Lab Report

ECPE 170 – Computer Systems and Networks – Fall 2021

Name: Dustin Schuette

Lab Topic: Performance Optimization (Memory Hierarchy) (Lab #: 07)

(1) Describe how a two-dimensional array is stored in one-dimensional computer memory.
if we think of a 2d array as a series of rows stacked on top of each other i.e. :

----
then we simply lay them side by side making one long row:
------------------------ and now the data is actually one dimensional.
(2) Describe how a three-dimensional array is stored in one-dimensional computer memory.
very similar to the example above instead each element of the row i.e. one dash is its own row we simply flatten it in a similar way where we start at the top row list all the elements of the fist row's dash then the next dash until they are all now one long line and continue to the next doing the same thing.

(3) Copy and paste the output of your program into your lab report, and be sure that the source code and Makefile is included in your Git repository.

2d array as a block
1 |2 |3 |4 |5 |
6 |7 |8 |9 |10|
11|12|13|14|15|
2d array as one line
1 |2 |3 |4 |5 |6 |7 |8 |9 |10|11|12|13|14|15|
3d array as a block
each block is a flat layer, think 3 layer cake
1 |2 |3 |4 |5 |6 |7 |
8 |9 |10 |11 |12 |13 |14 |
15 |16 |17 |18 |19 |20 |21 |
22 |23 |24 |25 |26 |27 |28 |
29 |30 |31 |32 |33 |34 |35 |

36 |37 |38 |39 |40 |41 |42 |
43 |44 |45 |46 |47 |48 |49 |
50 |51 |52 |53 |54 |55 |56 |
57 |58 |59 |60 |61 |62 |63 |
64 |65 |66 |67 |68 |69 |70 |

71 |72 |73 |74 |75 |76 |77 |
78 |79 |80 |81 |82 |83 |84 |
85 |86 |87 |88 |89 |90 |91 |
92 |93 |94 |95 |96 |97 |98 |
99 |100 |101 |102 |103 |104 |105 |

3d array as one line
1 |2 |3 |4 |5 |6 |7 |8 |9 |10 |11 |12 |13 |14 |15 |16 |17 |18 |19 |20 |21 |22 |23 |24 |25 |26 |27 |28 |29 |30
|31 |32 |33 |34 |35 |36 |37 |38 |39 |40 |41 |42 |43 |44 |45 |46 |47 |48 |49 |50 |51 |52 |53 |54 |55 |56 |57 |58
|59 |60 |61 |62 |63 |64 |65 |66 |67 |68 |69 |70 |71 |72 |73 |74 |75 |76 |77 |78 |79 |80 |81 |82 |83 |84 |85 |86
|87 |88 |89 |90 |91 |92 |93 |94 |95 |96 |97 |98 |99 |100 |101 |102 |103 |104 |105 |

(4) Provide an Access Pattern table for the sumarrayrows() function assuming ROWS=2 and COLS=3. The table should be sorted by ascending memory addresses, not by program access order.

from right to left, top to bottom the data is accessed as follows given that 0 is the first space in memory and 5 is the last space in memory.

0	1	2
3	4	5

(5) Does sumarrayrows() have good temporal or spatial locality?
For your answer to receive full credit, you must discuss the locality of both the array itself, and the scalar variables such as i that are present in the function.

In terms of Temporal locality I would say it's average as some variables (sum, i, j) are used several times and thus benefit from being in the cache, the array elements themselves though do not as they are only used once.

The spatial locality is very good and scalable as the array elements are accessed in the same order they appear in the memory address which means many hits and minimum misses.
the non array elements may or may not be in order we don't know so we should err on side of caution and say they probably do not benefit from spatial locality.

(6) Provide an Access Pattern table for the sumarraycols() function assuming ROWS=2 and COLS=3. The table should be sorted by ascending memory addresses, not by program access order.

from right to left, top to bottom the data is accessed as follows given that 0 is the first space in memory and 5 is the last space in memory.

