Lab Report

ECPE 170 – Computer Systems and Networks – Spring 2016

Name: Nicolas Ahn

Lab Topic: Performance Measurement (Lab #: 4)

Question #1 Describe how a two-dimensional array is stored in one-dimensional computer memory.

Answer: Two-dimensional arrays are stored continuously in one-dimensional computer memory.

Question #2 Describe how a three-dimensional array is stored in one-dimensional computer memory.

Answer: Three-dimensional arrays are stored continuously in one-dimensional computer memory.

Question #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 compressed folder to submit.

Answer:

```
-- Memory Access of 2D Array in two different options:

 - Option 1: Start at row=0, column=0. Step through each row until the end, then advance to the next column.

 - Option 2: Start at row=0, column=0. Step through each column until the end, then advance to the next row.


Memory addresses of Option 1:

[0][0] = 51c98350

[0][1] = 51c98354

[1][0] = 51c98358

[1][1] = 51c9835c


Memory addresses of Option 2:

[0][0] = 51c98350

[1][0] = 51c98358
```

```
[0][1] = 51c98354
[1][1] = 51c9835c


-- Memory Access of 3D Array:
Memory addresses in order i, j, k:
[0][0][0] = 51c98330
[0][0][1] = 51c98334
[0][1][0] = 51c98338
[0][1][1] = 51c9833c
[1][0][0] = 51c98340
[1][0][1] = 51c98344
[1][1][0] = 51c98348
[1][1][1] = 51c9834c

Memory addresses in order k, i, j:
[0][0][0] = 51c98330
[0][1][0] = 51c98338
[1][0][0] = 51c98340
[1][1][0] = 51c98348
[0][0][1] = 51c98334
[0][1][1] = 51c9833c
[1][0][1] = 51c98344
[1][1][1] = 51c9834c
```

As shown, the memory addresses of 2D and 3D arrays are stored continguously. When we try to print the address of all cells of the array, its address representation increments by 4, which is the byte size of an unsigned int.

As shown, the memory addresses of 2D and 3D arrays are stored continguously. When we try to print the address of all cells of the array, its address representation increments by 4, which is the byte size of an unsigned int.

Question #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.

Memory Address	0	4	8	12	16	20
Memory Contents	a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]
Program Access Order	1	2	3	4	5	6

Question #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.

sumarrayrows() have good spatial locality because elements of the array are read sequentially, but poor temporal locality as the elements of the array are only read once. On the other hand, the variables i, j, and sum have good temporal locality as they are accessed through each loop, but poor spatial locality for being scalar variables.

Question #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.

Memory Address	0	4	8	12	16	20
Memory Contents	a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]
Program Access Order	1	3	5	2	4	6

Question #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.

Using the same array from question 5, we can conclude that sumarraycols() have bad spatial locality because elements of the array are not read sequentially like in sumarrayrows(). It also has poor temporal locality as the elements of the array are only read once. The variables i, j, and sum have good temporal locality as they are accessed through each loop, but poor spatial locality for being scalar variables.

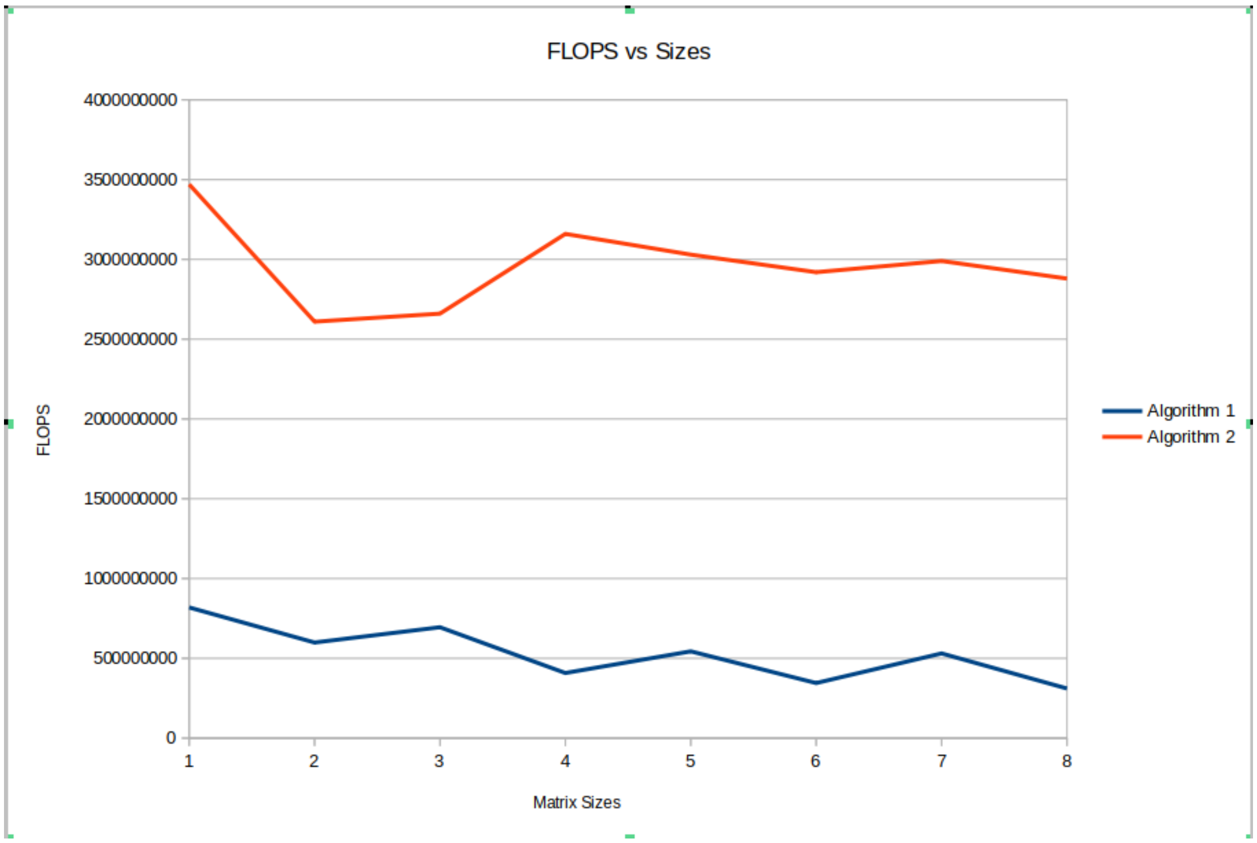
Question #8 Inspect the provided source code. Describe how the two-dimensional arrays are stored in memory, since the code only has one-dimensional array accesses like: a[element #].

