Lab Report

ECPE 170 – Computer Systems and Networks – Fall 2015

Name: Zhiyun Yan
Lab Topic: Performance Optimization (Memory Hierarchy)

Question(1) Describe how a two-dimensional array is stored in one-dimensional computer memory.
Answer: The computer store the first row and column(Row 0, column 0) in one block, store first row and second
column(Row 0, column 1) in next block. So the first element in 2D Array, arr, arr[0][0] store in address 1, the second arr[0][1] is stored in address 2, etc..

Question(2) Describe how a three-dimensional array is stored in one-dimensional computer memory.
Answer: The first element in 3D Array, arr, arr[0][0][0] store in address 1, the second arr[0][0][1] is stored in address 2, etc..

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 Mercurial repository.
Answer:

```
 For the 2d-array
located at row 1 column 1
array1 address = 0x7ffe1ba451b0

located at row 1 column 2
array1 address = 0x7ffe1ba451b4

located at row 1 column 3
array1 address = 0x7ffe1ba451b8

located at row 1 column 4
array1 address = 0x7ffe1ba451bc

located at row 1 column 5
array1 address = 0x7ffe1ba451c0

located at row 2 column 1
array1 address = 0x7ffe1ba451c4

located at row 2 column 2
array1 address = 0x7ffe1ba451c8

located at row 2 column 3
array1 address = 0x7ffe1ba451cc

located at row 2 column 4
array1 address = 0x7ffe1ba451d0

located at row 2 column 5
array1 address = 0x7ffe1ba451d4

located at row 3 column 1
array1 address = 0x7ffe1ba451d8
```

```
located at row 3 column 2
array1 address = 0x7ffe1ba451dc

located at row 3 column 3
array1 address = 0x7ffe1ba451e0

located at row 3 column 4
array1 address = 0x7ffe1ba451e4

located at row 3 column 5
array1 address = 0x7ffe1ba451e8
```

