DD2360 Assignment 4
Group 35
Shiyuan Shan
Github repository: ShiyuanShan/DD2360HT23 (github.com)

Exercise 1

1.1 16*16 threads per block / 32 threads per warp = 8 warps per block,
In total ceil(800/16) * ceil(600/16) = 1900 blocks.
Thus 1900 * 8 = 15200 warps.

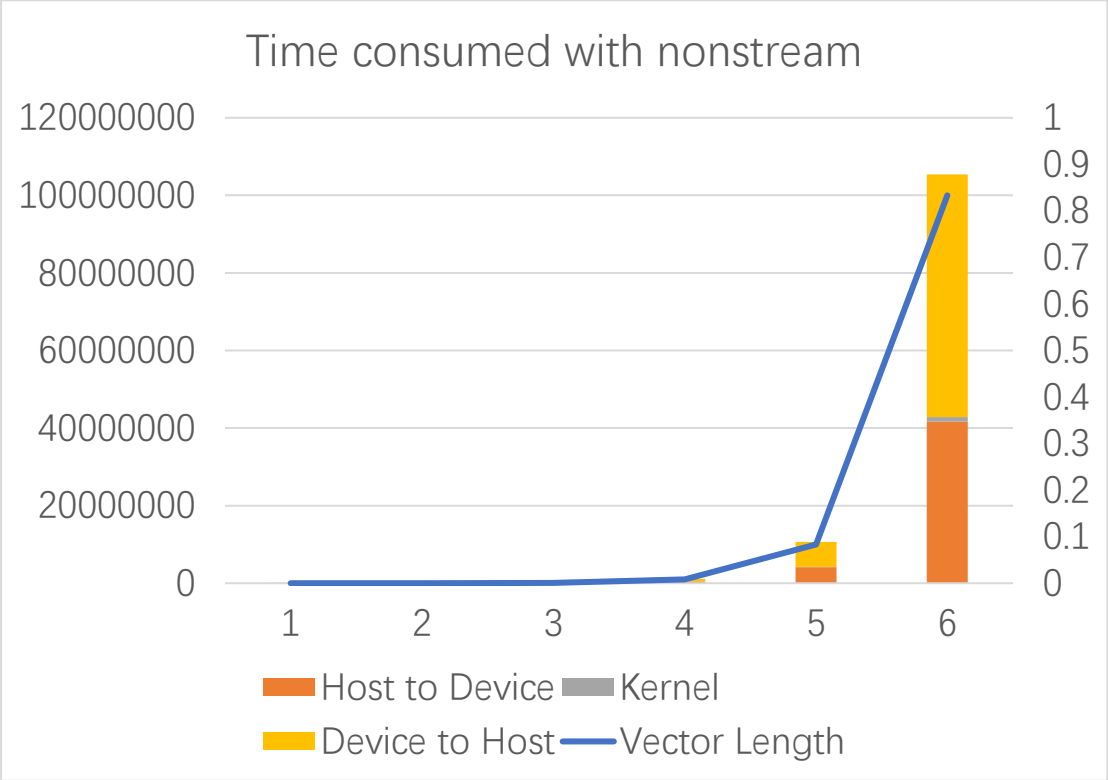
800 % 16 = 0, so no block has divergence in horizontal direction.
600 % 16 = 8, the bottom most blocks have divergence.
Each warp is a 16*2 thread array,
8 % 2 = 0, Thus no warp has divergence. In a bottom most block there would be 4 warps following “THEN” statement and the rest 4 following “ELSE” statement.