Since the array is stored in order by row, it can be divided into the amount of columns and turned into a 2-D array, where each new row is where the columns divide.

Question #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.

Algorithm 1	FLOPS	Algorithm 2	FLOPS
256	8.19E+08	256	3.47E+09
512	5.99E+08	512	2.61E+09
768	6.95E+08	768	2.66E+09
1024	4.08E+08	1024	3.16E+09
1280	5.44E+08	1280	3.03E+09
1536	3.45E+08	1536	2.92E+09
1792	5.31E+08	1792	2.99E+09
2048	3.11E+08	2048	2.88E+09

Question #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.



(Where 1-8 corresponds to matrices of size 256, 512, …, 2048).

Question #11 Be sure that the script source code is included in your compressed folder to submit.

Question #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       : AuthenticAMD

cpu family      : 25

model           : 80

model name      : AMD Ryzen 5 PRO 5650U with Radeon Graphics

stepping        : 0

microcode       : 0xa50000c

cpu MHz                 : 1600.000

cache size      : 512 KB

physical id     : 0

siblings        : 12

core id         : 0

cpu cores       : 6

apicid          : 0

initial apicid  : 0

fpu             : yes

fpu_exception   : yes

cpuid level     : 16

wp              : yes

flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc
rep_good nopl nonstop_tsc cpuid extd_apicid aperfmperf pni pclmulqdq monitor ssse3 fma cx16
sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand lahf_lm cmp_legacy svm extapic
cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce topoext perfctr_core
perfctr_nb bpext perfctr_llc mwaitx cpb cat_l3 cdp_l3 hw_pstate ssbd mba ibrs ibpb stibp
vmmcall fsgsbase bmi1 avx2 smep bmi2 erms invpcid cqm rdt_a rdseed adx smap clflushopt
clwb sha_ni xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total
cqm_mbm_local clzero irperf xsaveerptr rdpru wbnoinvd arat npt lbrv svm_lock nrip_save
tsc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold avic v_vmsave_vmload
vgif v_spec_ctrl umip pku ospke vaes vpclmulqdq rdpid overflow_recov succor smca fsrm
```

