Final Report

Team Members: Daniel Ferriss, Ronald Roy
Team NetIDs: ferriss2, rroy21
Team Name: Team Ginkgo Tree in Front of Noyes
School Affiliation: UIUC

M2

Rai Output:

```
* Running /bin/bash -c "time ./m2"
Test batch size: 10000
Loading fashion-mnist data...Done
Loading model...Done
Conv-CPU==
Op Time: 97216.1 ms
Conv-CPU==
Op Time: 332180 ms
Test Accuracy: 0.8714
real 8m43.457s
user 8m42.355s
sys  0m1.096s
```

M3

Rai Output:

```
* Running /bin/bash -c "./m3"
Test batch size: 10000
Loading fashion-mnist data...Done
Loading model...Done
Conv-GPU==
Op Time: 1472.66 ms
Conv-GPU==
Op Time: 480.87 ms
Test Accuracy: 0.8714

Exported successfully to
/build/report1.sqlite
Generating CUDA API Statistics...
CUDA API Statistics (nanoseconds)
```

Nsys Output:

```
CUDA API Statistics (nanoseconds)

Time(%)      Total Time       Calls         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------
--------------------------------------------------------------------------------
   78.7       980499966           6     163416661.0           83582       519301896  cudaMemcpy
   14.7       183039361           6      30506560.2           76925       181210085  cudaMalloc
```

```
    6.4        79156995           2      39578497.5        16167056        62989939  cudaDeviceSynchronize
    0.2         2284526           6        380754.3           57075          942550  cudaFree
    0.0          234438           2        117219.0           25982          208456  cudaLaunchKernel
Generating CUDA Kernel Statistics...

Generating CUDA Memory Operation Statistics...
CUDA Kernel Statistics (nanoseconds)

Time(%)      Total Time   Instances         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------
-------------------------------------------------------------------------------
  100.0        79301821           2      39650910.5        16165048        63136773  conv_forward_kernel


CUDA Memory Operation Statistics (nanoseconds)

Time(%)      Total Time  Operations         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------
-------------------------------------------------------------------------------
   92.6       903278350           2     451639175.0       384778100       518500250  [CUDA memcpy DtoH]
    7.4        71839534           4      17959883.5            1216        38667685  [CUDA memcpy HtoD]


CUDA Memory Operation Statistics (KiB)

             Total      Operations            Average            Minimum            Maximum  Name
----------------  --------------  -----------------  -----------------  -----------------
-------------------------------------------------------------------------------
        1722500.0               2           861250.0         722500.000          1000000.0  [CUDA memcpy DtoH]
         538919.0               4           134729.0              0.766           288906.0  [CUDA memcpy HtoD]
Generating Operating System Runtime API Statistics...
Operating System Runtime API Statistics (nanoseconds)

Time(%)      Total Time       Calls         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------
-------------------------------------------------------------------------------
   33.3     93773254392         952      98501317.6           51627       100195009  sem_timedwait
   33.3     93689722004         950      98620760.0           62402       100283932  poll
   22.1     62219329581           2   31109664790.5     22401061350     39818268231  pthread_cond_wait
   11.2     31508607416          63     500136625.7       500059315       500168001  pthread_cond_timedwait
    0.0       101611290         764        132999.1            1040        17448461  ioctl
    0.0        16216161        9072          1787.5            1022           18050  read
    0.0         3656327          97         37694.1            1186         1684989  mmap
    0.0          602695          97          6213.4            1856           21644  open64
    0.0          306044           1        306044.0          306044          306044  pthread_mutex_lock
    0.0          293833           5         58766.6           32017           97170  pthread_create
    0.0           71854           3         23951.3           11931           47790  fgets
    0.0           69345          18          3852.5            1199           20894  munmap
    0.0           65838          15          4389.2            2170            9377  write
    0.0           60118          20          3005.9            1032            9503  fopen
    0.0           58022           3         19340.7            2972           34067  fopen64
    0.0           44538           5          8907.6            2605           22158  open
    0.0           38972           8          4871.5            1103            7548  fflush
    0.0           15805           8          1975.6            1062            5807  fclose
    0.0           15321           3          5107.0            4771            5528  pipe2
    0.0           13718           2          6859.0            3953            9765  pthread_cond_signal
    0.0            9771           2          4885.5            3234            6537  socket
    0.0            5304           1          5304.0            5304            5304  connect
0.0            4345           2          2172.5            1039            3306  fwrite
    0.0            3433           3          1144.3            1029            1335  fcntl
0.0            1404           1          1404.0            1404            1404  bind
```

List of kernels that consume more than 90% program time:
conv_forward_kernel (100%)

List of cuda API calls that consume more than 90% program time:
cudaMemcpy (78.7%)
cudaMalloc (14.7%)

Difference between kernels and API calls:
The kernel is the code run on the device. It is run in parallel and is executed by different CUDA threads. Each CUDA thread that executes the kernel has a different ID.

The API calls are all on the host. An example of a API call is setting up the memory for the kernel. They are an extension to the language the developer is using, and calls the lower level C API.

GPU SOL utilization: