Assignment 1

Yu Sun Danyu Zhou

November 2023

0.1 Task1 (Sun)

0.1.1 1.List the main differences between GPUs and CPUs in terms of architecture.

1. CPU has tens of massive cores, CPU excels at irregular control-intensive work . Lots of hardware for control, fewer ALUs

2. GPU has thousands of small cores, GPU excels at regular math-intensive work . Lots of ALUs, little hardware for control

0.1.2 2.Top 500

1. In top 10 computer, All computer except Chinese TaihuLight use GPU, while 5 of them are provided by Nvidia and 2 from AMD 1 from Fujitsu, 1 from Matrix.

2. Power Efficiency

	Rmax (PFlop/s)	Power (kW)	Power Effciency
Frontier	1,102.00	21,100	0.0522
Fugaku	442.01	29,899	0.014783
LUMI	151.9	2942	0.0511
Leonardo	148.60	10,096	0.0147187
Summit	94.64	7438	0.01272
Sierra	93.01	15,371	0.00605
Sunway TaihuLight	70.87	2589	0.0273
Perlmutter	63.46	2646	0.02398
Selene	61.44	18,482	0.00332
Tianhe-2A	46.1	921	0.05

Task 2 (Zhou)

1. The screenshot of the output from running deviceQuery test in /1_Utilities.

```
! ./deviceQuery
```

```
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Tesla T4"
  CUDA Driver Version / Runtime Version          12.0 / 11.8
  CUDA Capability Major/Minor version number:    7.5
  Total amount of global memory:                 15102 MBytes (15835398144 bytes)
  (040) Multiprocessors, (064) CUDA Cores/MP:    2560 CUDA Cores
  GPU Max Clock rate:                            1590 MHz (1.59 GHz)
  Memory Clock rate:                             5001 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 4194304 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        65536 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1024
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 3 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 4
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.0, CUDA Runtime Version = 11.8, NumDevs = 1
Result = PASS
```

2. What is the Compute Capability of your GPU device?

   7.5

3. The screenshot of the output from running bandwidthTest test in /1_Utilities.

```
!./bandwidthTest
```

```
[CUDA Bandwidth Test] - Starting...
Running on...

 Device 0: Tesla T4
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(GB/s)
   32000000                     11.4

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(GB/s)
   32000000                     10.4

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(GB/s)
   32000000                     239.4

Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
```

4. How will you calculate the GPU memory bandwidth (in GB/s) using the output from deviceQuery? (Hint: memory bandwidth is typically determined by clock rate and bus width, and check what double date rate (DDR) may impact the bandwidth). Are they consistent with your results from bandwidthTest?

```
With DDR : 2 * 256 * 5001 * 10^6 = 320GB/s
Without DDR : 160GB/s
Both figures have a about 100 GB/s difference in bandwidth with
the Device to Device Bandwidth from the test
```

Task3 (Zhou)

1. Compile both OMP and CUDA versions of your selected benchmarks. Do you need to make any changes in Makefile?

   We need to modify the rodinia_3.1/cuda/lud/cuda/Makefile if we want to run the lud.

   ```
   remove: -arch=sm_13 \
   in  NVCCFLAGS +=  section
   ```

   For rodinia_3.1/cuda/b+tree/Makefile,

   ```
   remove line CUDA_FLAG = -arch sm_20 and all variables related to CUDA_FLAG
   ```

