EECE5640 Homework 3

Question 1

a.)

I choose four differet values(1, 34, 78, 120) as float and double inputs, and terms equals to 200000

The results of float format are as follow

```
Enter the value for x : 1
Enter the value for terms : 200000
(Float)Using a Taylor/Maclaurin series expansion: the sine of 1 degrees is
0.01745241
(Float)Using cmath sine to test: the sine of 1 degrees is 0.01745241
Time taken by function: 3.147 microseconds

Enter the value for x : 34
Enter the value for terms : 200000
(Float)Using a Taylor/Maclaurin series expansion: the sine of 34 degrees
is 0.5591929
(Float)Using cmath sine to test: the sine of 34 degrees is 0.5591929
Time taken by function: 3.22 microseconds


Enter the value for x : 78
Enter the value for terms : 200000
(Float)Using a Taylor/Maclaurin series expansion: the sine of 78 degrees
is 0.9781476
(Float)Using cmath sine to test: the sine of 78 degrees is 0.9781476
Time taken by function: 3.178 microseconds

Enter the value for x : 120
Enter the value for terms : 200000
(Float)Using a Taylor/Maclaurin series expansion: the sine of 120 degrees
is 0.8660253
(Float)Using cmath sine to test: the sine of 120 degrees is 0.8660254
Time taken by function: 3.207 microseconds
```

The results of double format are as follow

```
Enter the value for x : 1
Enter the value for terms : 200000
(double)Using a Taylor/Maclaurin series expansion: the sine of 1 degrees
is 0.0174524064173432
(double)Using cmath sine to test: the sine of 1 degrees is
0.0174524064173432
Time taken by function: 4.158 microseconds
```

```
Enter the value for x : 34
Enter the value for terms : 200000
(double)Using a Taylor/Maclaurin series expansion: the sine of 34 degrees
is 0.559192902908599
(double)Using cmath sine to test: the sine of 34 degrees is
0.559192902908599
Time taken by function: 3.598 microseconds

Enter the value for x : 78
Enter the value for terms : 200000
(double)Using a Taylor/Maclaurin series expansion: the sine of 78 degrees
is 0.978147600410383
(double)Using cmath sine to test: the sine of 78 degrees is
0.978147600410383
Time taken by function: 3.63 microseconds

Enter the value for x : 120
Enter the value for terms : 200000
(double)Using a Taylor/Maclaurin series expansion: the sine of 120 degrees
is 0.866025404981036
(double)Using cmath sine to test: the sine of 120 degrees is
0.866025404981036
Time taken by function: 5.978 microseconds
```

Results' differences between float and double format

As we can see float input running time is less than double input at same condition. The reasons are that float is less expensive, and requires less memory space as compared to double data type.

b.)

Request for 1 node with 1 cpu

```
srun --partition=short --nodes=1 --cpus-per-task=1 --pty /bin/bash
```

Compile using AVX

```
g++ -std=gnu++11 -o floatSinxAVX floatSinxAVX.cpp #SBATCH -
constraint=cascadelake -march=native
```

```
[li.junce@d0120 q1.b]$ ./floatSinxAVX
Enter the value for x : 2
Enter the value for terms : 20
(Float)Using a Taylor/Maclaurin series expansion: the sine of 2 degrees is
0.0348995
(Float)Using cmath sine to test: the sine of 2 degrees is 0.0348995
Time taken by function: 1072 ns
[li.junce@d0120 q1.b]$ g++ -std=gnu++11 -o floatSinx floatSinxAVX.cpp
[li.junce@d0120 q1.b]$ ./floatSinx
```

```
Enter the value for x : 2
Enter the value for terms : 20
(Float)Using a Taylor/Maclaurin series expansion: the sine of 2 degrees is
0.0348995
(Float)Using cmath sine to test: the sine of 2 degrees is 0.0348995
Time taken by function: 965 ns
```

As we can see, using AVX the performance has been improved. And the AVX-512 instruction set expands the register size of a CPU to improve performance. This improvement in performance makes it possible for CPUs to process data more quickly, enabling users to run video/audio compression algorithms more quickly.

Question 2

I choose using openMP to improve the performance of the matrix multiplication.

The results are as follow

```
[li.junce@d0010 q2]$ ./matmul
starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 7836.940617
ms
starting sparse matrix multiply
A result 0
The total time for matrix multiplication with sparse matrices =
7928.418560 ms
The sparsity of the a and b matrices = 0.750977


starting dense matrix multiply
a result 4.44488e+07
The total time for matrix multiplication with dense matrices = 3589.248809
ms
starting sparse matrix multiply
A result 0
The total time for matrix multiplication with sparse matrices =
3512.909992 ms
The sparsity of the a and b matrices = 0.750977
```

As you can see, we used openMP after that. There has been a significant reduction in running time.

Question 3

```
starting dense matrix multiply
a result 4.47104e+07
The total time for matrix multiplication with dense matrices = 15.669465
ms
starting sparse matrix multiply

A result -124
The total time for matrix multiplication with sparse matrices = 13.944186
ms
The sparsity of the a and b matrices = 0.750977
```

When we used GEMM method, the speed is very fast compared with the openMP and the normal one in question 2. Since OpenBLAS detects CPU capabilities at runtime and selects the fastest computation kernel for the running hardware.