0	3
1	4
2	5

(7) Does sumarraycols() have good temporal or spatial locality?
For your answer to receive full credit, you must discuss the locality of both the array itself, and the scalar variables such as i that are present in the function.

The answer is not simply yes or no, for spatial locality depending on how big one line of the cache memory it may not matter what order this small array is accessed as it may just grab the whole thing when grabbing the 0th item; However this is not scalable in a larger array the spatial locality would get much much worse so spatial locality for this style of access is not recommended.

Temporal locality is also not very good, while sun, i and j are accessed frequently and benefit from being in the cache, the array elements do not as they are only used once so I'd say the temporal locality here is alright.

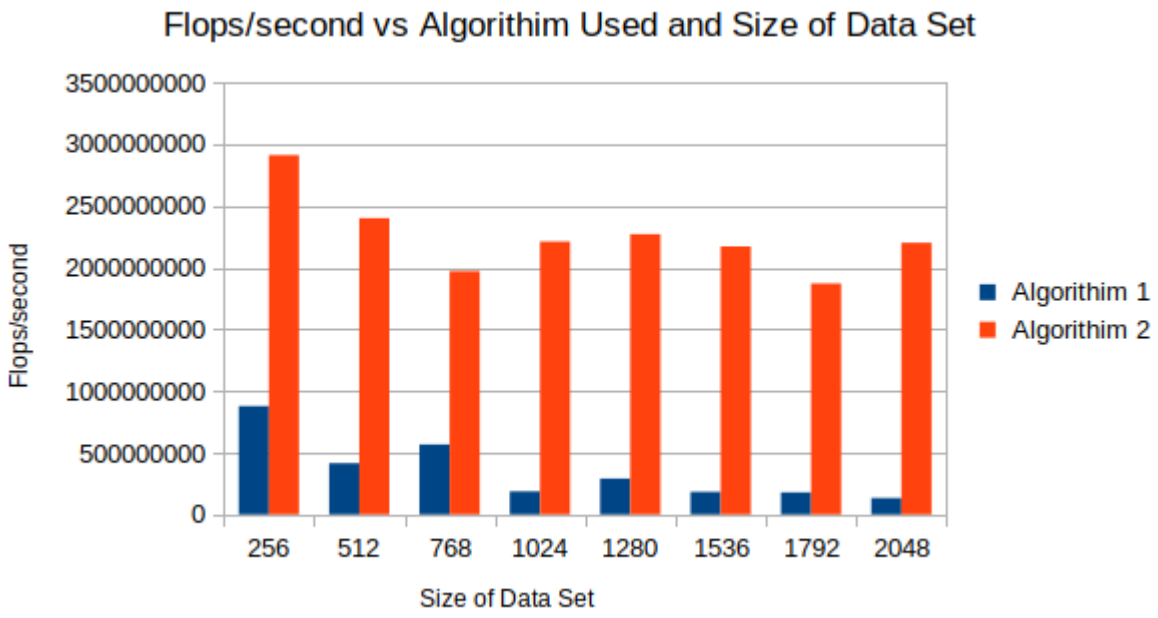
Extra Credit Option: Extend your script to automatically parse the output of the program, capture the number of FLOPS, and save it into a text file suitable for import into LibreOffice or other graphing program.
Value: +20 points on lab score

run.sh is extra credit script
(8) Inspect the provided source code. Describe how the two-dimensional arrays are stored in memory, since the code only has one-dimensional array accesses like: a[element #].
The arrays are flattened and so in order to access say the 15th element of an 4 x 8 array you'd use a[14]; this is achieved by the following code: a[n *i + j] where j is would be the innermost for loop and i is outermost for loop and n is the size of one row of the array.

(9) After running your experiment script, create a table that shows floating point operations per second for both algorithms at the array sizes listed in Table 2.

Size	Algorithim 1	Algorithim 2
256	877000000	2910000000
512	414000000	2400000000
768	566000000	1970000000
1024	186000000	2210000000
1280	290000000	2270000000
1536	182000000	2170000000
1792	178000000	1870000000
2048	132000000	2200000000

(10) After running your experiment script, create a graph that shows floating point operations per second for both algorithms at the array sizes listed in Table 2.



(12) Place the output of /proc/cpuinfo in your report. (I only need to see one processor core, not all the cores as reported)

processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 61
model name : Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
stepping : 4
microcode : 0x2f
cpu MHz : 1744.389
cache size : 3072 KB
physical id : 0
siblings : 4
core id : 0
cpu cores : 2
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 20
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx smap intel_pt xsaveopt dtherm ida arat pln pts md_clear flush_l1d
vmx flags : vnmi preemption_timer invvpid ept_x_only ept_ad ept_1gb flexpriority tsc_offset vtpr

mtf vapic ept vpid unrestricted_guest ple shadow_vmcs
bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa itlb_multihit srbds
bogomips : 4589.28
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical, 48 bits virtual
power management:

(13) Based on the processor type reported, obtain the following specifications for your CPU from cpu-world.com or cpudb.stanford.edu
You might have to settle for a close processor from the same family. Make sure the frequency and L3 cache size match the results from /proc/cpuinfo!

(a) L1 instruction cache size
2 x 32 KB 8-way set associative instruction caches

(b) L1 data cache size
2 x 32 KB 8-way set associative data caches

(c) L2 cache size
2 x 256 KB 8-way set associative caches

(d) L3 cache size
3 MB 12-way set associative shared cache

(e) What URL did you obtain the above specifications from?
https://www.cpu-world.com/CPUs/Core_i5/Intel-Core%20i5-5300U%20Mobile%20processor.html

(14) Why is it important to run the test program on an idle computer system?
Explain what would happen if the computer was running several other active programs in the background at the same time, and how that would impact the test results.
IN order to get a good read of performance we do not want our test program to be competing for resources as this would almost surely end with lower performance and thus poorer results. The other main reason is that since we rely on the cache so heavily in this test, if we had another program running it may use the cache and bottleneck our performance.

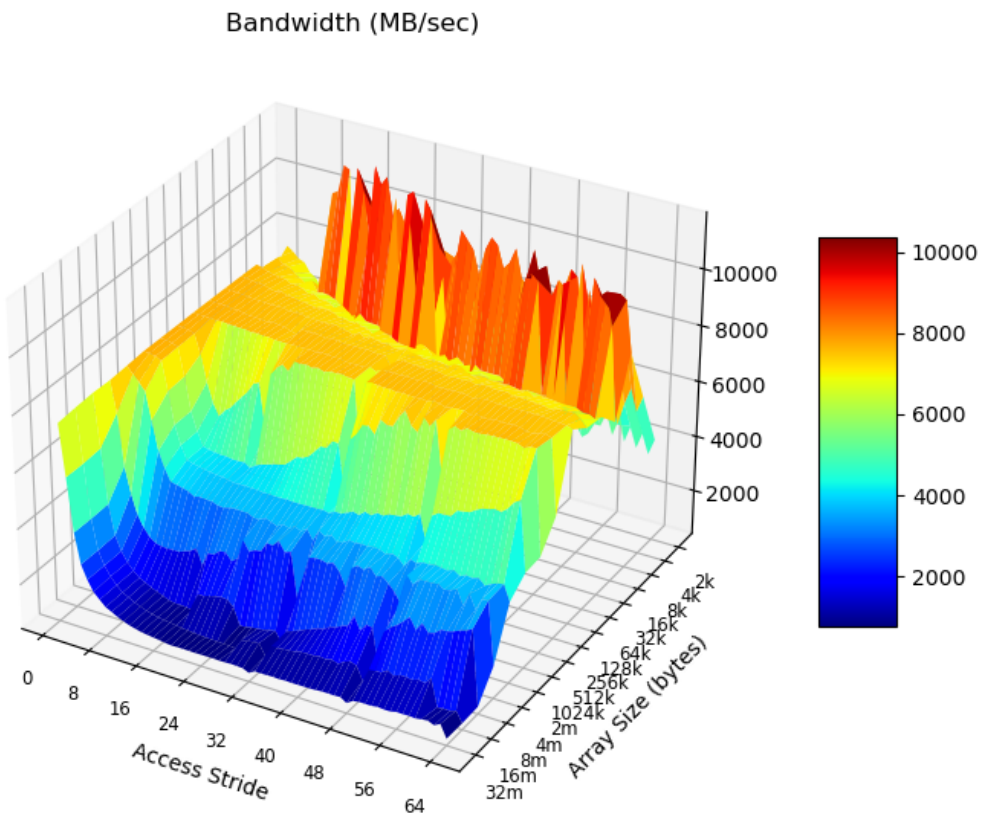
(15) What is the size (in bytes) of a data element read from the array in the test?
8 bytes

