Performance of Sample CUDA Benchmarks on A100 vs V100

Authors: Dingwen Tao and Jiannan Tian

Date: January 31, 2021

Experimental platform

JLSE testbeds at Argonne National Laboratory:

- V100 node: 2x Intel Xeon Gold 6152 CPUs 22c 2.1Ghz + 4 Tesla V100-SXM2-32GB interconnected with NVLinks.
- A100 node: 1x AMD 7532 32c 2.4Ghz + 2 Ampere A100-PCIE-40GB interconnected with PCIe.

Basic Operations

1. reduction - CUDA Parallel Reduction - A100 win

**********A100**********
GPU Device 0: "Ampere" with compute capability 8.0

Using Device 0: A100-PCIE-40GB

Reducing array of type int

16777216 elements
256 threads (max)
64 blocks

Reduction, Throughput = 230.8605 GB/s, Time = 0.00029 s, Size = 16777216 Elements, NumDevsUsed = 1, Workgroup = 256

GPU result = 2139353471
CPU result = 2139353471

Test passed

**********V100**********
GPU Device 0: "Volta" with compute capability 7.0

Using Device 0: Tesla V100-SXM2-32GB

Reducing array of type int

16777216 elements
256 threads (max)
64 blocks

Reduction, Throughput = 191.8327 GB/s, Time = 0.00035 s, Size = 16777216 Elements, NumDevsUsed = 1, Workgroup = 256

GPU result = 2139353471
CPU result = 2139353471

Test passed

2. shfl_scan - CUDA Parallel Prefix Sum with Shuffle Intrinsics - A100 win

```
****************************************A100****************************************
Starting shfl_scan
GPU Device 0: "Ampere" with compute capability 8.0

> Detected Compute SM 8.0 hardware with 108 multi-processors
Starting shfl_scan
GPU Device 0: "Ampere" with compute capability 8.0

> Detected Compute SM 8.0 hardware with 108 multi-processors
Computing Simple Sum test
---------------------------------------------------
Initialize test data [1, 1, 1...]
Scan summation for 65536 elements, 256 partial sums
Partial summing 256 elements with 1 blocks of size 256
Test Sum: 65536
Time (ms): 0.024288
65536 elements scanned in 0.024288 ms -> 2698.287109 MegaElements/s
CPU verify result diff (GPUvsCPU) = 0
CPU sum (naive) took 0.034770 ms

Computing Integral Image Test on size 1920 x 1080 synthetic data
---------------------------------------------------
Method: Fast  Time (GPU Timer): 0.011296 ms Diff = 0
Method: Vertical Scan  Time (GPU Timer): 0.078048 ms
CheckSum: 2073600, (expect 1920x1080=2073600)


****************************************V100****************************************
Starting shfl_scan
GPU Device 0: "Volta" with compute capability 7.0

> Detected Compute SM 7.0 hardware with 80 multi-processors
Starting shfl_scan
GPU Device 0: "Volta" with compute capability 7.0

> Detected Compute SM 7.0 hardware with 80 multi-processors
Computing Simple Sum test
---------------------------------------------------
Initialize test data [1, 1, 1...]
Scan summation for 65536 elements, 256 partial sums
Partial summing 256 elements with 1 blocks of size 256
```

Test Sum: 65536
Time (ms): 0.020192
65536 elements scanned in 0.020192 ms -> 3245.642090 MegaElements/s
CPU verify result diff (GPUvsCPU) = 0
CPU sum (naive) took 0.026140 ms

Computing Integral Image Test on size 1920 x 1080 synthetic data
---
Method: Fast Time (GPU Timer): 0.018016 ms Diff = 0
Method: Vertical Scan Time (GPU Timer): 0.102944 ms
CheckSum: 2073600, (expect 1920x1080=2073600)

Dense Matrix Kernels

3. matrixMul - Matrix Multiplication (CUDA Runtime API Version) - V100 win

```
****************************************A100****************************************
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Ampere" with compute capability 8.0

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 2333.57 GFlop/s, Time= 0.056 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS
```

```
****************************************V100****************************************
[Matrix Multiply Using CUDA] - Starting...
GPU Device 0: "Volta" with compute capability 7.0

MatrixA(320,320), MatrixB(640,320)
Computing result using CUDA Kernel...
done
Performance= 3287.66 GFlop/s, Time= 0.040 msec, Size= 131072000 Ops, WorkgroupSize= 1024 threads/block
Checking computed result for correctness: Result = PASS
```