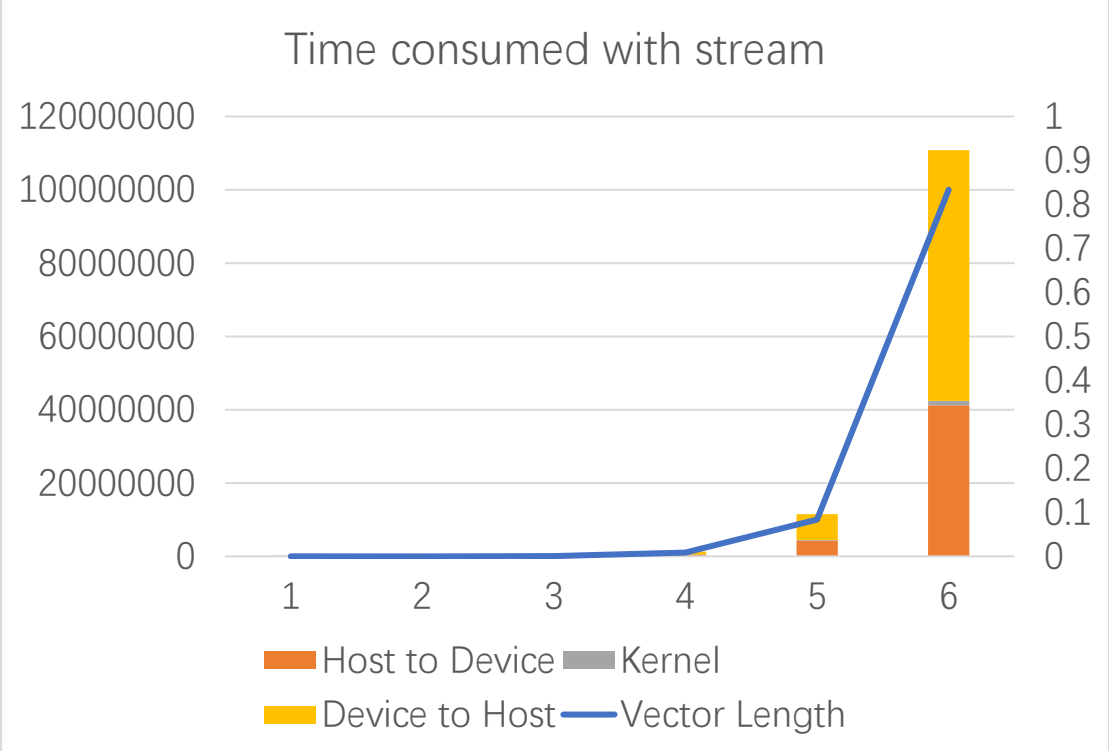
1.2 600 % 16 = 8, the right most blocks have divergence.
800 % 16 = 0, thus only the right most blocks have divergence.
Each warp is a 16*2 thread array.
800 / 2 = 400. There are 400 warps that have divergence.

1.3 600 % 16 = 8, the right most blocks have divergence.
799 % 16 = 15, the bottom most blocks have divergence.
Each warp is a 16*2 thread array.
15 % 2 = 1, each bottom most block has one warp that has control divergence.
Ceil(600 / 16) = 38. There are 38 bottom most warps with control divergence
Ceil(799 / 2) = 400. There are 400 right most warps with control divergence.
In total 38+400-1 = 437 warps with control divergence. (-1 is for bottom right corner overlap)

