Report of MatMul Simulation of Project 2 in Computer Architecture

By
Kenneth Nnadi
And
Quang Dang

Part 1 table

```
Part 1:
           Cache     Hits  Misses  AvgMissLatency  Hit_Percentage
Unified L1 cache 11810614    3490       89,913.75           99.97
```

The cache size for the part is 256KB and associativity of 2.

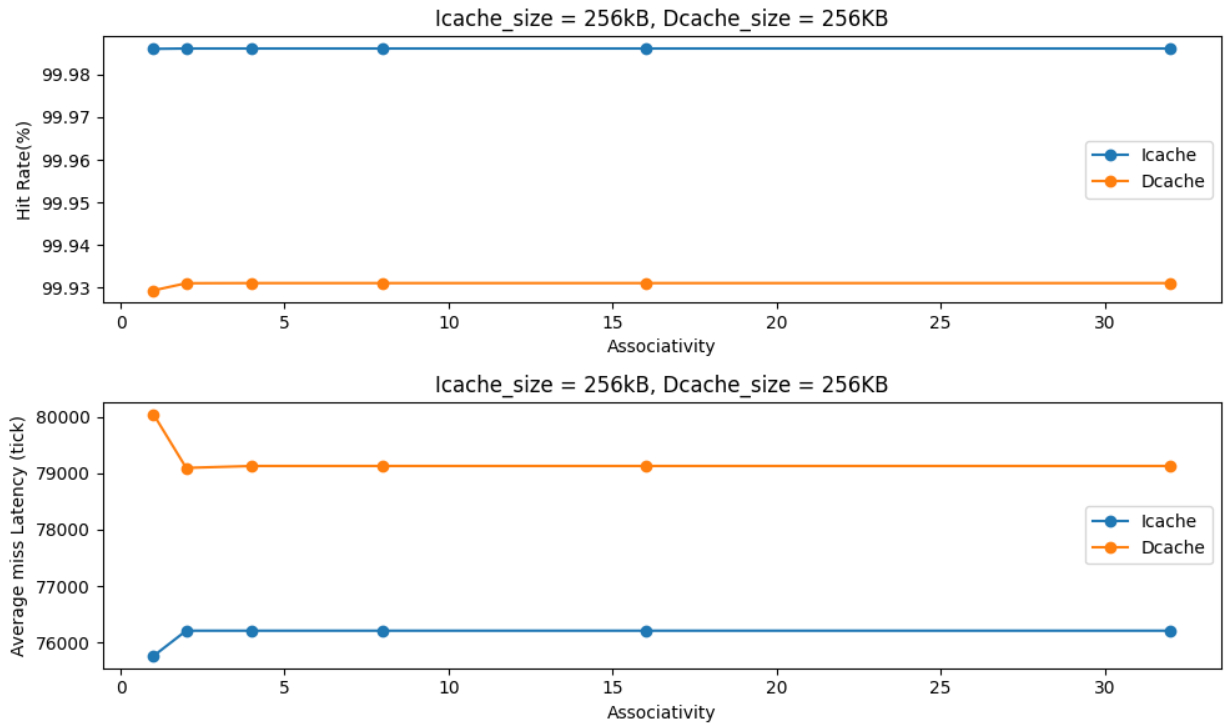
Part2 table

```
Part 2:
                       Cache  Dcache_Hits  Dcache_Misses  Dcache_AvgMissLatency  Icache_Hits  Icache_Misses  Icache_AvgMissLatency  Dcache_Hit_Percentage  Icache_Hit_Percentage
L1 data and instruction caches      3312982           2285             79,097.59      8497655           1182              76,206.43                  99.93                  99.99
```