4. matrixMulDrv - Matrix Multiplication (CUDA Driver API Version) - A100 win

```
****************************************A100****************************************
[ matrixMulDrv (Driver API) ]
> Using CUDA Device [0]: A100-PCIE-40GB
> GPU Device has SM 8.0 compute capability
  Total amount of global memory:     42505273344 bytes
> findModulePath found file at <./matrixMul_kernel64.fatbin>
> initCUDA loading module: <./matrixMul_kernel64.fatbin>
Processing time: 0.077000 (ms)
Checking computed result for correctness: Result = PASS
```

V100
[matrixMulDrv (Driver API)]
> Using CUDA Device [0]: Tesla V100-SXM2-32GB
> GPU Device has SM 7.0 compute capability
  Total amount of global memory: 34089730048 bytes
> findModulePath found file at <./matrixMul_kernel64.fatbin>
> initCUDA loading module: <./matrixMul_kernel64.fatbin>
Processing time: 0.090000 (ms)
Checking computed result for correctness: Result = PASS

5. simpleCUBLAS - Simple CUBLAS - V100 win

A100
simpleCUBLAS

GPU Device 0: "Ampere" with compute capability 8.0

simpleCUBLAS test running..
simpleCUBLAS test passed.
Matrix size = 275x275, TFLOPS: 2.645

V100
simpleCUBLAS

GPU Device 0: "Volta" with compute capability 7.0

simpleCUBLAS test running..
simpleCUBLAS test passed.
Matrix size = 275x275, TFLOPS: 4.371

6. cudaTensorCoreGemm - CUDA Tensor Core GEMM - A100 win

A100
Initializing...
GPU Device 0: "Ampere" with compute capability 8.0

M: 4096 (16 x 256)
N: 4096 (16 x 256)
K: 4096 (16 x 256)
Preparing data for GPU...
Required shared memory size: 64 Kb
Computing... using high performance kernel compute_gemm

Time: 2.693504 ms
TFLOPS: 51.03

**********V100**********
Initializing...
GPU Device 0: "Volta" with compute capability 7.0

M: 4096 (16 x 256)
N: 4096 (16 x 256)
K: 4096 (16 x 256)
Preparing data for GPU...
Required shared memory size: 64 Kb
Computing... using high performance kernel compute_gemm
Time: 2.839136 ms
TFLOPS: 48.41

7. dmmaTensorCoreGemm - Double Precision Tensor Core GEMM - V100 N/A

**********A100**********
Initializing...
GPU Device 0: "Ampere" with compute capability 8.0

M: 8192 (8 x 1024)
N: 8192 (8 x 1024)
K: 4096 (4 x 1024)
Preparing data for GPU...
Required shared memory size: 68 Kb
Computing using high performance kernel = 0 - compute_dgemm_async_copy
Time: 63.051456 ms
FP64 TFLOPS: 8.72

**********V100**********
Initializing...
GPU Device 0: "Volta" with compute capability 7.0

dmmaTensorCoreGemm requires SM 8.0 or higher. Exiting…

8. bf16TensorCoreGemm - bf16TensorCoreGemm - V100 N/A

**********A100**********

Initializing...
GPU Device 0: "Ampere" with compute capability 8.0

M: 8192 (16 x 512)
N: 8192 (16 x 512)
K: 8192 (16 x 512)
Preparing data for GPU...
Required shared memory size: 72 Kb
Computing using high performance kernel = 0 - compute_bf16gemm_async_copy
Time: 14.387936 ms
TFLOPS: 76.42

V100
Initializing...
GPU Device 0: "Volta" with compute capability 7.0

bf16TensorCoreGemm requires requires SM 8.0 or higher to use Tensor Cores. Exiting...

9. tf32TensorCoreGemm - tf32 Tensor Core GEMM - V100 N/A

A100
Initializing...
GPU Device 0: "Ampere" with compute capability 8.0

M: 8192 (16 x 512)
N: 8192 (16 x 512)
K: 4096 (8 x 512)
Preparing data for GPU...
Required shared memory size: 72 Kb
Computing using high performance kernel = 0 - compute_tf32gemm_async_copy
Time: 24.476160 ms
TFLOPS: 22.46

V100
Initializing...
GPU Device 0: "Volta" with compute capability 7.0

tf32TensorCoreGemm requires requires SM 8.0 or higher to use Tensor Cores. Exiting...

