CPSC 424/524 Assignment 1

yl2335, Yangxiaokang Liu

Environment

```
[cpsc424_yl2335@grace2 ~]$ module load intel/2018b
[cpsc424_yl2335@grace2 ~]$ module list

Currently Loaded Modules:
  1) StdEnv        (S)   3) binutils/2.30-GCCcore-7.3.0     5) ifort/2018.3.222-GCC-7.3.0-2.30
  7) impi/2018.3.222-iccifort-2018.3.222-GCC-7.3.0-2.30   9) imkl/2018.3.222-iimpi-2018b
  2) GCCcore/7.3.0       4) icc/2018.3.222-GCC-7.3.0-2.30   6) iccifort/2018.3.222-GCC-7.3.0-2.30
  8) iimpi/2018b                                         10) intel/2018b

  Where:
   S:  Module is Sticky, requires --force to unload or purge



[cpsc424_yl2335@grace2 ~]$ which icc
/gpfs/loomis/apps/avx/software/ifort/2018.3.222-GCC-7.3.0-2.30/compilers_and_libraries_2018.3.222/
linux/bin/intel64/icc
[cpsc424_yl2335@grace2 ~]$ icc --version
icc (ICC) 18.0.3 20180410
Copyright (C) 1985-2018 Intel Corporation.  All rights reserved.
```

Running Experiments

Command

srun --pty -p interactive -c 1 -t 6:00:00 --mem-per-cpu=6gb bash ./run.sh

Outputs

----------Make----------
rm ex1a ex1b ex1c ex2
icc -g -O0 -fno-alias -std=c99 -o ex1a ex1.c timing.o
icc -g -O3 -xHost -fno-alias -std=c99 -o ex2 ex2.c dummy.c timing.o
icc -g -O3 -no-vec -no-simd -fno-alias -std=c99 -o ex1b ex1.c timing.o
icc -g -O3 -xHost -fno-alias -std=c99 -o ex1c ex1.c timing.o

----------Ex1 Option a----------
Approximation = 3.141593
Sin(4 * area) = 0.000000
Millions of flops: 5000.000000
Time consumed: 5.434563
Millions of flops per second: 920.037150

----------Ex1 Option b----------
Approximation = 3.141593
Sin(4 * area) = 0.000000
Millions of flops: 5000.000000
Time consumed: 5.434214
Millions of flops per second: 920.096285

----------Ex1 Option c----------
Approximation = 3.141593
Sin(4 * area) = -0.000000
Millions of flops: 5000.000000
Time consumed: 2.751773
Millions of flops per second: 1817.010264

----------Ex2----------
N   Time   Mflops
9   1.772019   1363.370950
19 1.670107   3053.860625
40 1.350091   3976.553579
85 1.205822   4730.593307
180 1.130042   5344.754755
378 1.148801   5520.353099

794 1.210063 5504.304204
1667 1.171854 2983.266112
3502 1.235782 2971.489652
7355 1.465514 2631.253189
15447 1.325071 1527.970227
32439 1.414168 1503.302383
68122 1.483196 1505.007879
143056 1.573838 1489.244453
300419 1.647766 1493.557116
630880 1.038121 1244.596977
1324849 1.028950 659.237770
2782184 1.058159 673.092785
5842587 1.134145 659.396393
12269432 1.239925 633.299358
25765808 1.205754 683.809324
54108198 1.355506 638.677515
113627216 1.392969 652.575756

Exercise 1: Division Performance

1. Demonstrate that your program actually computes a reasonable approximation to π .
    a. The outputs printed as doubles are 3.141593 which is close to the true value of π.
    b. $sin(4 * area) = 0$ which means arroximation must be multiples of π. We already know from (a) that it can't be other multiples but just π.
2. Report performances of option (a), option (b) and option (c) and explain their differences.
    - Phenomena

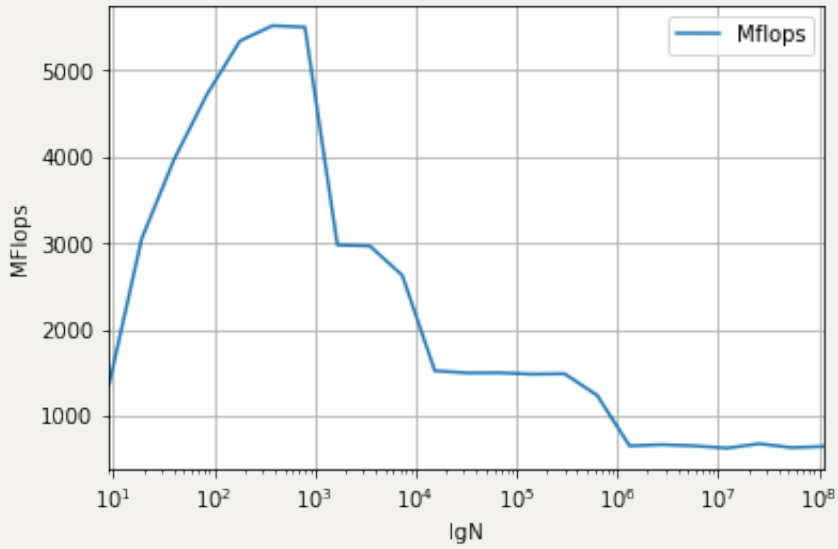
      The MFlops of option (a) and option (b) are almost the same. While the MFlops of option (c) almost doubles.
    - Explanation