we can see that every memory address 4 bit difference, which is right, because value type is uint32 which is 4 bit. so answer is option 2
and for the 3d-array.

```
located at  1  1 1
array1 address = 0x7ffe1ba451f0

located at  1  1 2
array1 address = 0x7ffe1ba451f4

located at  1  1 3
array1 address = 0x7ffe1ba451f8

located at  1  1 4
array1 address = 0x7ffe1ba451fc

located at  1  1 5
array1 address = 0x7ffe1ba45200

located at  1  1 6
array1 address = 0x7ffe1ba45204

located at  1  1 7
array1 address = 0x7ffe1ba45208

located at  1  2 1
array1 address = 0x7ffe1ba4520c

located at  1  2 2
array1 address = 0x7ffe1ba45210

located at  1  2 3
array1 address = 0x7ffe1ba45214

located at  1  2 4
array1 address = 0x7ffe1ba45218
```

```
located at  1  2 5
array1 address = 0x7ffe1ba4521c

located at  1  2 6
array1 address = 0x7ffe1ba45220

located at  1  2 7
array1 address = 0x7ffe1ba45224

located at  1  3 1
array1 address = 0x7ffe1ba45228

located at  1  3 2
array1 address = 0x7ffe1ba4522c

located at  1  3 3
array1 address = 0x7ffe1ba45230

located at  1  3 4
array1 address = 0x7ffe1ba45234

located at  1  3 5
array1 address = 0x7ffe1ba45238

located at  1  3 6
array1 address = 0x7ffe1ba4523c

located at  1  3 7
array1 address = 0x7ffe1ba45240

located at  1  4 1
array1 address = 0x7ffe1ba45244

located at  1  4 2
array1 address = 0x7ffe1ba45248

located at  1  4 3
array1 address = 0x7ffe1ba4524c

located at  1  4 4
array1 address = 0x7ffe1ba45250

located at  1  4 5
array1 address = 0x7ffe1ba45254

located at  1  4 6
array1 address = 0x7ffe1ba45258

located at  1  4 7
```

```
array1 address = 0x7ffe1ba4525c

located at  1  5 1
array1 address = 0x7ffe1ba45260

located at  1  5 2
array1 address = 0x7ffe1ba45264

located at  1  5 3
array1 address = 0x7ffe1ba45268

located at  1  5 4
array1 address = 0x7ffe1ba4526c

located at  1  5 5
array1 address = 0x7ffe1ba45270

located at  1  5 6
array1 address = 0x7ffe1ba45274

located at  1  5 7
array1 address = 0x7ffe1ba45278

located at  2  1 1
array1 address = 0x7ffe1ba4527c

located at  2  1 2
array1 address = 0x7ffe1ba45280

located at  2  1 3
array1 address = 0x7ffe1ba45284

located at  2  1 4
array1 address = 0x7ffe1ba45288

located at  2  1 5
array1 address = 0x7ffe1ba4528c

located at  2  1 6
array1 address = 0x7ffe1ba45290

located at  2  1 7
array1 address = 0x7ffe1ba45294

located at  2  2 1
array1 address = 0x7ffe1ba45298

located at  2  2 2
array1 address = 0x7ffe1ba4529c
```

```
located at  2  2 3
array1 address = 0x7ffe1ba452a0

located at  2  2 4
array1 address = 0x7ffe1ba452a4

located at  2  2 5
array1 address = 0x7ffe1ba452a8

located at  2  2 6
array1 address = 0x7ffe1ba452ac

located at  2  2 7
array1 address = 0x7ffe1ba452b0

located at  2  3 1
array1 address = 0x7ffe1ba452b4

located at  2  3 2
array1 address = 0x7ffe1ba452b8

located at  2  3 3
array1 address = 0x7ffe1ba452bc

located at  2  3 4
array1 address = 0x7ffe1ba452c0

located at  2  3 5
array1 address = 0x7ffe1ba452c4

located at  2  3 6
array1 address = 0x7ffe1ba452c8

located at  2  3 7
array1 address = 0x7ffe1ba452cc

located at  2  4 1
array1 address = 0x7ffe1ba452d0

located at  2  4 2
array1 address = 0x7ffe1ba452d4

located at  2  4 3
array1 address = 0x7ffe1ba452d8

located at  2  4 4
array1 address = 0x7ffe1ba452dc
```

```
located at  2  4 5
array1 address = 0x7ffe1ba452e0

located at  2  4 6
array1 address = 0x7ffe1ba452e4

located at  2  4 7
array1 address = 0x7ffe1ba452e8

located at  2  5 1
array1 address = 0x7ffe1ba452ec

located at  2  5 2
array1 address = 0x7ffe1ba452f0

located at  2  5 3
array1 address = 0x7ffe1ba452f4

located at  2  5 4
array1 address = 0x7ffe1ba452f8

located at  2  5 5
array1 address = 0x7ffe1ba452fc

located at  2  5 6
array1 address = 0x7ffe1ba45300

located at  2  5 7
array1 address = 0x7ffe1ba45304

located at  3  1 1
array1 address = 0x7ffe1ba45308

located at  3  1 2
array1 address = 0x7ffe1ba4530c

located at  3  1 3
array1 address = 0x7ffe1ba45310

located at  3  1 4
array1 address = 0x7ffe1ba45314

located at  3  1 5
array1 address = 0x7ffe1ba45318

located at  3  1 6
array1 address = 0x7ffe1ba4531c

located at  3  1 7
```

```
array1 address = 0x7ffe1ba45320

located at  3  2 1
array1 address = 0x7ffe1ba45324

located at  3  2 2
array1 address = 0x7ffe1ba45328

located at  3  2 3
array1 address = 0x7ffe1ba4532c

located at  3  2 4
array1 address = 0x7ffe1ba45330

located at  3  2 5
array1 address = 0x7ffe1ba45334

located at  3  2 6
array1 address = 0x7ffe1ba45338

located at  3  2 7
array1 address = 0x7ffe1ba4533c

located at  3  3 1
array1 address = 0x7ffe1ba45340

located at  3  3 2
array1 address = 0x7ffe1ba45344

located at  3  3 3
array1 address = 0x7ffe1ba45348

located at  3  3 4
array1 address = 0x7ffe1ba4534c

located at  3  3 5
array1 address = 0x7ffe1ba45350

located at  3  3 6
array1 address = 0x7ffe1ba45354

located at  3  3 7
array1 address = 0x7ffe1ba45358

located at  3  4 1
array1 address = 0x7ffe1ba4535c

located at  3  4 2
array1 address = 0x7ffe1ba45360
```

```
located at  3  4 3
array1 address = 0x7ffe1ba45364

located at  3  4 4
array1 address = 0x7ffe1ba45368

located at  3  4 5
array1 address = 0x7ffe1ba4536c

located at  3  4 6
array1 address = 0x7ffe1ba45370

located at  3  4 7
array1 address = 0x7ffe1ba45374

located at  3  5 1
array1 address = 0x7ffe1ba45378

located at  3  5 2
array1 address = 0x7ffe1ba4537c

located at  3  5 3
array1 address = 0x7ffe1ba45380

located at  3  5 4
array1 address = 0x7ffe1ba45384

located at  3  5 5
array1 address = 0x7ffe1ba45388

located at  3  5 6
array1 address = 0x7ffe1ba4538c

located at  3  5 7
array1 address = 0x7ffe1ba45390
```

