Katherine Perry
CPEG 455
09/11/2019

Lab 0: Hello World

CPU CONFIGURATION: Intel Core i7-7500U 2.7Ghz
OS: Windows 10-64bit
Compiler: IDE Eclipse, C/C++ for developers

First version of Code:

Originally the code of version one ran at a time of 148837.50 microseconds. While changing the order of the loop(the first for loop being "j"), this increases runtime to 89069.0000 microseconds. The reasoning for this is because initializing "j" before "i" will increase the initial amount of cache misses. This is due to the order of the array, that being a[i,j]; if the order of the array was a[j,i] this change of code would decrease runtime. By increasing cache miss, spatial locality is ignored. Instead of producing the next most likely position of the array to hit, it jumps to a position that is more likely to miss due to the order of the array.

Second version of code:

Version two of the sequential access had a runtime of 178198.0 microseconds. After manipulating the code so that the "j" for loop is right before the "i'' for loop, the runtimes increases to 381251.10 microcesonds. In the first version of the nested for loop, it only contains three for loops. After manipulating it to increase cache misses, the runtime increased more than it did in the second version. The second version of the for loop has one more for loop that iterates "i" using stride. Since the order order of the array is a[i,j], it makes sense that changing the location of the "j' for loop would increase cache miss. However the cache miss must not be as high as it was in the first version of the code. The reason why cache miss is less is due to the extra for loop. To get the greatest number of cache hits for this particular code, "i' for loop should be iterated first. In this version of manipulated code, "i'" is dependent upon the variable "stride." Since stride is still the first for loop, the order of a[i,j] is still somewhat maintained. By this I mean the amount of cache hits should be higher because sequential locality is not completely ignored, unlike in the first version of the code.

Third version of code:

In the third version of the code, the original runtime is 169372.0 microseconds. After adjusting the for loop so that "j" is initialized first, the runtime increases to 90303.500000 microseconds. This change of code is similar to version one in which "j' is initialized first, however there is one less for loop. The increase in runtimes are similar, so the cache miss/ cache hit ratio could be similar. The cache miss increasing is due to ignoring sequential locality, however does initializing stride after "j" have an effect on the cache hit/miss ratio? My guess is not enough for it to matter. The effect is negligible given by the differences in runtimes. The initialization of "stride" and "i" can be combined into one line, in result stride does not have a measurable effect on the ratio as long as it is initialized before "i."