```
bugs            : sysret_ss_attrs spectre_v1 spectre_v2 spec_store_bypass

bogomips        : 4591.32

TLB size        : 2560 4K pages

clflush size    : 64

cache_alignment         : 64

address sizes   : 48 bits physical, 48 bits virtual

power management: ts ttp tm hwpstate cpb eff_freq_ro [13] [14]
```

Question #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

(b) L1 data cache size

(c) L2 cache size

(d) L3 cache size

(e) What URL did you obtain the above specifications from?

Answer:

a) 64 KiB
b) 384 KB
c) 3 MB
d) 16 MB
e) https://www.guru3d.com/articles-pages/amd-ryzen-5-1600-review,3.html#:~:text=The%206%2Dcore%20part%20with,of%20512%20KiB%20per%20core.
   https://www.notebookcheck.net/AMD-Ryzen-5-PRO-5650U-Processor-Benchmarks-and-Specs.527811.0.html

Question #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.

Answer: The more background programs running on a computer, the less bandwidth the system has to run the code. This means results will be limited due to the lack of bandwidth.

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

Answer:

8

Question #16 What is the range (min, max) of strides used in the test?

Answer:

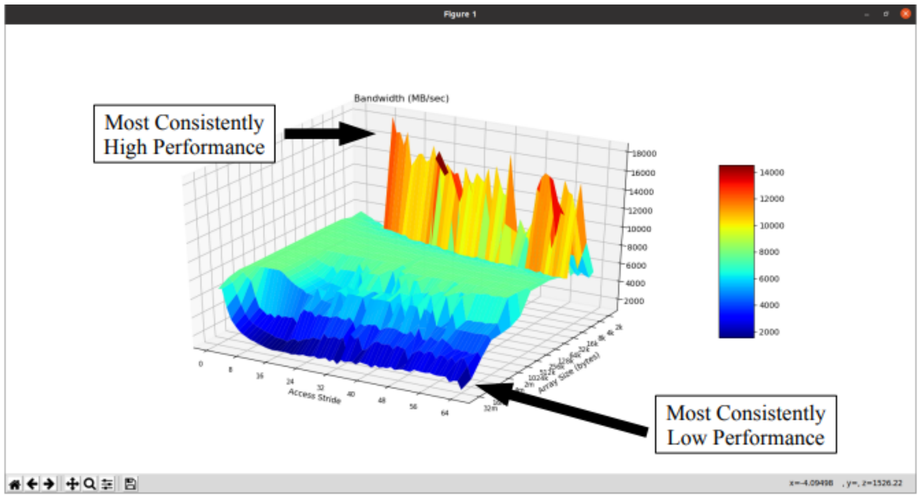
(0-64)

Question #17 What is the range (min, max) of array sizes used in the test?

Answer:

2kB - 32mB

Question #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



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.

Question #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.

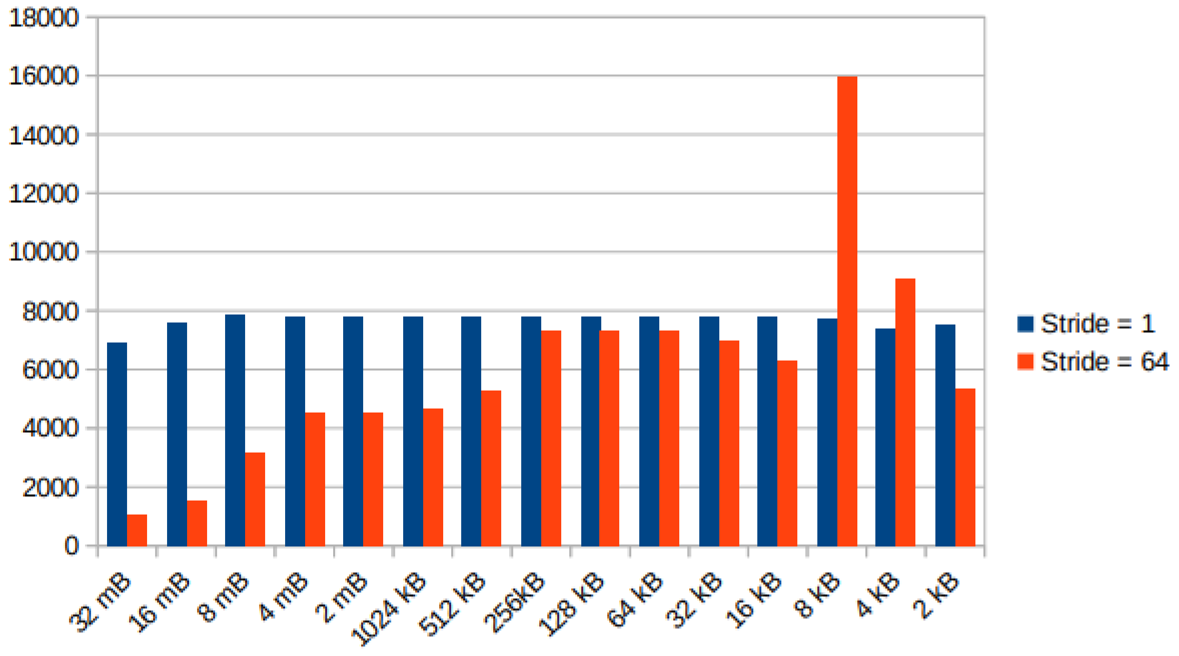
Answer:

most consistently low performance: 32mB

read bandwidth: 2000 MB/sec

Question #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.

Answer:



Question #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.

Answer:

The greater the stride, the better the spatial locality as the memory is stored much closer. In stride = 64, not every element is stored in the cache. Therefore, its spatial locality becomes significantly worse, leading to worse performance.

Question #23: What is temporal locality? What is spatial locality?

Answer:

Temporal locality is memory that is accessed relatively soon.

Spatial locality is memory that is accessed relatively close to other memory addresses.

Question #24: Adjusting the total array size impacts temporal locality - why? Will an increased array size increase or decrease temporal locality?

Answer:

Increasing the array size will negatively affect temporal locality since it will take a longer amount of time before an element will be re-accessed.

Question #25: Adjusting the read stride impacts spatial locality - why? Will an increased read stride increase or decrease spatial locality?

Answer:

Increasing the read stride will decrease spatial locality because the memory being accessed will be further away from each other.

Question #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.

Answer:

Increase spatial locality by accessing as much memory as possible and increase temporal locality by using less data variables.