Both option (b) and option (c) uses -O3. According to the manual icc, **O3 performs O2 optimizations and enables more aggresive loop transformations such as Fusion, Block-Unroll-and-Jam, and collapsing IF statements. However, option (b) uses -no-vec which disables all auto-vectorization, including vectorization of array notation statements. The option -no-simd disables vectorization of loops that have simd pragmas.** These two options make option (b) as slow as option (a). Option (c) uses -xHost which tells the compiler to generate instructions for the highest instruction set available on the compilation host processor. **"Ivy Bridge" nodes use AVX (Advanced Vector Extensions) instruction set which widen the vector registers from 128 bits to 256 bits, so the floating-point hardware can sustain 16 single-precision or 8 double-precision floating point operations per cycle, MFlops therefore doubles.**

Exercise 2: Vector Triad Performance

1. Generate a plot of the performance in MFlops vs. N. **Use a logarithmic scale for N on the x-axis**.

a.

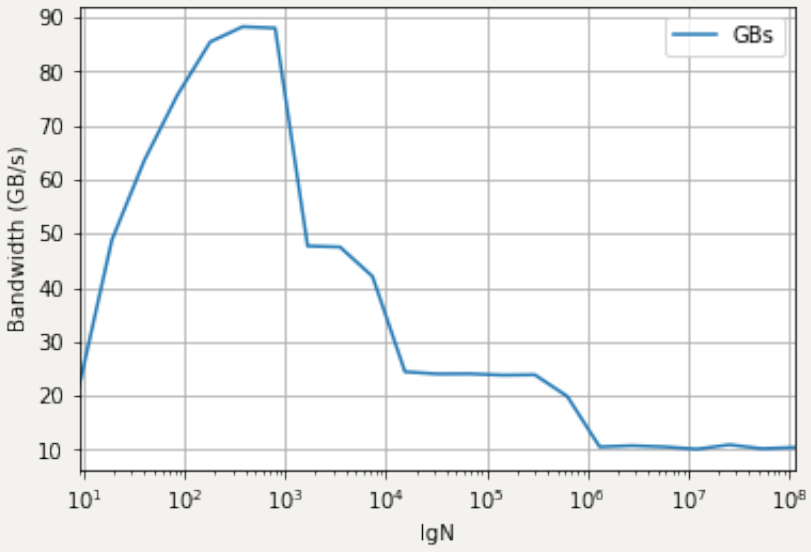


2. Explain the variations in terms of the processor architecture, by computing and discussing the apparent memory bandwidth that your data show near the variations in your plot.

a. Use lscpu to list out the information of the cpu

Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 16
On-line CPU(s) list: 0-15
Thread(s) per core: 2
Core(s) per socket: 8
Socket(s): 1
NUMA node(s): 1
Vendor ID: GenuineIntel
CPU family: 6
Model: 62
Model name: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
Stepping: 4
CPU MHz: 2600.016
BogoMIPS: 5200.03
Virtualization: VT-x
L1d cache: 32K
L1i cache: 32K
L2 cache: 256K
L3 cache: 20480K
NUMA node0 CPU(s): 0-15

b. We can also draw the performane in Bandwidth vs. N.



c. From the data we can identify three major drops of the bandwidth

    i. Bandwidth dropped from 88 GB/s to 48 GB/s when N increases from 794 to 1667. **The bandwidth drops because when N = 794, the size of the vectors is 4 * N * sizeof(double) = 25408 Bytes which is smaller than the capacity of the L1 cache 32 KB, when N = 1667, the size of the vectors is 53344 Bytes which exceeds the capacity of the L1d cache. Cache misses cause the bandwidth to drop.**
    ii. Bandwidth dropped from 42 GB/s to 24 GB/s when N increases from 7335 to 15447. **When N = 7335, the size of of the vectors is 234720 B which is smaller than the capacity of L2 cache 256KB, when N = 15447, the size of the vectors is 494304 B which exceeds L2's capacity. The bandwidth dropped**.
    iii. Bandwidth dropped from 20 GB/s to 11 GB/s when N increases from 630880 to 1324849. **When N = 630880, the size of of the vectors is 19715.0 KB which is smaller than the capacity of L3 cache 20480 KB, when N = 1324849, the size of the vectors is 41401.53125 KB which exceeds L3's capacity. The bandwidth dropped**.