it is same, option2 is right

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.
Answer:

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.
Answer: This function looks have poor temporal, but good spatial locality. First, scalar variables i,j and sum: These variables are on each pass through the loop, so they each have temporal locality. But, being a scalar, there is no guarantee of spatial locality. For the array variable, all variables are accessed by program only once.

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.
Answer:

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.
Answer: This function looks don't have good temporal and spatial locality. First, scalar variables i,j and sum: These variables are on each pass through the loop, so they each have temporal locality. But, being a scalar, there is no guarantee of spatial locality. However, For the array variable, all variables are not accessed by program only once. So this function have poor temporal and spatial locality.

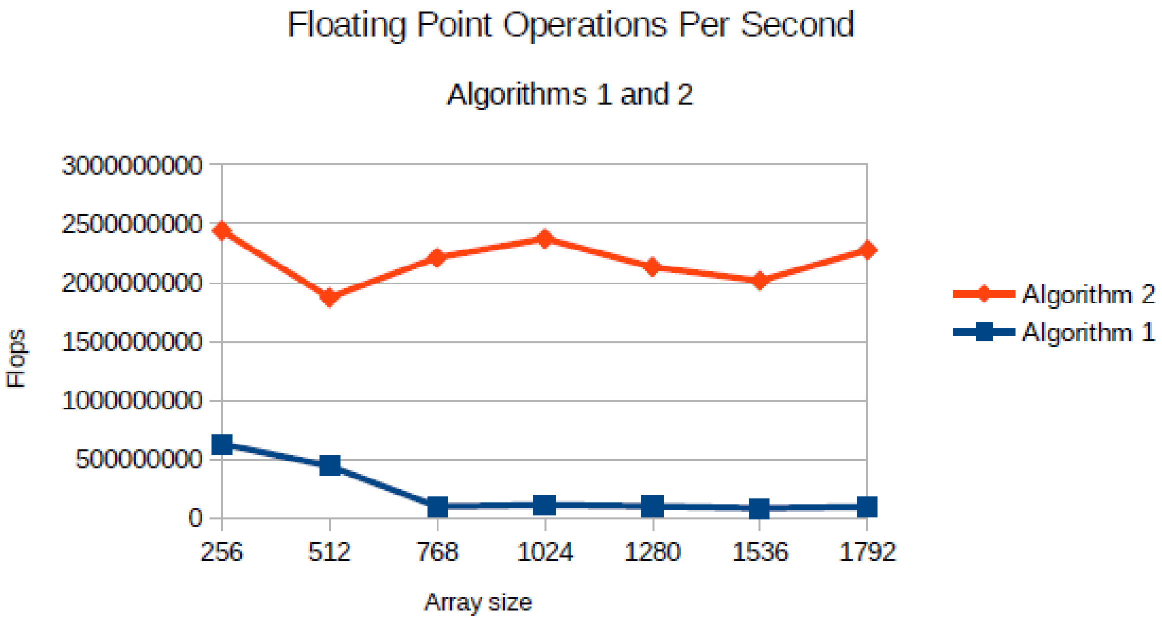
Question(8) Inspect the provided source code. Describe how the two-dimensional arrays are stored in memory, since the code only has one-dimensional array accesses like: a[element #].
Answer: The two-dimensional arrays will be stored in memory by major format. The first row is stored, then second row, third row. So that to access a specific row and column, instead of access a[1][2], the program access a[1*1+2].