(16) What is the range (min, max) of strides used in the test?
0 to 64

(17) What is the range (min, max) of array sizes used in the test?
2Kb to 32m

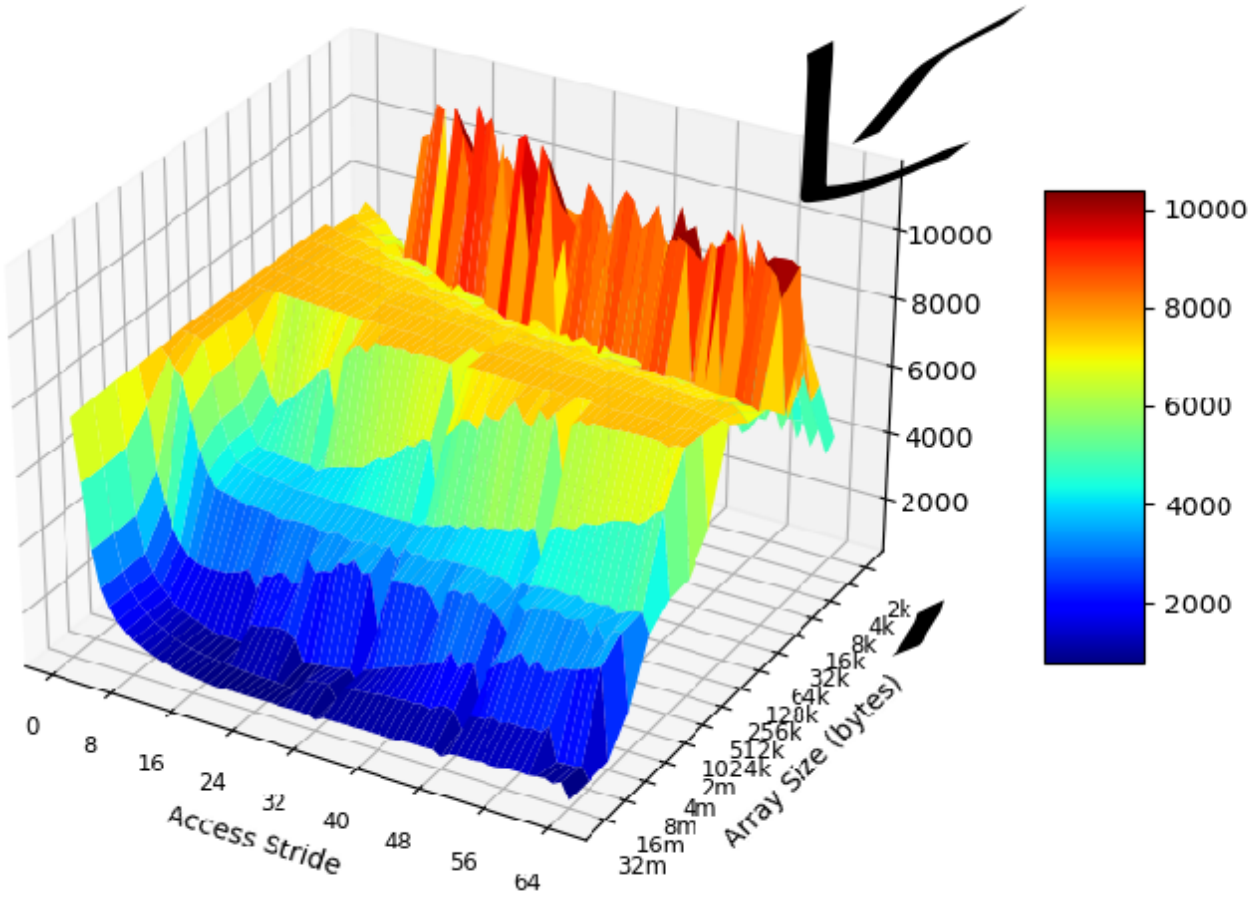
(18) Take a screen capture of the displayed "memory mountain" (maximize the window so it's