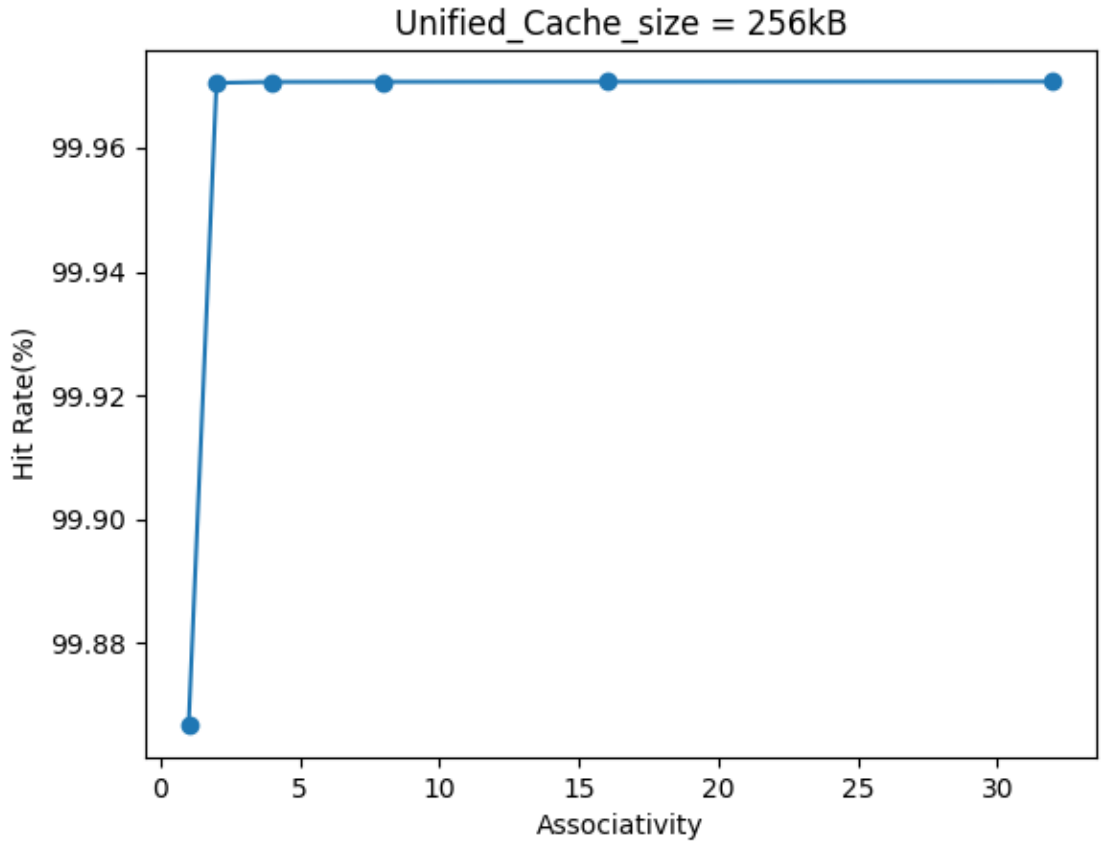
The cache size for both ichache and dcache are 256KB and associativity of 2.

Part 3:

```
Part 3:
                                      Cache     Hits  Misses  AvgMissLatency  Hit_Percentage
             Unified L1 cache (Associativity=1) 11810614    3490       89,913.75           99.97
             Unified L1 cache (Associativity=2) 11810614    3490       89,913.75           99.97
             Unified L1 cache (Associativity=4) 11810632    3472       89,985.31           99.97
             Unified L1 cache (Associativity=8) 11810634    3470       90,031.99           99.97
            Unified L1 cache (Associativity=16) 11810636    3468       90,078.72           99.97
            Unified L1 cache (Associativity=32) 11810638    3466       90,125.50           99.97
         L1 instruction caches (Associativity=1)  8497648    1189       75,756.09           99.93
         L1 instruction caches (Associativity=2)  8497655    1182       76,206.42           99.93
         L1 instruction caches (Associativity=4)  8497655    1182       76,206.42           99.93
         L1 instruction caches (Associativity=8)  8497655    1182       76,206.42           99.93
        L1 instruction caches (Associativity=16)  8497655    1182       76,206.42           99.93
        L1 instruction caches (Associativity=32)  8497655    1182       76,206.42           99.93
                L1 data caches (Associativity=1)  3312926    2341       80,052.54           99.93
                L1 data caches (Associativity=2)  3312982    2285       79,097.59           99.93
                L1 data caches (Associativity=4)  3312983    2284       79,131.79           99.93
                L1 data caches (Associativity=8)  3312983    2284       79,131.79           99.93
               L1 data caches (Associativity=16)  3312983    2284       79,131.79           99.93
               L1 data caches (Associativity=32)  3312983    2284       79,131.79           99.93
```



In this plot, we are fixing the size of icache and dcache to 256KB. Initially in the hit rate vs associativity, the dcache hit rate slightly increased and became stable after associativity reaches 2. Also, in the Average miss latency vs associativity, after associativity of 2, the Icache slightly increased while the dcache dropped and became stable. This makes sense that the dcache miss latency increased because the hit rate of dcache improved which led to lower miss latency. Surprisingly, the icache miss latency increases even though the hit rate remained mostly unchanged.

