78
79 # Set up the system
80 system.mem_model = "timing" # Use timing accesses
81 system.mem_ranges = [AddrRange("32MB")] # Create an address range
82
83 # Create a single CPU
84 system.cpu = X86TimingSimpleCPU()
85
86 # Create an L1 instruction and data cache
87 system.cpu.cache = L1Cache(args)
88 system.cpu.cache = L1DCache(args)
```

3.修改 caches.py 文件的 cache 的容量和相联度，记录不命中率

执行 build/X86/gem5.opt ./configs/learning_gem5/newcache/one_level_cache.py

不命中率记录在 gem5/m5out/start.txt 文档



```
38 sys.util.add_path(../..)
39
40 from common import SimpleOpts
41
42 # Some specific options for caches
43 # For all options see src/mem/cache/BaseCache.py
44
45 class L1Cache(Cache):
46     """Single L1 Cache with default values"""
47
48     assoc = 1
49     data_latency = 2
50     response_latency = 2
51     mshr = 1
52     l1ts_per_mshr = 20
53
54     def __init__(self, options=None):
55         super().__init__()
56
57         pass
58
59     def connectBus(self, bus):
60         """Connect this cache to a memory-side bus"""
61         self.mem_side = bus.cpu_side_ports
62
63     def connectCPU(self, cpu):
64         """Connect this cache's port to a CPU-side port
65         This must be defined in a subclass"""
66         raise NotImplementedError
67
68
69 class L1ICache(L1Cache):
70     """Single L1 instruction cache with default values"""
71
72     # Set the default size
73     size = "64KB"
74
75     SimpleOpts.add_option(
76         "--l1_size", help="L1 instruction cache size. Default: {size}"
77     )
78
79     def __init__(self, opts=None):
80         super().__init__(opts)
81
82         if not opts or not opts.l1_size:
83             return
84         self.size = opts.l1_size
85
86     def connectCPU(self, cpu):
87         """Connect this cache's port to a CPU cache port"""
88         self.cpu_side = cpu.cache_port
89
90
91 class L1DCache(L1Cache):
92     """Single L1 data cache with default values"""
93
94     # Set the default size
95     size = "64KB"
96
97     SimpleOpts.add_option(
98         "--l1d_size", help="L1 data cache size. Default: {size}"
99     )
100
101     def __init__(self, opts=None):
102         super().__init__(opts)
103         if not opts or not opts.l1d_size:
104             return
105         self.size = opts.l1d_size
106
```

五. 运行过程截图

```
Activities Terminal 6月8日 07:47 fly@ubuntu: ~/gem5

fly@ubuntu:~/gem5$ build/X86/gem5.opt ./configs/learning_gem5/newcache/one_level_cache.py
gem5 Simulator System. https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

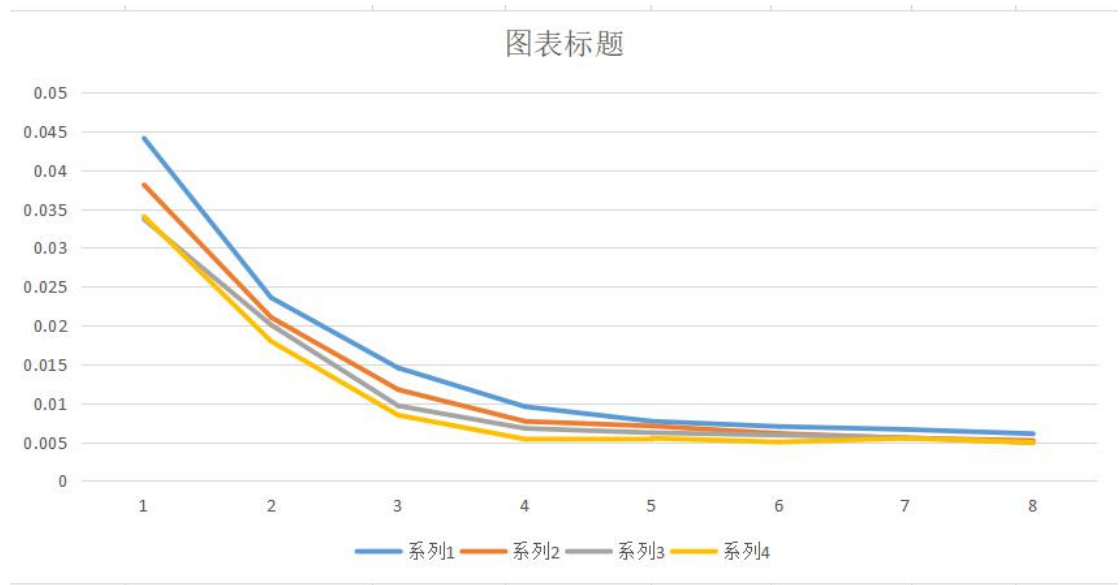
gem5 version 23.1.0.0
gem5 compiled May 31 2024 05:29:15
gem5 started Jun 8 2024 07:47:03
gem5 executing on ubuntu, pid 2276
command line: build/X86/gem5.opt ./configs/learning_gem5/newcache/one_level_cache.py

Global frequency set at 100000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and pdf.
src/main/rom/Interface.cc:695: warn: DRAM device capacity (8320 Mbytes) does not match the address range assigned (512 Mbytes)
src/base/statistics.hh:279: warn: One of the stats is a legacy stat. Legacy stat is a stat that does not belong to any statistics::Group. Legacy stat is deprecated.
system.remote_gdb: listening for connections on port 7000
Beginning simulation
src/sim/simulate.cc:199: info: Entering event queue @ 0. Starting simulation...
src/sim/new_state.cc:448: info: Increasing stack size by one page.
src/sim/syscall_emul.cc:74: warn: Ignoring syscall nprotect(...)
src/sim/syscall_emul.cc:74: warn: Ignoring syscall nprotect(...)
src/sim/syscall_emul.cc:74: warn: Ignoring syscall nprotect(...)
Sorted array:
2 8 12 15 16 32 34 39 42 46 47 57 57 63 64 64 66 70 75 81 87 100 102 105 106 108 108 110 121 122 131 131 131 136 145 147 147 148 153 157 164 165 167 175 177 177 179 198 198 200 200 210 221 22
3 228 234 235 242 243 252 255 256 259 261 268 272 274 279 281 283 283 310 320 321 329 330 342 345 362 371 372 373 381 393 403 410 414 418 429 430 433 439 444 452 460 465 465 469 470 472 476 485 494 4
99 585 589 510 511 519 525 545 547 550 555 561 572 589 590 597 597 606 609 612 618 619 622 623 625 627 629 630 636 637 638 661 665 672 672 695 700 700 706 707 717 718 722 723 727 761 772 774 778 783 788
888 813 816 817 818 828 833 837 842 847 853 861 863 864 866 875 876 880 883 884 889 891 892 892 893 893 902 905 923 928 930 930 930 941 943 947 949 960 961 964 975 994 994
Exiting @ tick 398983000 because exiting with last active thread context
fly@ubuntu:~/gem5$
```

Statistics		Unit		Value		Description	
122	system.cpu.processor.eventQueueAccesses::total		2904			# number of queue accesses (Count)	
124	system.cpu.cache.overallMisses::cpu.data		2904			# number of overall misses (Count)	
125	system.cpu.cache.overallMisses::total		2904			# number of overall misses (Count)	
126	system.cpu.cache.demandMissLatency::cpu.data		176440000			# number of demand (readwrite) miss ticks (Tick)	
127	system.cpu.cache.demandMissLatency::total		176440000			# number of demand (readwrite) miss ticks (Tick)	
128	system.cpu.cache.overallMissLatency::cpu.data		176440000			# number of overall miss ticks (Tick)	
129	system.cpu.cache.overallMissLatency::total		176440000			# number of overall miss ticks (Tick)	
130	system.cpu.cache.demandAccesses::cpu.data		526470			# number of demand (readwrite) accesses (Count)	
131	system.cpu.cache.demandAccesses::total		526470			# number of demand (readwrite) accesses (Count)	
132	system.cpu.cache.overallAccesses::cpu.data		526470			# number of overall (readwrite) accesses (Count)	
133	system.cpu.cache.overallAccesses::total		526470			# number of overall (readwrite) accesses (Count)	
134	system.cpu.cache.demandMissRate::cpu.data		0.005516			# miss rate for demand accesses (Ratio)	
135	system.cpu.cache.demandMissRate::total		0.005516			# miss rate for demand accesses (Ratio)	
136	system.cpu.cache.demandMissRate::cpu.data		0.005516			# miss rate for demand accesses (Ratio)	
137	system.cpu.cache.demandMissRate::total		0.005516			# miss rate for demand accesses (Ratio)	
138	system.cpu.cache.demandAvgMissLatency::cpu.data		60757.575758			# average overall miss latency in ticks ((Tick/Count))	
139	system.cpu.cache.demandAvgMissLatency::total		60757.575758			# average overall miss latency in ticks ((Tick/Count))	