Exercise 2

2.1



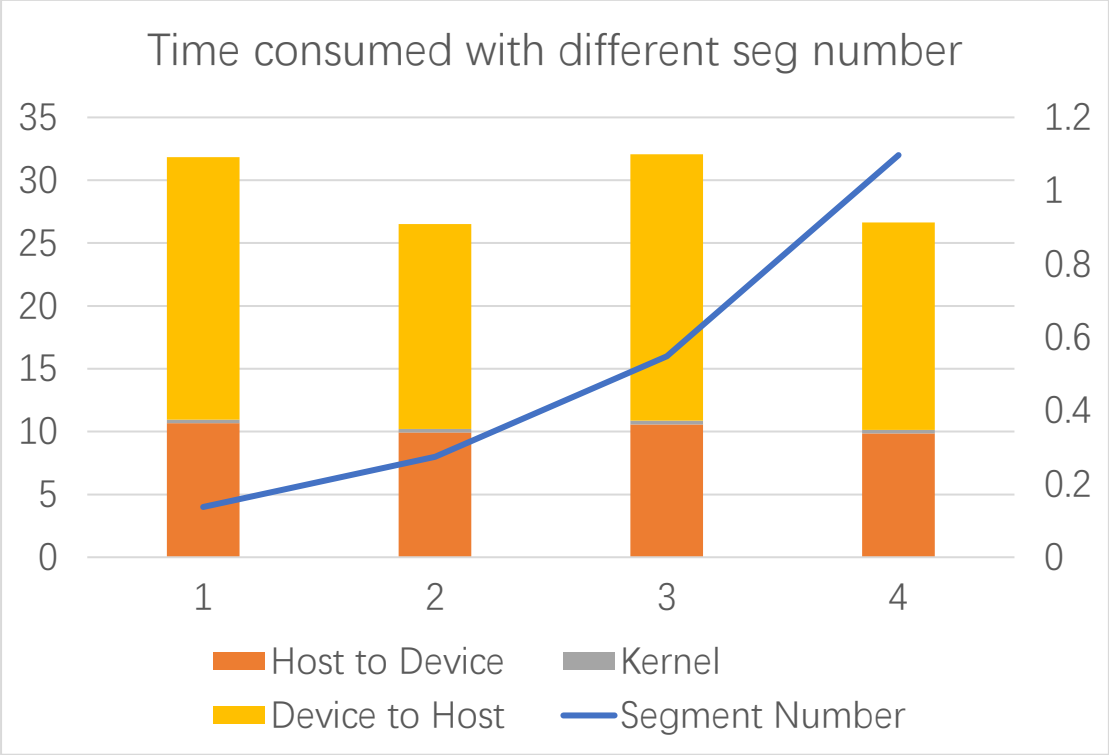


With the increment of vector size from 1000 to 100000000, the performance decrement becomes less and less. It is sensible to make a guess that with even bigger vector size, the performance will increase.

2.2

```
!nvprof --output-profile hw4_ex2_stream_profile.nvprof ./hw4_ex2
100000000 4
```

2.3



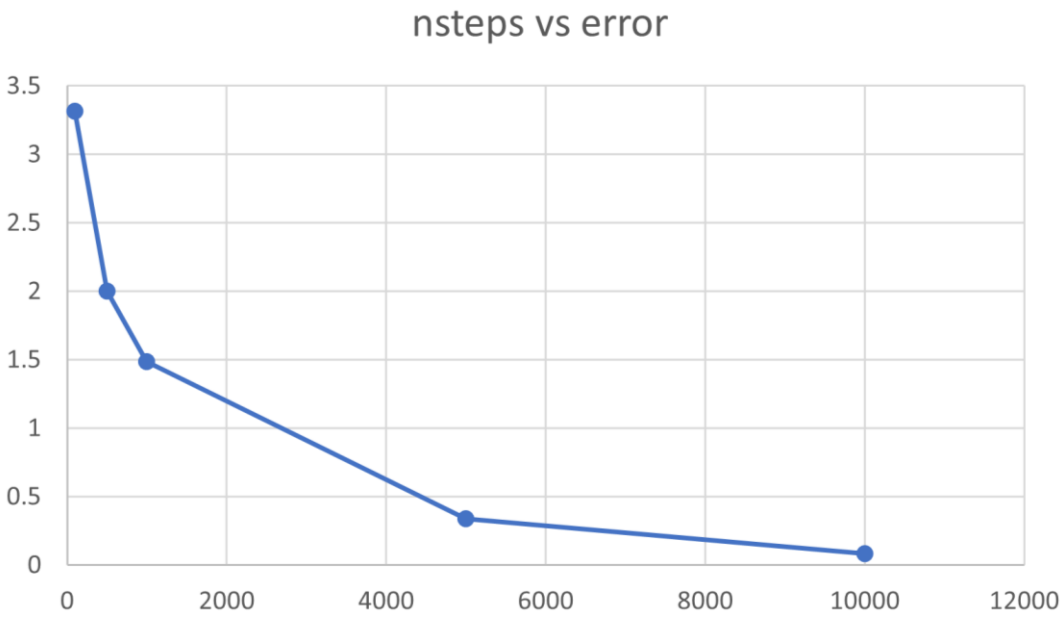
Here we use fixed vector size of 100000000 and different segment number of 4, 8, 16, and 32. No big difference is spotted.