Question 4

First computer system info:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                56
On-line CPU(s) list:   0-55
Thread(s) per core:    1
Core(s) per socket:    28
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Platinum 8276 CPU @ 2.20GHz
Stepping:              7
CPU MHz:               3000.695
CPU max MHz:           4000.0000
CPU min MHz:           1000.0000
BogoMIPS:              4400.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              39424K
NUMA node0 CPU(s):
0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,5
2,54
NUMA node1 CPU(s):
1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,5
3,55
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 art arch_perfmon pebs bts
rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq
dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid
dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx
f16c rdrand lahf_lm abm 3dnowprefetch epb cat_l3 cdp_l3 invpcid_single
intel_ppin intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms
invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb
avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc
cqm_mbm_total cqm_mbm_local dtherm ida arat pln pts pku ospke avx512_vnni
md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities
```

Fisrt computer system benchmark result:

```
23 October 2022 08:23:43 PM

LINPACK_BENCH
  C version

  The LINPACK benchmark.
  Language: C
  Datatype: Double precision real
  Matrix order N               = 1000
  Leading matrix dimension LDA = 1001

     Norm. Resid      Resid           MACHEP         X[1]          X[N]

        6.491510        0.000000    2.220446e-16        1.000000
1.000000

      Factor     Solve      Total     MFLOPS       Unit      Cray-Ratio

   0.700000   0.000000   0.700000  955.238095   0.002094  12.500000

LINPACK_BENCH
  Normal end of execution.

23 October 2022 08:23:43 PM
```

Second computer system info:

```
[li.junce@c0222 ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                48
On-line CPU(s) list:   0-47
Thread(s) per core:    2
```

```
Core(s) per socket:    12
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz
Stepping:              2
CPU MHz:               1273.657
CPU max MHz:           3500.0000
CPU min MHz:           1200.0000
BogoMIPS:              5199.93
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0-11,24-35
NUMA node1 CPU(s):     12-23,36-47
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 aperfmperf eagerfpu pni pclmulqdq dtes64
monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca
sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c
rdrand lahf_lm abm epb invpcid_single intel_ppin ssbd ibrs ibpb stibp
tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep
bmi2 erms invpcid cqm xsaveopt cqm_llc cqm_occup_llc dtherm ida arat pln
pts md_clear spec_ctrl intel_stibp flush_l1d
```

Second computer system benchmark result:

```
[li.junce@c0222 q4]$ ./linpack
23 October 2022 08:29:33 PM

LINPACK_BENCH
  C version

  The LINPACK benchmark.
  Language: C
  Datatype: Double precision real
  Matrix order N               = 1000
  Leading matrix dimension LDA = 1001

     Norm. Resid      Resid           MACHEP         X[1]          X[N]

        6.491510        0.000000    2.220446e-16        1.000000
1.000000

      Factor     Solve      Total     MFLOPS       Unit      Cray-Ratio
```

```
    0.840000   0.000000   0.840000  796.031746   0.002512  15.000000

LINPACK_BENCH
  Normal end of execution.

23 October 2022 08:29:34 PM
```

Third computer system info:

```
[li.junce@d0083 ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                56
On-line CPU(s) list:   0-55
Thread(s) per core:    1
Core(s) per socket:    28
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Platinum 8276 CPU @ 2.20GHz
Stepping:              7
CPU MHz:               3361.499
CPU max MHz:           4000.0000
CPU min MHz:           1000.0000
BogoMIPS:              4400.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              39424K
NUMA node0 CPU(s):
0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,5
2,54
NUMA node1 CPU(s):
1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,5
3,55
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 art arch_perfmon pebs bts
rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq
dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid
dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx
f16c rdrand lahf_lm abm 3dnowprefetch epb cat_l3 cdp_l3 invpcid_single
intel_ppin intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms
invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb
avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc
```

```
cqm_mbm_total cqm_mbm_local dtherm ida arat pln pts pku ospke avx512_vnni
md_clear spec_ctrl intel_stibp flush_l1d arch_capabilities
```

Third computer system benchmark result:

```
23 October 2022 08:46:45 PM

LINPACK_BENCH
  C version

  The LINPACK benchmark.
  Language: C
  Datatype: Double precision real
  Matrix order N               = 1000
  Leading matrix dimension LDA = 1001

     Norm. Resid      Resid           MACHEP         X[1]          X[N]

        6.491510        0.000000    2.220446e-16        1.000000
1.000000

      Factor     Solve      Total     MFLOPS       Unit      Cray-Ratio

   0.610000   0.010000   0.620000  1078.494624   0.001854  11.071429

LINPACK_BENCH
  Normal end of execution.

23 October 2022 08:46:46 PM
```

Question 5

Source

UVB (Unit Vector Block)

This storage format UVB (Unit Vector Block) for sparse matrix and vector, which needs less memory capacity compared to current popular formats for sparse vector and matrix such as COO, CSR, DIA, ELL and etc. Moreover, a new SpMV algorithm based on this new format can make full use of Intel ® AVX-512 instruction set, and decrease the execution time by up to 99% compared with the most popular LIBSVM kernel.

Utilizing the idea of a block-matrix or block-vector, this novel storage format divides a matrix or vector into numerous sections known as unit blocks. Then, to keep track of the positions of nonzero elements for each unit block, we introduce the Unit Vector Block (UVB) data structure.

In the novel storage format, a sparse matrix or vector is primarily represented by two arrays: the UVB array, which tracks the positions of nonzero elements, and the nonzero value array, which continuously stores the nonzero value of the sparse matrix or vector.

UVB has two fields: value offset, which is the index of the value array for the first nonzero element of a block, and bitmap, which indicates the position of nonzero elements in a block. Each bit in the bitmap is used to designate whether an element in a block is zero or nonzero, with 1s designating nonzero and 0s designating zero.