Assignment 3: Is it on cache?

Antonio Aguilar Bravo, Brayan Alfaro Cerdas

April 25, 2017

1 First reference misses

1.1 Why did the first run of your application report a high cache miss rate?

Because no data of the application is found in the cache the first time, the application "main" is only stored in the hard disk. When the application is launched the system move the application data ("code") from the hard disk to the cache.

1.2 Why did any subsequent execution report a decremental cache miss rate?

Because more data of the application are stored into the cache. The system does not remove the data stored into the cache from the first run because expects that those data will be used soon, this is called the temporal locality principle.

2 Optimizing memory accesses on loops

2.1 What kind of optimization is performed on the program?

The kind of optimization is loop permutation, which takes advantage of the memory ordering used by an specific programming language for data allocation improving data locality.

2.2 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.

When the system brings data from the disk into the cache it does by little chuck of cache lines, typically 64 bytes, so when 1 byte needs to be accessed, it will be move into the cache together with others 63 consecutive bytes. The consecutive bytes are stored according to the language, "C" uses row major schema, that means the outer loop has to be for row iterations to take advantage of spacial locality principle. According to this, we would expect to have lower cache miss rate, however this is not the case. Probably there is something in the cache design like mid-level cache (MLC) and last-level cache (LLC) that we need to take into account. First we need to understand if the *perf stat* program used to get the results take the data from MLC or LLC. The MLC is nearer to the CPU so it is faster and using the spacial locality principle we would get more cache hits, So I would say that the *perf stat* program only take the data from the LLC, because it seems that it is not seeing these cache hits.

2.3 Why do the cache references are higher for the *column* results? Provide detail on your thoughts.

We see that the spacial locality principle does not only depends on the kind of scheme the language uses to store data but also depends on how the cache stores those data, the system has the MLC which is smaller that the LLC, and we expect more misses in the MLC using the *column* results, because the requested data are very apart one from other, but this cause the cache lines be moved from the MLC to the LLC, so we get more misses in the MLC but more hits in the LLC, and probably the *perf stat* program only see the hits and misses of the LLC.

2.4 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.

According to the loop optimization method implemented we expect more cache misses in the column-major, but this is true thinking about the MLC level, but as the application takes the results from the LLC, the cache misses could be similar in both methods.

2.5 How is the loop optimization related to the instructions per cycle reported for each test case (*insns per cycle*) ?

When the optimization Row-Major is used we get higher *insns per cycle*, this is because as we mention earlier, more data are store in the MLC which is a cache level nearer to the CPU and faster.