IS2202 COMPUTER SYSTEMS ARCHITECTURE

Lab 1: Simulating Caches and RISC-V Processors Cores in GEM5

YiHang Chen: yihche@ug.kth.se
RuoLan Wang: ruolanw@kth.se

Exercise 1: Run and analyze statistics

Question 1) How long time did the application execute inside the simulator system? How long time did the application execution take outside the simulator system? How much slower was a computer simulation? Why is it this much slower?

Execution time inside the simulator: 0.069903 seconds.

Execution time outside the simulator: 77.21 seconds.

The simulation is 1104 times slower. This is because we are executing every tick of this computer. The host could simulate 19395083 per second. But the simulation frequency of this system is 1000000000000(tick/second) .The data is shown as figure 1

simSeconds	0.069903	# Number of seconds simulated (Second)
simTicks	69903155000	# Number of ticks simulated (Tick)
finalTick	5768099395000	# Number of ticks from beginning of simulation (restored from checkpoints and never reset) (Tick)
simFreq	1000000000000	# The number of ticks per simulated second ((Tick/Second))
hostSeconds	77.21	# Real time elapsed on the host (Second)
hostTickRate	905388052	# The number of ticks simulated per host second (ticks/s) ((Tick/Second))
hostMemory	8554348	# Number of bytes of host memory used (Byte)
simInsts	1497456711	# Number of instructions simulated (Count)
simOps	1498493882	# Number of ops (including micro ops) simulated (Count)
hostInstRate	19395083	# Simulator instruction rate (inst/s) ((Count/Second))
hostOpRate	19408514	# Simulator op (including micro ops) rate (op/s) ((Count/Second))

Figure 1. stats of HotSpot3D

Question 2) Under gem5 there is a library called m5out. Inspect the m5out/stats.txt.

a) How many instructions was simulated (in Gem5)? 1497456711

b) How many host seconds passed (on the VM Ubuntu running Gem5)? 77.21 seconds

c) How many seconds passed in the simulated machine (UcanLinux)? 0.06903

d) What was the hostInstRate (on VM Ubuntu running Gem5)? 19395083

e) How much slower is it to run the simulation compared to run it Natively (compare hostSeconds with simSeconds) 1104 times

Question 3) Now, repeat above steps but also for the applications SRAD and LUD.

And the results are shown in Figure 2 and Figure 3.

SRAD:

Simulated instructions: 1603987410

hostSeconds: 1976.64

simSeconds: 1.761877

hostInstRate: 8111473

hostSeconds/simSeconds: 1118 times

simSeconds	1.761877	# Number of seconds simulated (Second)
simTicks	1761876842000	# Number of ticks simulated (Tick)
finalTick	39178003338000	# Number of ticks from beginning of simulation (restored from checkpoints and never reset) (Tick)
simFreq	1000000000000	# The number of ticks per simulated second ((Tick/Second))
hostSeconds	1976.64	# Real time elapsed on the host (Second)
hostTickRate	891350454	# The number of ticks simulated per host second (ticks/s) ((Tick/Second))
hostMemory	8554352	# Number of bytes of host memory used (Byte)
simInsts	1603987410	# Number of instructions simulated (Count)
simOps	1608846109	# Number of ops (including micro ops) simulated (Count)
hostInstRate	811473	# Simulator instruction rate (inst/s) ((Count/Second))
hostOpRate	813931	# Simulator op (including micro ops) rate (op/s) ((Count/Second))

Figure2.the stats of SRAD

LUD:

Simulated instructions: 507033890

hostSeconds: 535.00

simSeconds: 0.450742

hostInstRate: 947732

hostSeconds/simSeconds: 1186 times

simSeconds	0.450742	# Number of seconds simulated (Second)
simTicks	450742217000	# Number of ticks simulated (Tick)
finalTick	7192850637000	# Number of ticks from beginning of simulation (restored from checkpoints and never reset) (Tick)
simFreq	1000000000000	# The number of ticks per simulated second ((Tick/Second))
hostSeconds	535.00	# Real time elapsed on the host (Second)
hostTickRate	842513449	# The number of ticks simulated per host second (ticks/s) ((Tick/Second))
hostMemory	8554352	# Number of bytes of host memory used (Byte)
simInsts	507033890	# Number of instructions simulated (Count)
simOps	507525332	# Number of ops (including micro ops) simulated (Count)
hostInstRate	947732	# Simulator instruction rate (inst/s) ((Count/Second))
hostOpRate	948650	# Simulator op (including micro ops) rate (op/s) ((Count/Second))