Sparse Matrix Kernels

10. conjugateGradientCudaGraphs - conjugate gradient solver using CUBLAS and CUSPARSE library calls - V100 win

**********A100**********
GPU Device 0: "Ampere" with compute capability 8.0

> GPU device has 108 Multi-Processors, SM 8.0 compute capabilities

iteration = 1, residual = 4.449882e+01
iteration = 2, residual = 3.245218e+00
iteration = 3, residual = 2.690220e-01
iteration = 4, residual = 2.307639e-02
iteration = 5, residual = 1.993140e-03
iteration = 6, residual = 1.846192e-04
iteration = 7, residual = 1.693378e-05
iteration = 8, residual = 1.600115e-06
Test Summary: Error amount = 0.000000
Total time: 1.663 ms

**********V100**********
GPU Device 0: "Volta" with compute capability 7.0

> GPU device has 80 Multi-Processors, SM 7.0 compute capabilities

iteration = 1, residual = 4.449882e+01
iteration = 2, residual = 3.245218e+00
iteration = 3, residual = 2.690220e-01
iteration = 4, residual = 2.307639e-02
iteration = 5, residual = 1.993140e-03
iteration = 6, residual = 1.846192e-04
iteration = 7, residual = 1.693379e-05
iteration = 8, residual = 1.600115e-06
Test Summary: Error amount = 0.000000

Total time: 1.637 ms

11. conjugateGradientCudaGraphs - conjugate gradient solver using MultiBlock Cooperative Groups - A100 win

**********A100**********
Starting [conjugateGradientMultiBlockCG]...
GPU Device 0: "Ampere" with compute capability 8.0

> GPU device has 108 Multi-Processors, SM 8.0 compute capabilities

GPU Final, residual = 1.600115e-06, kernel execution time = 8.533664 ms
Test Summary: Error amount = 0.000000
&&&& conjugateGradientMultiBlockCG PASSED

**********V100**********
Starting [conjugateGradientMultiBlockCG]...
GPU Device 0: "Volta" with compute capability 7.0

> GPU device has 80 Multi-Processors, SM 7.0 compute capabilities

GPU Final, residual = 1.600115e-06, kernel execution time = 10.300832 ms
Test Summary: Error amount = 0.000000
&&&& conjugateGradientMultiBlockCG PASSED

Compression Kernels

12. nvJPEG_decoder - NVJPEG Decoder - A100 win

**********A100**********
GPU Device 0: "Ampere" with compute capability 8.0

Using GPU 0 (A100-PCIE-40GB, 108 SMs, 2048 th/SM max, CC 8.0, ECC on)
Decoding images in directory: ../../../../Samples/nvJPEG/images/, total 8, batchsize 1
Processing: ../../../../Samples/nvJPEG/images/img1.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img2.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img3.jpg
Image is 3 channels.
Channel #0 size: 640 x 426
Channel #1 size: 320 x 213
Channel #2 size: 320 x 213
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img4.jpg
Image is 3 channels.
Channel #0 size: 640 x 426
Channel #1 size: 320 x 213
Channel #2 size: 320 x 213
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img5.jpg
Image is 3 channels.
Channel #0 size: 640 x 480
Channel #1 size: 320 x 240
Channel #2 size: 320 x 240
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img6.jpg
Image is 3 channels.
Channel #0 size: 640 x 480
Channel #1 size: 320 x 240
Channel #2 size: 320 x 240

YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img7.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img8.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Total decoding time: 5.14954
Avg decoding time per image: 0.643692
Avg images per sec: 1.55354
Avg decoding time per batch: 0.643692

V100
GPU Device 0: "Volta" with compute capability 7.0

Using GPU 0 (Tesla V100-SXM2-32GB, 80 SMs, 2048 th/SM max, CC 7.0, ECC on)
Decoding images in directory: ../../../../Samples/nvJPEG/images/, total 8, batchsize 1
Processing: ../../../../Samples/nvJPEG/images/img1.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img2.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img3.jpg
Image is 3 channels.
Channel #0 size: 640 x 426
Channel #1 size: 320 x 213
Channel #2 size: 320 x 213
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img4.jpg
Image is 3 channels.

Channel #0 size: 640 x 426
Channel #1 size: 320 x 213
Channel #2 size: 320 x 213
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img5.jpg
Image is 3 channels.
Channel #0 size: 640 x 480
Channel #1 size: 320 x 240
Channel #2 size: 320 x 240
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img6.jpg
Image is 3 channels.
Channel #0 size: 640 x 480
Channel #1 size: 320 x 240
Channel #2 size: 320 x 240
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img7.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Processing: ../../../../Samples/nvJPEG/images/img8.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Total decoding time: 11.6794
Avg decoding time per image: 1.45992
Avg images per sec: 0.684967
Avg decoding time per batch: 1.45992

13. nvJPEG_encoder - NVJPEG Encoder - V100 win

**********A100**********
GPU Device 0: "Ampere" with compute capability 8.0

Using GPU 0 (A100-PCIE-40GB, 108 SMs, 2048 th/SM max, CC 8.0, ECC on)
Processing file: ../../../../Samples/nvJPEG_encoder/images/img1.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320

Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img1.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img2.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img2.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img3.jpg
Image is 3 channels.
Channel #0 size: 640 x 426
Channel #1 size: 320 x 213
Channel #2 size: 320 x 213
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img3.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img4.jpg
Image is 3 channels.
Channel #0 size: 640 x 426
Channel #1 size: 320 x 213
Channel #2 size: 320 x 213
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img4.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img5.jpg
Image is 3 channels.
Channel #0 size: 640 x 480
Channel #1 size: 320 x 240
Channel #2 size: 320 x 240
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img5.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img6.jpg
Image is 3 channels.
Channel #0 size: 640 x 480
Channel #1 size: 320 x 240
Channel #2 size: 320 x 240
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img6.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img7.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling

Writing JPEG file: encode_output/img7.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img8.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img8.jpg
Total images processed: 8
Total time spent on encoding: 2.2943
Avg time/image: 0.286788

V100
GPU Device 0: "Volta" with compute capability 7.0

Using GPU 0 (Tesla V100-SXM2-32GB, 80 SMs, 2048 th/SM max, CC 7.0, ECC on)
Processing file: ../../../../Samples/nvJPEG_encoder/images/img1.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img1.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img2.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img2.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img3.jpg
Image is 3 channels.
Channel #0 size: 640 x 426
Channel #1 size: 320 x 213
Channel #2 size: 320 x 213
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img3.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img4.jpg
Image is 3 channels.
Channel #0 size: 640 x 426
Channel #1 size: 320 x 213
Channel #2 size: 320 x 213
YUV 4:2:0 chroma subsampling

Writing JPEG file: encode_output/img4.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img5.jpg
Image is 3 channels.
Channel #0 size: 640 x 480
Channel #1 size: 320 x 240
Channel #2 size: 320 x 240
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img5.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img6.jpg
Image is 3 channels.
Channel #0 size: 640 x 480
Channel #1 size: 320 x 240
Channel #2 size: 320 x 240
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img6.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img7.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img7.jpg
Processing file: ../../../../Samples/nvJPEG_encoder/images/img8.jpg
Image is 3 channels.
Channel #0 size: 480 x 640
Channel #1 size: 240 x 320
Channel #2 size: 240 x 320
YUV 4:2:0 chroma subsampling
Writing JPEG file: encode_output/img8.jpg
Total images processed: 8
Total time spent on encoding: 1.7017
Avg time/image: 0.212712

Nvidia Production Libraries

14. cuSolverDn_LinearSolver - cuSolverDn Linear Solver - V100 win

**********A100**********
GPU Device 0: "Ampere" with compute capability 8.0

step 1: read matrix market format
Using default input file [../../../../Samples/cuSolverDn_LinearSolver/gr_900_900_crg.mtx]
sparse matrix A is 900 x 900 with 7744 nonzeros, base=1
step 2: convert CSR(A) to dense matrix
step 3: set right hand side vector (b) to 1
step 4: prepare data on device
step 5: solve A*x = b
timing: cholesky = 0.002851 sec
step 6: evaluate residual
|b - A*x| = 1.136868E-13
|A| = 1.600000E+01
|x| = 2.357708E+01
|b - A*x|/(|A|*|x|) = 3.013701E-16

**********V100**********
GPU Device 0: "Volta" with compute capability 7.0

step 1: read matrix market format
Using default input file [../../../../Samples/cuSolverDn_LinearSolver/gr_900_900_crg.mtx]
sparse matrix A is 900 x 900 with 7744 nonzeros, base=1
step 2: convert CSR(A) to dense matrix
step 3: set right hand side vector (b) to 1
step 4: prepare data on device
step 5: solve A*x = b
timing: cholesky = 0.001751 sec
step 6: evaluate residual
|b - A*x| = 1.136868E-13
|A| = 1.600000E+01
|x| = 2.357708E+01
|b - A*x|/(|A|*|x|) = 3.013701E-16