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.
Answer:

Array Size	Algorithm1	Algorithm2
256	8.91e+08	2.39e+09
512	6.30e+08	1.81e+09
768	4.43e+08	1.43e+09
1024	1.01e+08	2.11e+09
1280	1.13e+08	2.26e+09
1536	1.03e+08	2.03e+09

1792	8.59e+07	1.93e+09
2048	9.69e+07	2.18e+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.
Answer:



Question(11) Be sure that the script source code is included in your Mercurial repository.
Answer:

```
for i in {256..2048..256}
do
	./matrix_math 1 $i
	./matrix_math 2 $i
done
```

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)
Answer:

```
processor	: 0
vendor_id	: GenuineIntel
cpu family	: 6
model		: 60
model name	: Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz
stepping	: 3
microcode	: 0x1c
cpu MHz			: 2494.280
cache size	: 6144 KB
physical id	: 0
```

```
siblings	: 1
core id		: 0
cpu cores	: 1
apicid		: 0
initial apicid	: 0
fpu		: yes
fpu_exception	: yes
cpuid level	: 13
wp		: yes
flags		: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon
pebs bts nopl xtopology tsc_reliable nonstop_tsc aperfmperf eagerfpu pni pclmulqdq ssse3
fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c
rdrand hypervisor lahf_lm abm ida arat epb pln pts dtherm fsgsbase tsc_adjust bmi1 avx2
smep bmi2 invpcid xsaveopt
bugs		:
bogomips	: 4988.56
clflush size	: 64
cache_alignment	: 64
address sizes	: 42 bits physical, 48 bits virtual
power management:
```

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) 4 x 32 KB 8-way set associative instruction caches
(b) 4 x 32 KB 8-way set associative data caches
(c) 6 MB 12-way set associative shared cache
(d) http://www.cpu-world.com/CPUs/Core_i7/Intel-Core%20i7-4710HQ%20Mobile%20processor.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: If computer was running several other active programs in the background, they will share the cpu cache with program mountain

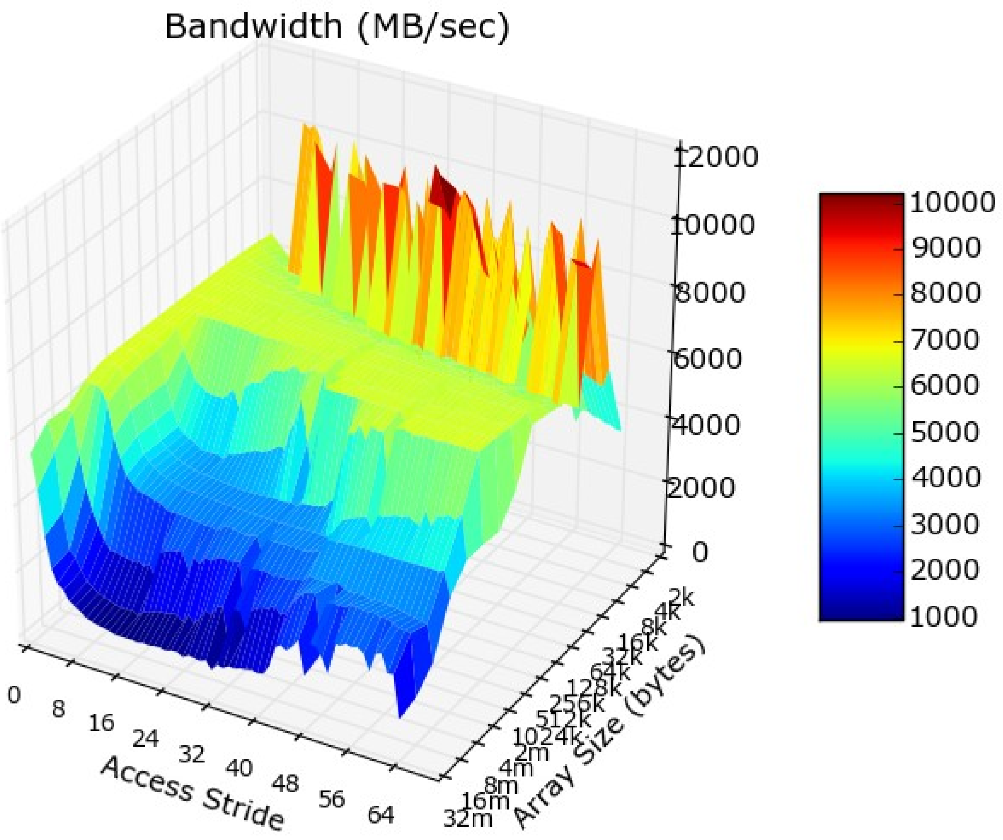
Question(15) What is the size (in bytes) of a data element read from the array in the test?
Answer: 2k bytes