2. Ensure the same input problem is used for OMP and CUDA versions. Report and compare their execution time.

```
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/openmp/lud# bash run
Generate input matrix internally, size =256
Creating matrix internally size=256
running OMP on host
Time consumed(ms): 27.457000
```

```
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/cuda/lud# bash run
WG size of kernel = 16 X 16
Generate input matrix internally, size =256
Creating matrix internally size=256
Before LUD
Time consumed(ms): 3.209000
After LUD
>>>Verify<<<<
```

```
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/openmp/b+tree# bash run
Input File: ../../data/b+tree/mil.txt
Command File: ../../data/b+tree/command.txt
Command Buffer:
j 6000 3000
k 10000


Getting input from file core...
Transforming data to a GPU suitable structure...
Tree transformation took 0.098952
Waiting for command
>
******command: j count=6000, rSize=6000
Time spent in different stages of CPU/MCPU KERNEL:
 0.000005000000 s,  0.047732695937 % : MCPU: SET DEVICE
 0.010470000096 s, 99.952270507812 % : CPU/MCPU: KERNEL
Total time:
0.010475000367 s
> > > > > > > > > > > >
 ******command: k count=10000
Time spent in different stages of CPU/MCPU KERNEL:
 0.000001000000 s,  0.008475294337 % : MCPU: SET DEVICE
 0.011797999963 s, 99.991523742676 % : CPU/MCPU: KERNEL
Total time:
0.011799000204 s
> > > > > > > > > > >
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/openmp/b+tree#
```

```
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/cuda/b+tree# bash run
WG size of kernel 1 & 2  = 256
Selecting device 0
Input File: ../../data/b+tree/mil.txt
Command File: ../../data/b+tree/command.txt
Command Buffer:
j 6000 3000
k 10000
Getting input from file ../../data/b+tree/mil.txt...
Transforming data to a GPU suitable structure...
Tree transformation took 0.403506
Waiting for command
>
******command: j count=6000, rSize=6000
knodes_elem=7874, knodes_unit_mem=2068, knodes_mem=16283432
# of blocks = 6000, # of threads/block = 256 (ensure that device can handle)
Time spent in different stages of GPU_CUDA KERNEL:
 0.864696025848 s, 98.529289245605 % : GPU: SET DEVICE / DRIVER INIT
 0.002187999897 s,  0.249315470457 % : GPU MEM: ALO
 0.008689999580 s,  0.990197181702 % : GPU MEM: COPY IN
 0.000292999990 s,  0.033386394382 % : GPU: KERNEL
 0.000315000012 s,  0.035893220454 % : GPU MEM: COPY OUT
 0.001420999994 s,  0.161918312311 % : GPU MEM: FRE
Total time:
0.877602994442 s
> > > > > > > > > > > > >
 ******command: k count=10000
records_elem=1000000, records_unit_mem=4, records_mem=4000000
knodes_elem=7874, knodes_unit_mem=2068, knodes_mem=16283432
# of blocks = 10000, # of threads/block = 256 (ensure that device can handle)
Time spent in different stages of GPU_CUDA KERNEL:
 0.000015000000 s,  0.278035223484 % : GPU: SET DEVICE / DRIVER INIT
 0.001618999988 s, 30.009265899658 % : GPU MEM: ALO
 0.002870999975 s, 53.215938568115 % : GPU MEM: COPY IN
 0.000209000005 s,  3.873957395554 % : GPU: KERNEL
 0.000032000000 s,  0.593141794205 % : GPU MEM: COPY OUT
 0.000648999994 s, 12.029656410217 % : GPU MEM: FRE
Total time:
0.005394999869 s
> > > > > > > > > > >
```

3. Do you observe expected speedup on GPU compared to CPU? Why or Why not?

There's speedup for lud, 27.45ms is reduced to 3.21ms when running on gpu.

For b+tree, 0.877602994442s for gpu and 0.017s for cpu.

Therefore, not all algorithms could benefit from parallelism.

Task4 (Both)

1. How do you launch the code on GPU on Dardel supercomputer?

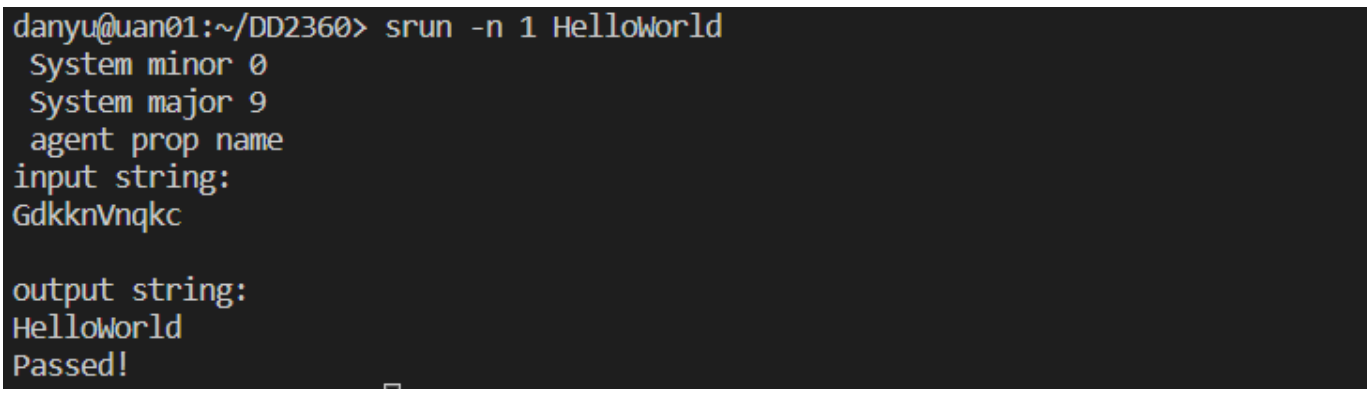
    1. Connect to Dardel according to the tutorial

    2. Upload cpp and makefile through scp command

    3. Compile remotely on Dardel

    4. ```
       salloc -A edu23.dd2360 -p gpu -N 1 -t 00:10:00
       srun -n 1 HelloWorld
       ```

2. Include a screenshot of your output from Dardel


       ```