Figure 3.the stats of LUD

Exercise 2: Checkpoints

Question 4: How much faster did it take to launch from the checkpoint compared to booting the entire system?

Much faster. Without a checkpoint, it may take more than 10 minutes. But with a checkpoint, it just takes few seconds. And it really helps to boot the whole system because we do not have to wait for the startup times.

Exercise 3: Simulating a cache memory hierarchy

Question 5.1 How much slower did it run without Cache?

Test the LUD without cache:

simSeconds: 23.154063

hostSeconds: 1727.28

Compared to the LUD with cache, the simSeconds is 51 times slower.

simSeconds	23.154063	# Number of seconds simulated (Second)
simTicks	23154062822000	# Number of ticks simulated (Tick)
finalTick	57297664827000	# Number of ticks from beginning of simulation (restored from checkpoints and never reset) (Tick)
simFreq	1000000000000	# The number of ticks per simulated second ((Tick/Second))
hostSeconds	1727.28	# Real time elapsed on the host (Second)
hostTickRate	13404942758	# The number of ticks simulated per host second (ticks/s) ((Tick/Second))
hostMemory	8540016	# Number of bytes of host memory used (Byte)
simInsts	537267627	# Number of instructions simulated (Count)
simOps	537782187	# Number of ops (including micro ops) simulated (Count)
hostInstRate	311049	# Simulator instruction rate (inst/s) ((Count/Second))
hostOpRate	311347	# Simulator op (including micro ops) rate (op/s) ((Count/Second))

Figure 4. the stats of LUD without cache

Question 5.2 Chose one of the three applications of your choice, and run them with the cache (use the checkpoint!). What was the impact on the performance compared to with the case NoCache? Why? How many clock cycles did it cost in Latency? Test the LUD application with privateL1 privateL2.

Test the LUD application with privateL1 privateL2.

simSeconds 0.457489

hostSeconds 506.98

simSeconds	0.457489	# Number of seconds simulated (Second)
simTicks	457489268000	# Number of ticks simulated (Tick)
finalTick	33892533996000	# Number of ticks from beginning of simulation (restored from checkpoints and never reset) (Tick)
simFreq	1000000000000	# The number of ticks per simulated second ((Tick/Second))
hostSeconds	506.98	# Real time elapsed on the host (Second)
hostTickRate	902387634	# The number of ticks simulated per host second (ticks/s) ((Tick/Second))
hostMemory	8552304	# Number of bytes of host memory used (Byte)
simInsts	525924684	# Number of instructions simulated (Count)
simOps	526472390	# Number of ops (including micro ops) simulated (Count)
hostInstRate	1037375	# Simulator instruction rate (inst/s) ((Count/Second))
hostOpRate	1038455	# Simulator op (including micro ops) rate (op/s) ((Count/Second))

Figure 5.the stats of LUD with PrivateL1PrivateL2CacheHierarchy(l1d_size="2KiB", l1i_size="2KiB", l2_size=”64KiB”)

board.cache_hierarchy.l1dcaches.overallHits::processor.cores.core.data 93348978 # number of overall hits (Count)

Figure 6. l1dcaches.overallHits: 93348978

board.cache_hierarchy.l1icaches.overallHits::total 355831776 # number of overall hits (Count)

Figure 7. l1icaches.overallHits: 355831776

board.cache_hierarchy.l1dcaches.overallMisses::total 370809 # number of overall misses (Count)

Figure 8. l1dcaches.overallMisses: 370809

board.cache_hierarchy.l1icaches.overallMisses::total 34063 # number of overall misses (Count)