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

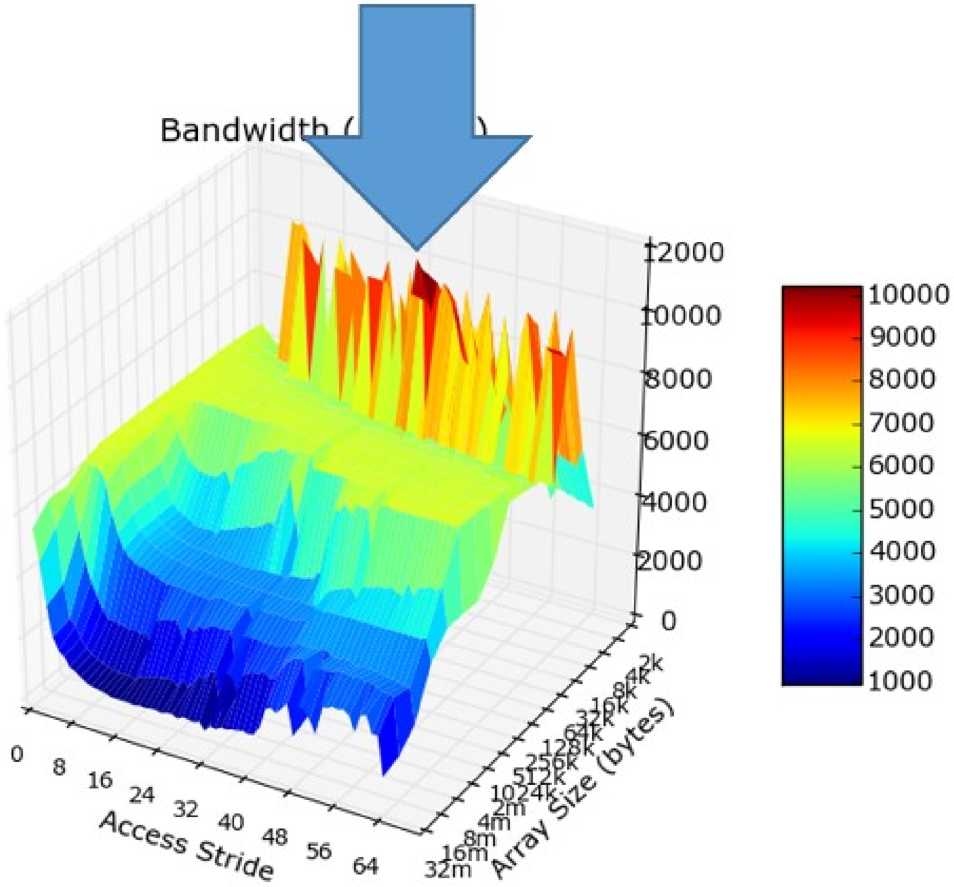
Answer: The range is from 1 to 64

Question(17) What is the range (min, max) of array sizes used in the test?
Answer: The array size is 2k to 32m

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
Answer:

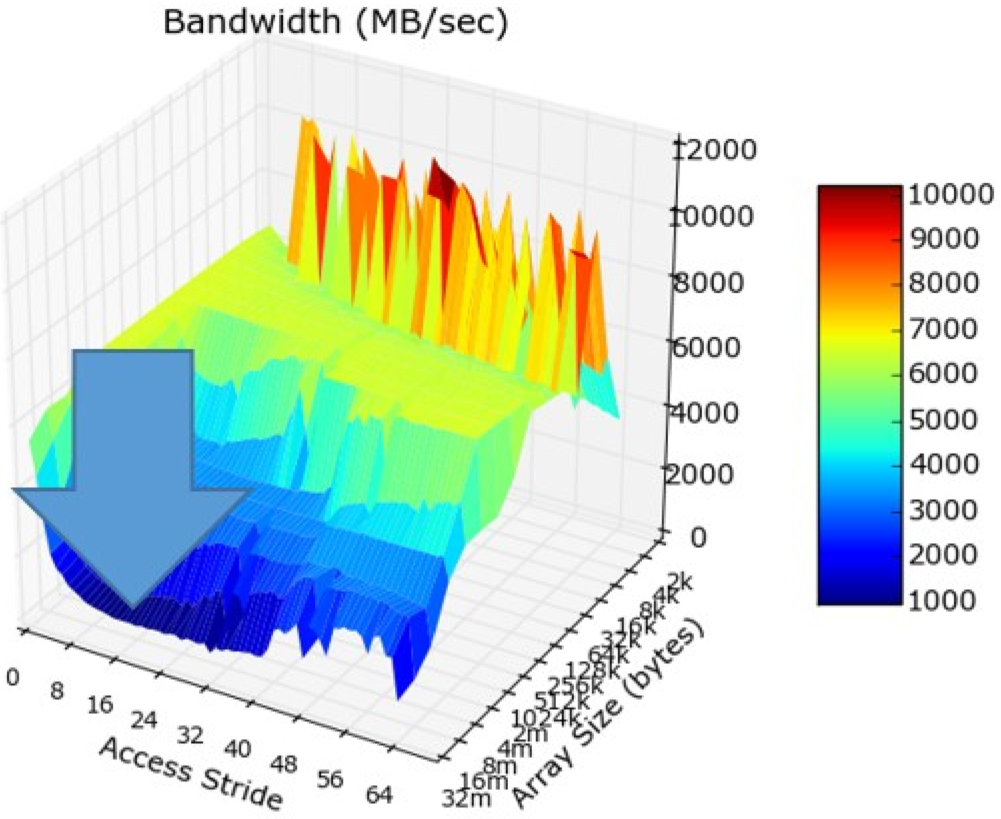


Question(19) What region (array size, stride) gets the most consistently high performance? (Ignoring spikes in the graph that are noisy results...) What is the read bandwidth reported? Annotate your figure by drawing an arrow on it.
Answer:



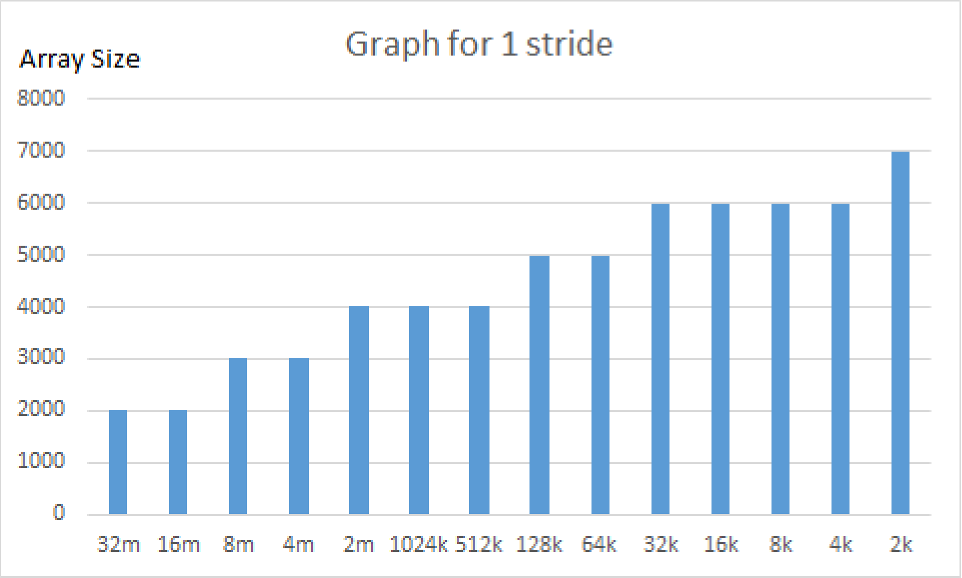
Stride 32, array size is 2k bytes has the highest performance. The bandwidth is round 10000 MB/s.