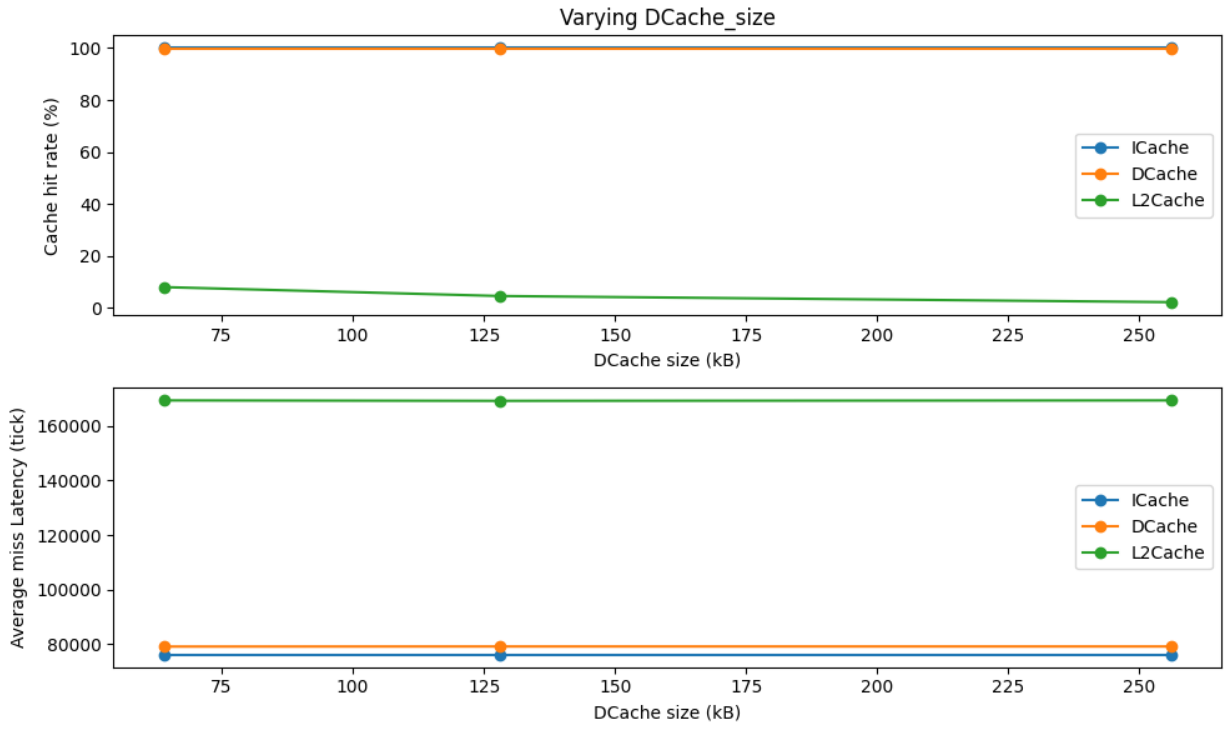


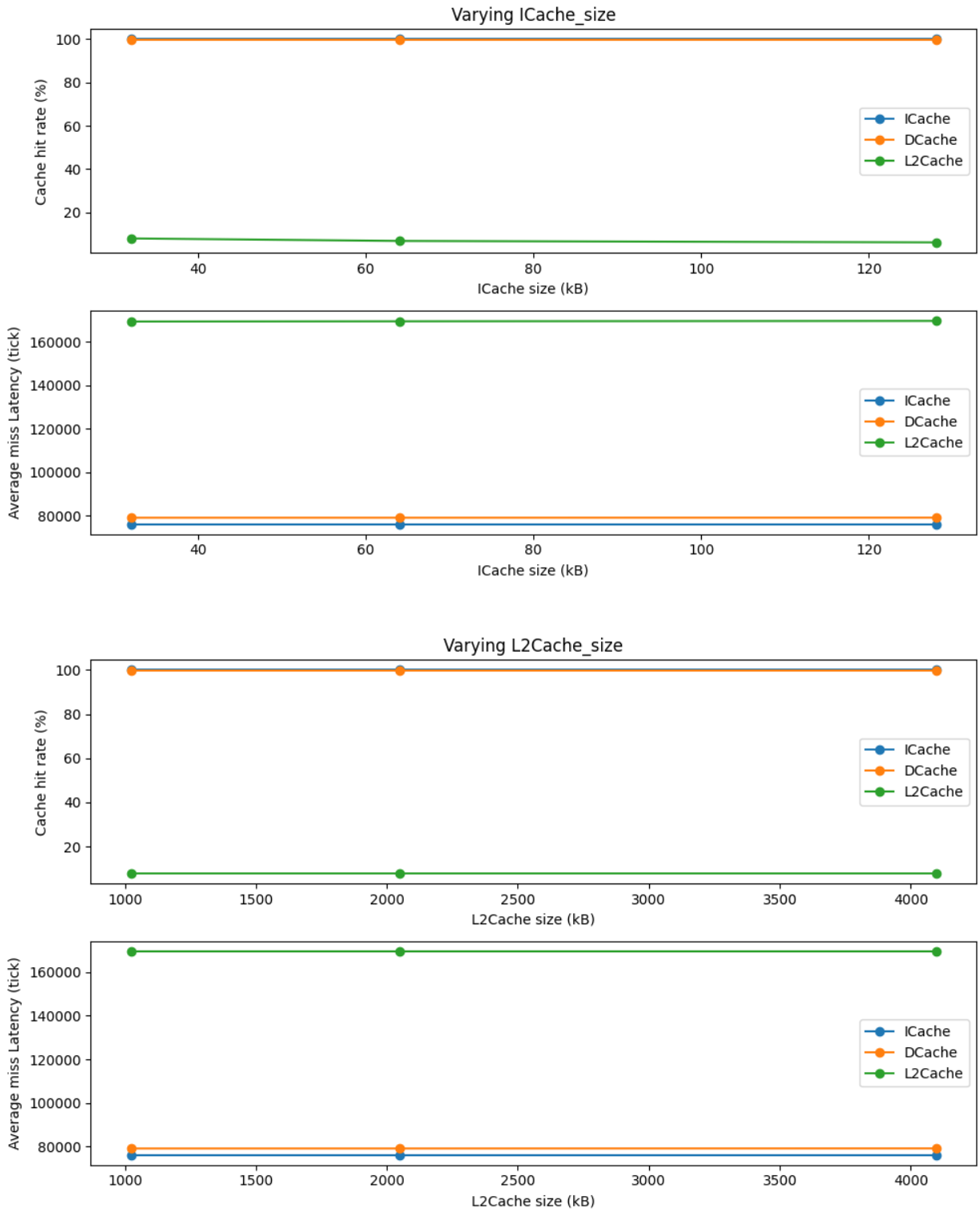
Similar to the separate cache scheme, the unified cache also increased and stabilize after associativity goes up to 2.

Part 4:

Part 4:

Cache	Dcache_Hits	Dcache_Misses	Dcache_AvgMissLatency	Icache_Hits	Icache_Misses	Icache_AvgMissLatency	L2cache_Hits	L2cache_Misses	L2cache_AvgMissLatency	Dcache_Hit_Percentage	Icache_Hit_Percentage	L2cache_Hit_Percentage
Size=(l1i_size=64kB, l1d_size=128kB, l2_size=2MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	299	3468	169,391.58	99.93	99.99	7.94
Size=(l1i_size=64kB, l1d_size=128kB, l2_size=4MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	299	3468	169,391.58	99.93	99.99	7.94
Size=(l1i_size=64kB, l1d_size=256kB, l2_size=2MB)	3312982	2285	79,131.79	8497655	1182	76,206.43	299	3468	169,391.58	99.93	99.99	7.94
Size=(l1i_size=64kB, l1d_size=256kB, l2_size=4MB)	3312982	2285	79,131.79	8497655	1182	76,206.43	164	3468	169,227.80	99.93	99.99	4.52
Size=(l1i_size=128kB, l1d_size=128kB, l2_size=2MB)	3312982	2285	79,097.59	8497653	1184	76,206.43	164	3468	169,227.80	99.93	99.99	4.52
Size=(l1i_size=128kB, l1d_size=128kB, l2_size=4MB)	3312982	2285	79,097.59	8497653	1184	76,206.43	164	3468	169,227.80	99.93	99.99	4.52
Size=(l1i_size=128kB, l1d_size=256kB, l2_size=2MB)	3312982	2285	79,131.79	8497655	1182	76,206.43	76	3468	169,389.85	99.93	99.99	2.14
Size=(l1i_size=128kB, l1d_size=256kB, l2_size=4MB)	3312982	2285	79,131.79	8497655	1182	76,206.43	76	3468	169,389.85	99.93	99.99	2.14
Size=(l1i_size=256kB, l1d_size=128kB, l2_size=2MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	76	3468	169,389.85	99.93	99.99	2.14
Size=(l1i_size=256kB, l1d_size=128kB, l2_size=4MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	252	3468	169,488.75	99.93	99.99	6.77
Size=(l1i_size=256kB, l1d_size=256kB, l2_size=2MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	252	3468	169,488.75	99.93	99.99	6.77
Size=(l1i_size=256kB, l1d_size=256kB, l2_size=4MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	252	3468	169,488.75	99.93	99.99	6.77
Size=(l1i_size=64kB, l1d_size=64kB, l2_size=2MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	117	3468	169,313.44	99.93	99.99	3.26
Size=(l1i_size=64kB, l1d_size=64kB, l2_size=4MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	117	3468	169,313.44	99.93	99.99	3.26
Size=(l1i_size=64kB, l1d_size=32kB, l2_size=2MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	117	3468	169,313.44	99.93	99.99	3.26
Size=(l1i_size=64kB, l1d_size=32kB, l2_size=4MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	29	3468	169,628.60	99.93	99.99	0.83
Size=(l1i_size=32kB, l1d_size=64kB, l2_size=2MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	29	3468	169,628.60	99.93	99.99	0.83
Size=(l1i_size=32kB, l1d_size=64kB, l2_size=4MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	29	3468	169,628.60	99.93	99.99	0.83
Size=(l1i_size=32kB, l1d_size=32kB, l2_size=2MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	226	3468	169,666.67	99.93	99.99	6.12
Size=(l1i_size=32kB, l1d_size=32kB, l2_size=4MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	226	3468	169,666.67	99.93	99.99	6.12
Size=(l1i_size=128kB, l1d_size=64kB, l2_size=2MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	226	3468	169,666.67	99.93	99.99	6.12
Size=(l1i_size=128kB, l1d_size=64kB, l2_size=4MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	91	3468	169,117.94	99.93	99.99	2.56
Size=(l1i_size=128kB, l1d_size=32kB, l2_size=2MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	91	3468	169,117.94	99.93	99.99	2.56
Size=(l1i_size=128kB, l1d_size=32kB, l2_size=4MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	91	3468	169,117.94	99.93	99.99	2.56
Size=(l1i_size=256kB, l1d_size=64kB, l2_size=2MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	3	3468	169,158.59	99.93	99.99	0.09
Size=(l1i_size=256kB, l1d_size=64kB, l2_size=4MB)	3312982	2285	79,097.59	8497655	1182	76,206.43	3	3468	169,158.59	99.93	99.99	0.09
Size=(l1i_size=256kB, l1d_size=32kB, l2_size=2MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	3	3468	169,158.59	99.93	99.99	0.09
Size=(l1i_size=256kB, l1d_size=32kB, l2_size=4MB)	3312982	2285	79,131.79	8497653	1184	76,206.43	3	3468	169,158.59	99.93	99.99	0.09





Regardless of which cache size we varied in the simulation of part 4, the hit rate and the average miss latency remains unchanged. The miss rate of L2 is significantly lower than that of the icache and dcache. The reason is because matmul operation (matrix

multiplication) only performs instructions on the same set of input repeatedly. So, after the first few cache miss, then the CPU doesn't need to go to L2 to fetch the instruction and data anymore. It will continuously operate on the instruction in the L1 because it has the same input which leads to very high hit rate in L1. L2 then suffer from high miss rate since only the first few requests which are misses will get down to L2 and afterwards barely any request will make it to L2.

Note:
In our zip file, we provided the following codes:

1. Code to generate the table-------> gem5_result_table.py
2. Code to generate the plots ----------> generate_plots.py
3. Gem5 python config script ---------- simple_cache.py and cache_working.py
4. Bash script to run gem5 sim---------> cache_sim.sh

Instruction to run the code :

1. Build gem5.opt either on docker or on virtual machine.
2. Extract the zip to proj2 folder.
3. Relocate proj2 folder to the following path **/gem5/src/learning-gem5/**
4. Run cache_sim.sh to generate the four output files for each part (1-4) called filtered_output_p(1-4).txt. The output files are already provided when doing a matmul operation with the imput r1=r2=c1=c2=50.
5. Run gem5_result_table.py to generate table.
6. Run generate_plots.py to generate plots.