Figure 9. l1icaches.overallMisses: 34063

board.cache_hierarchy.l2caches.overallHits::total 506728 # number of overall hits (Count)

Figure 10.l2caches.overallHits: 506728

board.cache_hierarchy.l2caches.overallMisses::total 41729 # number of overall misses (Count)

Figure 11.l2caches.overallMisses: 41729

Compared to the case NoCache, the LUD application with private L1 and private L2 is much faster. According to the hits number and miss number of L1 cache and L2 cache, we will find that most of the accesses could hit on L1 cache or L2 cache, which means the processor does not need to access memory frequently, which saves a lot of time

Exercise 4: Finding out the application cache working set

Question 6: Identify the cache working set size for each benchmark. Plot a graph showing how the execution time is reduced as a function of increased cache size

The results are shown in Table 1,2,3. And then we draw plots in Figure 12,13,14.
Since the L2 cache is 64KiB, there is no need to make the L1 cache bigger then L2 cache

Table 1. LUD's Execution Time with Different L1D Cache Size

L1D cache size(KiB)	Execution Time(s)
2	0.457489
4	0.456842
8	0.456543
16	0.454716
32	0.453644
64	0.451959

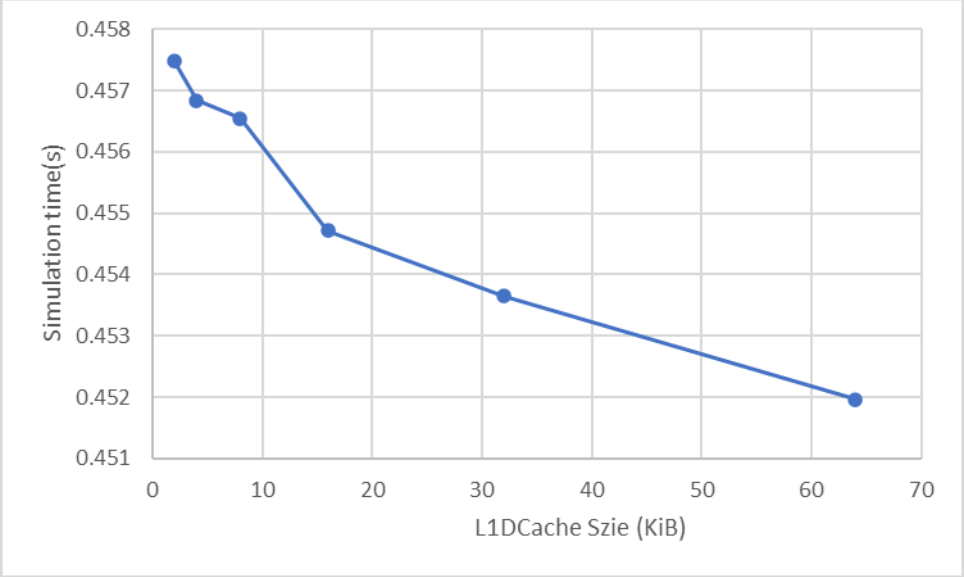

Figure 12 Relationship between Execution time and L1D Cache Size (LUD)