sufficiently large to read), and place the resulting image in your report



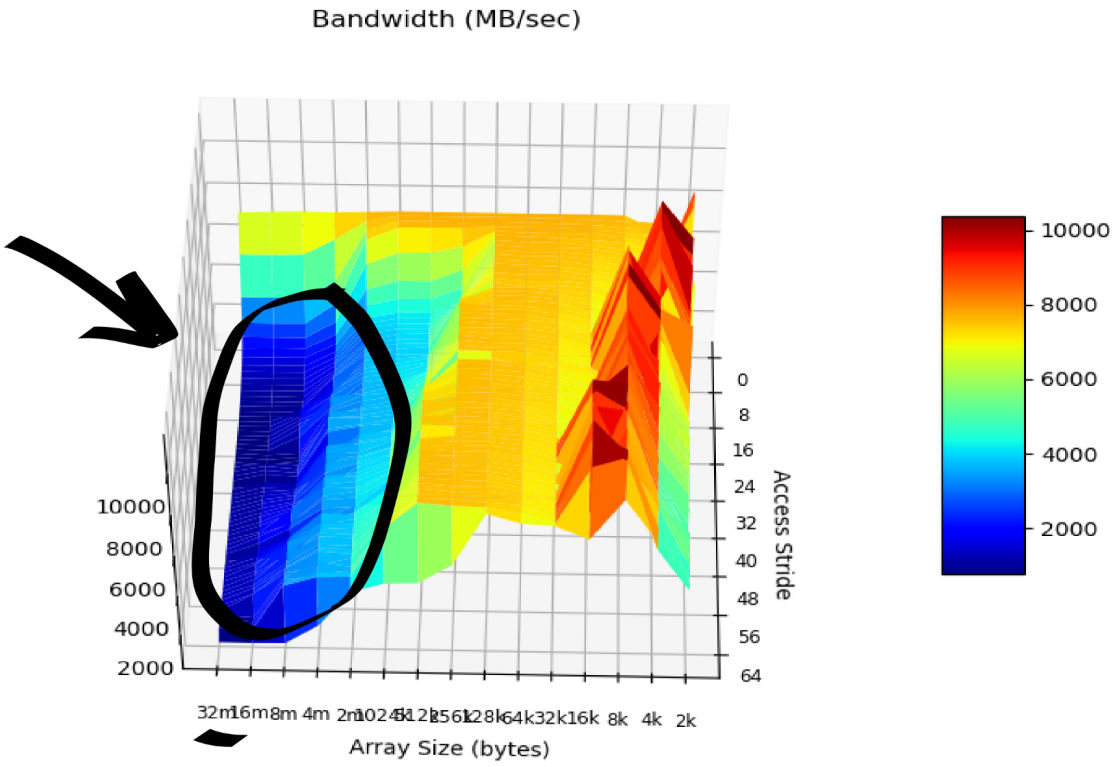
(19) What region (array size, stride) gets the most consistently high performance? (Ignoring spikes in the graph that are noisy results...) What is the read bandwidth reported? Annotate your figure by drawing an arrow on it.