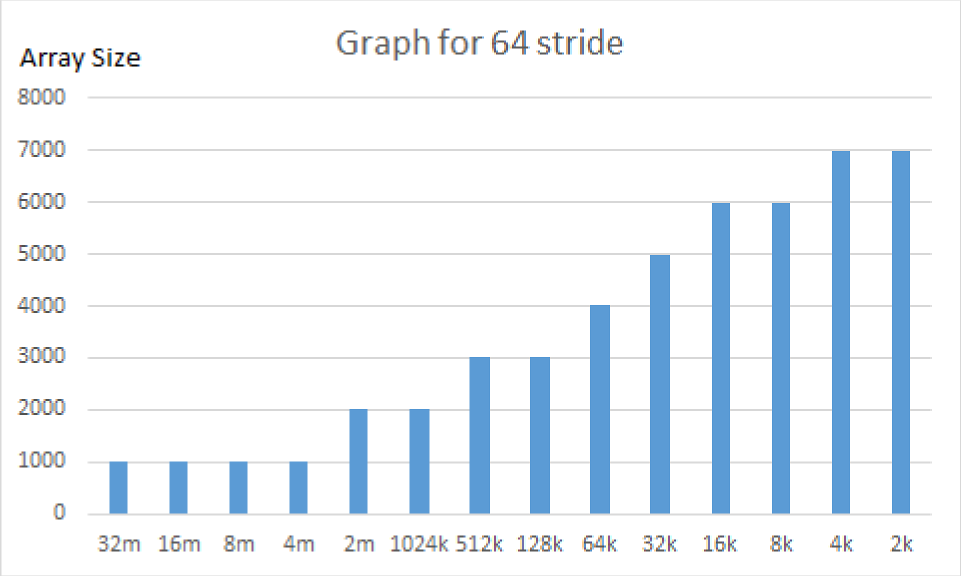
Question(20) What region (array size, stride) gets the most consistently low performance? (Ignoring spikes in the graph that are noisy results...) What is the read bandwidth reported? Annotate your figure by drawing an arrow on it.
Answer:



Stride 16, array size is 32m bytes has the lowest performance. The bandwidth is round 1000 MB/s.

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.
Answer: The reason why the stride = 1 is fast than stride = 64, even for large array size that are much larger than the L3 cache size, is due to the all array elements is in order. When the program read the data which is already in the cache. And all element in cache is already has been loaded when the first element was read. While the program is reading one element, the next address of element will be also read.

Question(23) What is temporal locality? What is spatial locality?
Answer: Temporal locality is when variables in a program are accessed multiple times.Spatial locality is when
variables in nearby memory locations are accessed too.

Question(24) Adjusting the total array size impacts temporal locality - why? Will an increased array size increase or decrease temporal locality?
Answer: Yes. Because the program loop the array element by using the temporary variable, j, the array size will affect the large number of the variable j. And the variable j is temporal locality, if there is a few elements in array and the program will access the variable j less times.

Question(25) Adjusting the read stride impacts spatial locality - why? Will an increased read stride increase or decrease spatial locality?
Answer: Adjusting the read stride impacts spatial locality because stride changes who the way to access the data. Increase the stride will decrease the spatial locality because the cache is reading in all adjacent memory
addresses, but the program is now accessing elements out of order.

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: 1. Try to write program that have read stride 1 when access data.
2. Try to re-use variables, such as the loop variable and some condition variables.