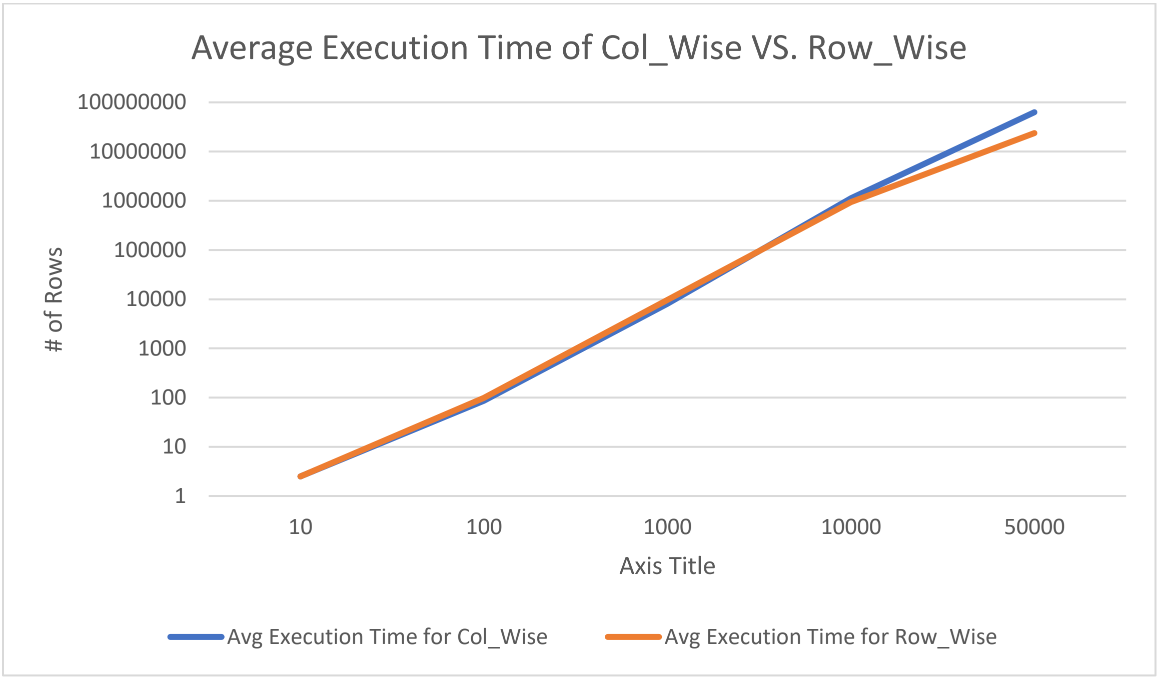
# of rows	Average execution time for col_wise (micro seconds)	Average execution time for row_wise (micro seconds)
10	2.5 ms	2.5 ms
100	86.1 ms	98.4 ms
1000	8021.1 ms	9570 ms
10000	1120709.3 ms	943930.9 ms
50000	62991377.3 ms	23712575 ms



Part 1 Questions:

1. Spatial locality can refer to the property of data where when a specific memory location is referenced, then it will be more likely to access the nearby memory locations in the future. The number of memory accesses can be reduced in a program by utilizing spatial locality, optimizing its performance.
2. The Row-Wise program should perform faster because the memory locations accessed are next to the memory locations that will be accessed in the future. This is because 2D arrays are stored in row-major order in memory. When the program accesses an element in a row, it is likely to access the next element in that row later.
3. Initially the column-wise program had a shorter average execution time than the row-wise program. However, when the array size was increased significantly past 100 and 1000 to 10000 and 50000, the average execution time for the column-wise program was significantly higher than the row-wise program.

4. Yes, to an extent since the average execution time of the column-wise was initially shorter but as the array size increased greatly the row-wise program was faster.
5. With smaller array size the execution time can vary greatly and may not be an accurate representation of how the program will do in exponential array sizes. How running times are often measured are worst case scenario, or with the idea in mind that it will take in an enormous array size (to fully understand the efficiency of the program).
6. As the array size increased the execution times started to vary greatly. If we executed each implementation once and just recorded that time, it would very inaccurate as it could have been an extreme outlier (high or low) compared to the rest of the 9 times we would have run the program. Taking the average eliminates outliers and gives us an execution time where we can predict it will land near for majority of the time.

Part 2 Questions:

1. Sizes
   a. L1d cache: 768 KiB
   b. L1i cache: 768 KiB (L1 cache likely both combined: 1.536 MiB)
   c. L2 cache: 12 MiB
   d. L3 cache: 192 MiB
2. 515388 MiB
3. Yes, the size of RAM and cache influence execution time. This is because the amount of usable RAM determines how much data is stored in memory at a given time, and the size of the cache determines how much data can be stored in the memory that is faster and closer to the CPU. When there is no more available RAM the system may resort to larger but slower memory sources. The same concept applies to the cache, where if there is not enough cache the CPU will access data from a slower memory source.

	Col_wise execution		Row_wise execution	
# of rows	Cache hits	Cache misses	Cache hits	Cache misses
10	56789	2351	56787	3251
100	243580	4293	243304	4573
1000	17927657	1126825	18926161	128325
10000	77518540	112483325	177498540	12503325
50000	224734662	22034176 (reached segmentation fault)	224734662	22034176 (reached segmentation fault)

Part 3 Questions:

1. I tried to use perf stat ./rw but it said I needed to download packages which it wouldn't allow me to while in silo. So I tried to use valgrind –tool=cachegrind ./rw instead and gave me some information on cache misses and refs (hits = refs – misses).

2. A cache hit indicates the CPU has requested data that is already stored in the cache and the CPU can retrieve the data from there instead of getting it from a slower memory source, reducing the time it takes to access data. A cache miss indicates CPU has requested data that is not currently stored in the cache, forcing it to get data from slower memory, increase the time it takes to access data.
3. Spatial locality and cache hits and misses determine a programs efficiency. Good spatial locality will be more likely to generate more cache hits and less cache misses, because the program will be more likely to access nearby memory locations of the memory locations it has already accessed. Poor spatial locality will be more likely to generate cache misses and less cache hits because the data required to be access will be scattered throughout multiple memory sources, making it difficult to retrieve and store efficiently.