Table 2. SRAD's Execution Time with Different L1D Cache Size

L1D cache size(KiB)	Execution Time(s)
2	2.020320
4	1.870934
8	1.847100

16	1.842533
32	1.837110
64	1.833649

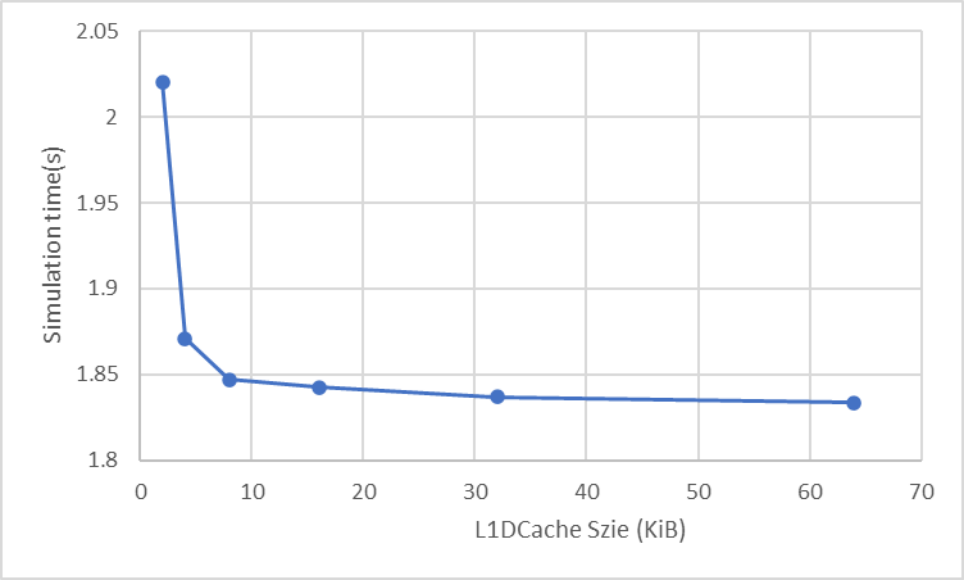

Figure 13 Relationship between Execution time and L1D Cache Size (SRAD)

Table 3. HotSpot3D's Execution Time with Different L1D Cache Size

L1D cache size(KiB)	Execution Time(s)
2	0.071510
4	0.071548
8	0.071450
16	0.071020
32	0.070641
64	0.070640

Figure 14 Relationship between Execution time and L1D Cache Size (HotSpot3D)

Question 7: Next, for the three applications (and for the best configuration above), repeat the action by doubling the L2 cache size, and monitor the change in execution performance. How did the performance change? How well did the L2 cache absorb misses from the L1? What is the total miss latency the processor experience due to misses to the cache?

Table 4. Execution Time with Different L2 Cache Size

Benchmarks	L2 Cache Size (KiB)	Execution Time (s)
HotSpot	64	0.070640
HotSpot	128	0.070640
SRAD	64	1.833649
SRAD	128	1.792921
LUD	64	0.451959
LUD	128	0.451595

From Table 4, we can see the execution time is slightly decreased by doubling the L2 cache size.

Table 5. L1 Cache Misses with Different L2 Cache Size

Benchmarks	L2 Cache Size (KiB)	Number of L1D Cache Misses	Number of L1I Cache Misses
HotSpot	64	26543	29030

HotSpot	128	26325	29030
SRAD	64	4934052	1907568
SRAD	128	4916667	1907567
LUD	64	21951	34639
LUD	128	21805	34269

Form Table 5, we know the number of L1 cache misses absorbed by increasing L2D Data-cache Sizes. We can see that there is no obvious influence on absorbed L1 cache misses

Table 6. L1 Cache Misses with Different L2 Cache Size

Benchmarks	L2 Cache Size (KiB)	Number of L1D Miss Latency (ticks)	Number of L1I Miss Latency (ticks)
HotSpot	64	2933756273	629545000
HotSpot	128	2934461936	621639000
SRAD	64	516077733126	33749273000
SRAD	128	417408353175	33070062000
LUD	64	1996985070	920453000
LUD	128	1893047071	657063000

From Table 6, we know the cache miss latency with different L2 cache size

Question 8: Which benchmark has the largest working set?

SRAD has the largest working set because the simulation needs the longest time and other date related to the time and the number of instruction also dominates the other two.

Question 9: In the above measurements, compute the L1 and L2 cache hit rates. Which application has the highest L1 and L2 cache hits? Which has the lowest? Discuss the results.