15. cuSolverSp_LinearSolver - cuSolverSp Linear Solver - V100 win

**********A100**********
GPU Device 0: "Ampere" with compute capability 8.0

```
Using default input file [../../../../Samples/cuSolverSp_LinearSolver/lap2D_5pt_n100.mtx]
step 1: read matrix market format
sparse matrix A is 10000 x 10000 with 49600 nonzeros, base=1
step 2: reorder the matrix A to minimize zero fill-in
      if the user choose a reordering by -P=symrcm, -P=symamd or -P=metis
step 2.1: no reordering is chosen, Q = 0:n-1
step 2.2: B = A(Q,Q)
step 3: b(j) = 1 + j/n
step 4: prepare data on device
step 5: solve A*x = b on CPU
step 6: evaluate residual r = b - A*x (result on CPU)
(CPU) |b - A*x| = 4.999334E-12
(CPU) |A| = 8.000000E+00
(CPU) |x| = 1.136492E+03
(CPU) |b| = 1.999900E+00
(CPU) |b - A*x|/(|A|*|x| + |b|) = 5.497437E-16
step 7: solve A*x = b on GPU
step 8: evaluate residual r = b - A*x (result on GPU)
(GPU) |b - A*x| = 1.984857E-12
(GPU) |A| = 8.000000E+00
(GPU) |x| = 1.136492E+03
(GPU) |b| = 1.999900E+00
(GPU) |b - A*x|/(|A|*|x| + |b|) = 2.182616E-16
timing chol: CPU =   0.115463 sec , GPU =   0.283314 sec
show last 10 elements of solution vector (GPU)
consistent result for different reordering and solver
x[9990] = 3.000016E+01
x[9991] = 2.807343E+01
x[9992] = 2.601354E+01
x[9993] = 2.380285E+01
x[9994] = 2.141866E+01
x[9995] = 1.883070E+01
x[9996] = 1.599668E+01
x[9997] = 1.285365E+01
x[9998] = 9.299423E+00
x[9999] = 5.147265E+00


****************************************V100****************************************
GPU Device 0: "Volta" with compute capability 7.0

Using default input file [../../../../Samples/cuSolverSp_LinearSolver/lap2D_5pt_n100.mtx]
step 1: read matrix market format
```

```
sparse matrix A is 10000 x 10000 with 49600 nonzeros, base=1
step 2: reorder the matrix A to minimize zero fill-in
      if the user choose a reordering by -P=symrcm, -P=symamd or -P=metis
step 2.1: no reordering is chosen, Q = 0:n-1
step 2.2: B = A(Q,Q)
step 3: b(j) = 1 + j/n
step 4: prepare data on device
step 5: solve A*x = b on CPU
step 6: evaluate residual r = b - A*x (result on CPU)
(CPU) |b - A*x| = 4.999334E-12
(CPU) |A| = 8.000000E+00
(CPU) |x| = 1.136492E+03
(CPU) |b| = 1.999900E+00
(CPU) |b - A*x|/(|A|*|x| + |b|) = 5.497437E-16
step 7: solve A*x = b on GPU
step 8: evaluate residual r = b - A*x (result on GPU)
(GPU) |b - A*x| = 1.984857E-12
(GPU) |A| = 8.000000E+00
(GPU) |x| = 1.136492E+03
(GPU) |b| = 1.999900E+00
(GPU) |b - A*x|/(|A|*|x| + |b|) = 2.182616E-16
timing chol: CPU =   0.121303 sec , GPU =   0.112445 sec
show last 10 elements of solution vector (GPU)
consistent result for different reordering and solver
x[9990] = 3.000016E+01
x[9991] = 2.807343E+01
x[9992] = 2.601354E+01
x[9993] = 2.380285E+01
x[9994] = 2.141866E+01
x[9995] = 1.883070E+01
x[9996] = 1.599668E+01
x[9997] = 1.285365E+01
x[9998] = 9.299423E+00
x[9999] = 5.147265E+00
```

16. simpleCUFFT - Simple CUFFT - V100 win

```
****************************************A100****************************************
[simpleCUFFT] is starting...
GPU Device 0: "Volta" with compute capability 7.0

Temporary buffer size 448 bytes
Transforming signal cufftExecC2C
Launching ComplexPointwiseMulAndScale<<< >>>
```

Transforming signal back cufftExecC2C

Total Time	Instances	Average	Minimum	Maximum	Name
13856	3	4618.7	4192	5024	void composite_2way_fft<56u, 7u, 8u
2368	1	2368.0	2368	2368	ComplexPointwiseMulAndScale

Total time: 0.016224 ms

**********V100**********
[simpleCUFFT] is starting...
GPU Device 0: "Volta" with compute capability 7.0

Temporary buffer size 448 bytes
Transforming signal cufftExecC2C
Launching ComplexPointwiseMulAndScale<<< >>>
Transforming signal back cufftExecC2C

Total Time	Instances	Average	Minimum	Maximum	Name
8672	3	2890.7	2304	3840	void composite_2way_fft<56u, 7u, 8u
1536	1	1536.0	1536	1536	ComplexPointwiseMulAndScale

Total time: 0.010208 ms