Assigment 3 - Is it on cache?

Marco Espinoza
Jose Campos

Instituto Tecnolgico de Costa Rica

1 First reference misses

The program main.c was executed 4 times using the program perf stat to track the number of cache references and cache misses during our main program execution. The results obtained are the following:

Table 1. Perf stat results after 4 main program executions

#	Time elapsed (ms)	Task-clock (msec)	cycles	instructions	cache-references	cache-misses
1	10,909	0,267	630722	423164	22265	10477
2	0,446	0,202	595425	420416	22047	5813
3	0,389	0,192	529091	426025	23513	2800
4	0,487	0,223	517475	425037	21689	1907

1.1 Questions

a. Why did the first run of your application report a high cache miss rate?
Reason of this is because during the first execution the program is stored in the hard drive, and it is not loaded in the main memory or in the L2 cache. It produces the core must to move the data from the drive to the cache, and it produces misses during the program execution.

b. Why did any subsequent execution report a decremental cache miss rate?
Because after the first execution, part of the program is stored in the L2 cache, and it produces hits during the second or the next program executions, and as the program is being executed multiple times, due to the locality of reference, the number of misses is reduced.
Those locality of reference consists on temporal and spacial locality, where the temporal locality refers to the reuse of specific data, and/or resources, within a relatively small time duration. Spatial locality refers to the use of data elements within relatively close storage locations memory.
For those reasons the number of cache miss rate is reduced after the subsequent program execution.

2 Optimizing memory accesses on loops

On this section, the main application located into the part_2 directory of the project files was used to be compiled through 2 kind of memory ordering, row and columns. The program was executed using both methods 5 times using perf stat to compare both results in terms of cache memory access. The results will be exposed in the following sections.

2.1 Row results

The results obtained for the row results are presented in the next table.

Table 2. Perf stat results using row ordering during 5 times

#	Time elapsed (ms)	Task-clock (msec)	cycles	instructions	cache-references	cache-misses
1	134.57	4.583	11628737	24742637	202382	121747
2	4.174	3.894	10964320	24591759	150668	59725
3	7.578	7.061	11608162	24534858	153962	86481
4	27.237	4.173	11261520	24515829	148946	77911
5	15.436	3.953	11056268	24542301	155608	76989

2.2 Column results

The results obtained for the column results are presented in the table 3.

Table 3. Perf stat results using row ordering during 5 times

#	Time elapsed (ms)	Task-clock (msec)	cycles	instructions	cache-references	cache-misses
1	57.02	5.117	13173982	24687156	258039	122711
2	4.549	4.25	12371551	24591174	206147	59734
3	35.83	4.42	12422663	24580902	212715	79864
4	17.95	4.803	13089132	24571754	237982	85326
5	21.60	4.45	12696407	24502248	232095	78962

2.3 Questions

a. What kind of optimization is performed on the program?
The kind of optimization performed on the program is called permutation. Loop permutation takes advantage of the memory ordering used by an specific programming language for data allocation improving data locality. The followed image present both examples used on this section, row and column ordering.

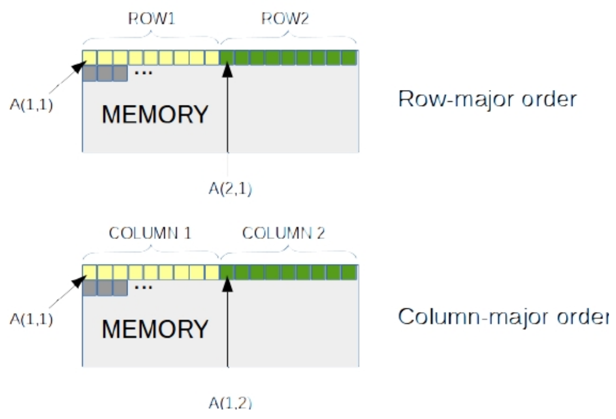

Fig. 1. Figure1

b. Why does the cache miss rate is higher for the row results? Is this expected according to the loop optimizations implemented? Provide detail on your thoughts.

Because with the row implementation, during each iteration, the program is accessing the next cache line, increasing the probability to have misses, because when the next cache line is accessed, it is possible the old cache line could be replaced for a new data because the program will continue moving trough the cache lines. As result, when a data which is stored in the old cache lines is required, it could be possible the data is not in the L2, and the core will require to wait until the data is again in the L2, producing a higher miss rate compared with the column ordering
The next table presents the miss-rate for both orderings.

Table 4. Row miss rate vs Column miss rate

#	Time elapsed (ms)	Task-clock (msec)	cycles	instructions	cache-references	cache-misses
1	10,909	0,267	630722	423164	22265	10477
2	0,446	0,202	595425	420416	22047	5813
3	0,389	0,192	529091	426025	23513	2800
4	0,487	0,223	517475	425037	21689	1907

c. Why do the cache references are higher for the column results? Provide detail on your thoughts.
The cache references is higher for the column results due to during each iteration, the same cache line is accessed, and it helps to increase the number of hits due to the special locality. Because the program is accessing addresses belonging to the same cache line, and it increments the probability to continue accessing addresses close to the current address. The following table shows the results for the cache references obtained in both methods.

Table 5. Column cache references between Row cache references

#	Column Cache-References	Row Cache References
1	258039	202382
2	206147	150668
3	212715	153962
4	237982	148946
5	232095	155608

d. Why do the cache misses are similar between both tests? Is this expected according to the loop optimization method implemented? Provide detail on your thoughts.
e. How is the loop optimization related to the instructions per cycle reported for each test case (insns per cycle) ?