140	system.cpu.cache.demandAvgMissLatency::cpu.data		60757.575758			# average overall miss latency in ticks ((Tick/Count))	
141	system.cpu.cache.demandAvgMissLatency::total		60757.575758			# average overall miss latency in ticks ((Tick/Count))	
142	system.cpu.cache.blockedCycles::no_mshr		0			# number of cycles access was blocked (Cycle)	
143	system.cpu.cache.blockedCycles::no_targets		0			# number of cycles access was blocked (Cycle)	
144	system.cpu.cache.blockedCycles::no_mshr		0			# number of lines access was blocked (Count)	
145	system.cpu.cache.blockedCycles::no_targets		0			# number of lines access was blocked (Count)	
146	system.cpu.cache.avgBlocked::no_mshr		nan			# average number of cycles each access was blocked ((Cycle/Count))	
147	system.cpu.cache.avgBlocked::no_targets		nan			# average number of cycles each access was blocked ((Cycle/Count))	
148	system.cpu.cache.writebacks::writebacks		543			# number of writebacks (Count)	
149	system.cpu.cache.writebacks::total		543			# number of writebacks (Count)	
150	system.cpu.cache.demandMshrMisses::cpu.data		2904			# number of demand (readwrite) MSHR misses (Count)	
151	system.cpu.cache.demandMshrMisses::total		2904			# number of demand (readwrite) MSHR misses (Count)	
152	system.cpu.cache.overallMshrMisses::cpu.data		2904			# number of overall MSHR misses (Count)	
153	system.cpu.cache.overallMshrMisses::total		2904			# number of overall MSHR misses (Count)	
154	system.cpu.cache.demandMshrMissLatency::cpu.data		170632000			# number of demand (readwrite) MSHR miss ticks (Tick)	
155	system.cpu.cache.demandMshrMissLatency::total		170632000			# number of demand (readwrite) MSHR miss ticks (Tick)	
156	system.cpu.cache.demandMshrMissLatency::cpu.data		170632000			# number of overall MSHR miss ticks (Tick)	
157	system.cpu.cache.demandMshrMissLatency::total		170632000			# number of overall MSHR miss ticks (Tick)	
158	system.cpu.cache.demandMshrMissRate::cpu.data		0.005516			# mshr miss ratio for demand accesses (Ratio)	
159	system.cpu.cache.demandMshrMissRate::total		0.005516			# mshr miss ratio for demand accesses (Ratio)	
160	system.cpu.cache.demandMshrMissRate::cpu.data		0.005516			# mshr miss ratio for overall accesses (Ratio)	
161	system.cpu.cache.demandMshrMissRate::total		0.005516			# mshr miss ratio for overall accesses (Ratio)	
162	system.cpu.cache.demandAvgMshrMissLatency::cpu.data		58757.575758			# average overall mshr miss latency ((Tick/Count))	
163	system.cpu.cache.demandAvgMshrMissLatency::total		58757.575758			# average overall mshr miss latency ((Tick/Count))	
164	system.cpu.cache.demandAvgMshrMissLatency::cpu.data		58757.575758			# average overall mshr miss latency ((Tick/Count))	
165	system.cpu.cache.demandAvgMshrMissLatency::total		58757.575758			# average overall mshr miss latency ((Tick/Count))	
166	system.cpu.cache.avgBlocked::no_mshr		0.000			# number of cycles access was blocked (Cycle)	

六. 实验结论

E	F	G	H	I
cache容量\相联度	1	2	4	8
1kB	0.044099	0.038099	0.033664	0.034038
2kB	0.023589	0.021053	0.020085	0.017965
4kB	0.014546	0.011754	0.009672	0.008465
8kB	0.009554	0.007661	0.006756	0.005377
16kB	0.007672	0.007065	0.006202	0.005489
32kB	0.006998	0.006113	0.005896	0.005002
64kB	0.006621	0.005516	0.005571	0.005484
128kB	0.006074	0.005195	0.004908	0.004954



我们可以得出如下结论：

- (1) 当 **cache** 容量相同时，相联度越高，不命中率会越低
- (2) 相联度相同时，**cache** 容量越大，不命中率越低
- (3) **cache** 容量相同时，相联度大于 4 之后，不命中率变化变得不太明显

七. 实验感想

这里可以写实验过程中遇到的困难、如何调试，也可以写实验感想。随意发挥。