A low Array size has the best performance while as the size gets larger the lower the stride gives better performance. This area is getting over 10000 MB/s in read speed.

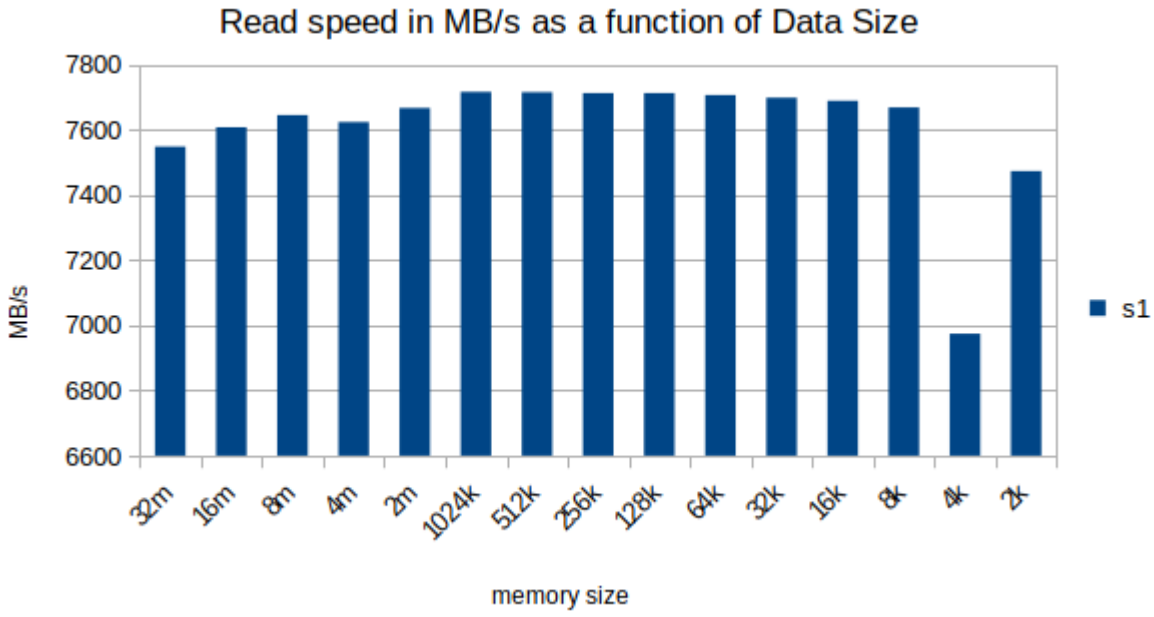


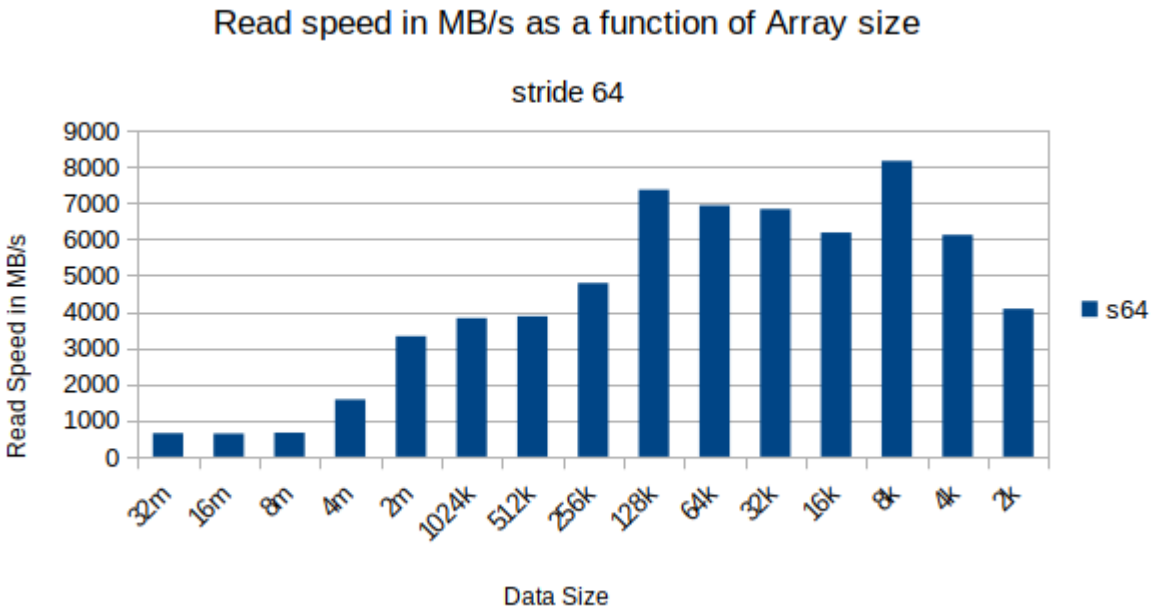
(20) What region (array size, stride) gets the most consistently low performance? (Ignoring spikes in the graph that are noisy results...) What is the read bandwidth reported? Annotate your figure by drawing an arrow on it.