Table 7. L1 L2 Cache Hit Rate with Different L2 Cache Size

Benchmarks	L2 Cache Size (KiB)	L1D Hit Rate	L1I Hit Rate	L2 Hit Rate
HotSpot	64	99.8615%	99.9415%	57.3713%
HotSpot	128	99.8828%	99.9415%	57.3200%

SRAD	64	97.8687%	99.8601%	54.5343%
SRAD	128	97.8760%	99.8602%	66.6571%
LUD	64	99.9766%	99.9903%	53.1030%
LUD	128	99.9767%	99.9904%	61.8718%

We have L1D hit rate, L1I hit rate, L2 hit rate separately shown in Table 7. Among the results, LUD has the highest hit rates for both L1 and L2 cache. And SRAD has the lowest hit rates overall. When doubling the L2 cache size, we can find SRAD L2 hit rate increases significantly. The reason could be the SRAD algorithm is relatively sensitive to cache size and configuration.

Exercise 5: Multithreading and Speed-up

Question 10: Study the impact on changing the number of cores on the different applications, and plot how the performance improves with the number of cores. Discuss the results.

The results are shown in Table 8. The results of HotPot3D does not show an obvious speedup. Maybe it's because the HotSpot3D is not parallelized.

The results of SRAD and LUD show that the speed of 2 cores is approximately equal to twice as the 1 core speed. Figure 15 to Figure 17 show the different performance.

Table 8. Performance of one-core and two-core

	HotSpot3D		SRAD		LUD	
Core number	1 core	2 core	1 core	2 core	1 core	2 core
Number of second simulated	0.070640	0.070723	1.833649	0.900428	0.451959	0.230306

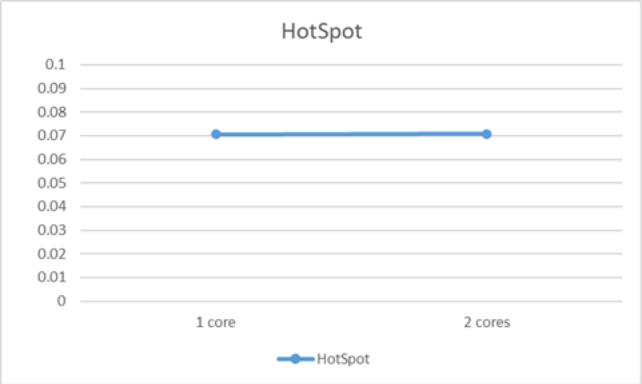

Figure 15 Hotspot performance increased by doubling the number of cores

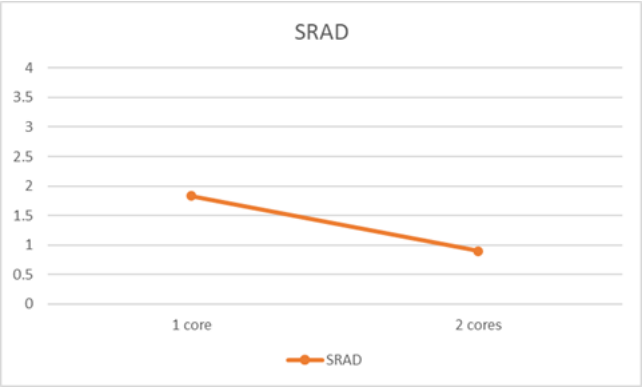

Figure 16 SRAD performance increased by doubling the number of cores

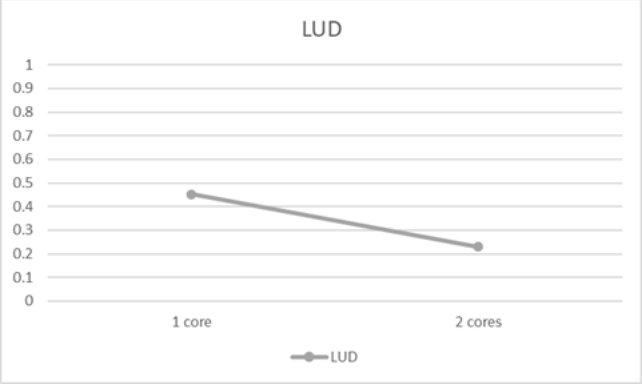

Figure 17 LUD performance increased by doubling the number of cores

Question 11: One of the applications have not been parallelized, and thus could not benefit from more cores. Which application was this?

It's HotSpot. The two simulation results of HotSpot are very similar. It could not benefit from more cores