The lowest performance area on the graph was large arrays with higher stride values. The bandwidth is about 500 MB/s



(21) Using LibreOffice calc, create two new bar graphs: One for stride=1, and the other for stride=64. Place them side-by-size in the report.
No credit will be given for sloppy graphs that lack X and Y axis labels and a title.
You can obtain the raw data from results.txt. Open it in gedit and turn off Text Wrapping in the preferences. (Otherwise, the columns will be a mess)





(22) When you look at the graph for stride=1, you (should) see relatively high performance compared to stride=64. This is true even for large array sizes that are much larger than the L3 cache size reported in /proc/cpuinfo.
How is this possible, when the array cannot possibly all fit into the cache? Your explanation should include a brief overview of hardware prefetching as it applies to caches.
Since stride 1 is accessing array elements sequentially it benefits from spatial locality and on a hardware level precaching in which more than just the needed element is grabbed, for example if we need a 2 byte piece of data but one line of cache memory is 16 bytes the computer will grab 8 items instead of just the one as it takes the same amount of work, this means for something like arrays where they are written sequentially the computer just grabbed the next 7 elements it needs whereas stride64 would be much less likely to benefit form this as it's unlikely one line of cache can hold 64 elements and even if it could that would only be one more hit and when it looks for the 3 element it is even more unlikely to be in cache meaning the computer has to fetch it slowing down the while program.

(23) What is temporal locality? What is spatial locality?
temporal locality refers to recalling recently used information that still resides in the cache if temporal locality is good the data needed is more likely to be on the cache while using data once or with long periods in between its calls will be less likely to still be in cache and thus temporal locality would be low. Spatial locality on the other hand relies on accessing data that has been written sequentially such as sequential elements in an array, when data is grabbed and placed in the cache the computer will grab not just the data it needs (unless the data is quite large) but the data right after it and so when the next data is needed it is already in the cache ready to be read.

(24) Adjusting the total array size impacts temporal locality - why? Will an increased array size increase or decrease temporal locality?
As the size of the array increases it outgrows the amount of cache available and thus no longer benefits from temporal locality as when it is called again it the cache has already been overwritten, decreasing performance.

(25) Adjusting the read stride impacts spatial locality - why? Will an increased read stride increase or decrease spatial locality?
increasing the stride means more misses for the program and more calls to find the array item needed, and so increasing the stride size will decrease the spatial locality as it's increasingly less likely the needed array element will already be in cache from the previous call as when the stride increases the distance between elements needed to expand.

(26) As a software designer, describe at least 2 broad "guidelines" to ensure your programs run in the high-performing region of the graph instead of the low-performing region.

(1) Access elements in order when possible i.e. if you need all the members of an array, access them sequentially.
(2) If you need to use something more than once try to call on it again as soon as possible as it will benefit from temporal locality.