Exercise 3

3.1

The sparse matrix-vector multiplication consists of 2*nzv = 6*dimX-12 FLOPs.
AXPY needs 2*dimX FLOPs.
Vector Norm Calculation requires 2*dimX FLOPs for squaring and summing.
In total there thus are 10*dimX-12 FLOPS per nstep.
Greater input size will increase the FLOP rate to approach the peak.

3.2 Higher nstep leads to lower error.



3.3 Performance with prefetching is shown as followed.

```
running ex3 with dimX=128 nsteps = 100, 500, 1000, 5000, 10000 with prefetching
The X dimension of the grid is 128
The number of time steps to perform is 100
Timing - Allocating device memory.              Elasped 174862 microseconds
Timing - Prefetching GPU memory to the host.            Elasped 526 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 5 microseconds
Timing - Initializing memory on the host.               Elasped 0 microseconds
Timing - Prefetching GPU memory to the device.          Elasped 387 microseconds
The relative error of the approximation is 3.318021
The X dimension of the grid is 128
The number of time steps to perform is 500
Timing - Allocating device memory.              Elasped 165599 microseconds
Timing - Prefetching GPU memory to the host.            Elasped 452 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 5 microseconds
Timing - Initializing memory on the host.               Elasped 0 microseconds
Timing - Prefetching GPU memory to the device.          Elasped 359 microseconds
The relative error of the approximation is 2.002595
The X dimension of the grid is 128
The number of time steps to perform is 1000
Timing - Allocating device memory.              Elasped 168551 microseconds
Timing - Prefetching GPU memory to the host.            Elasped 443 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 5 microseconds
Timing - Initializing memory on the host.               Elasped 0 microseconds
Timing - Prefetching GPU memory to the device.          Elasped 439 microseconds
The relative error of the approximation is 1.488118
The X dimension of the grid is 128
The number of time steps to perform is 5000
Timing - Allocating device memory.              Elasped 182679 microseconds
Timing - Prefetching GPU memory to the host.            Elasped 495 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 5 microseconds
Timing - Initializing memory on the host.               Elasped 0 microseconds
Timing - Prefetching GPU memory to the device.          Elasped 412 microseconds
The relative error of the approximation is 0.337648
The X dimension of the grid is 128
The number of time steps to perform is 10000
Timing - Allocating device memory.              Elasped 177807 microseconds
Timing - Prefetching GPU memory to the host.            Elasped 502 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 6 microseconds
Timing - Initializing memory on the host.               Elasped 0 microseconds
Timing - Prefetching GPU memory to the device.          Elasped 300 microseconds
The relative error of the approximation is 0.082647
```

And here is the performance without prefetching.

```
running ex3 with dimX=128 nsteps = 100, 500, 1000, 5000, 10000 without prefetching
The X dimension of the grid is 128
The number of time steps to perform is 100
Timing - Allocating device memory.              Elasped 204552 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 452 microseconds
Timing - Initializing memory on the host.               Elasped 0 microseconds
The relative error of the approximation is 3.318021
The X dimension of the grid is 128
The number of time steps to perform is 500
Timing - Allocating device memory.              Elasped 209013 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 422 microseconds
Timing - Initializing memory on the host.               Elasped 0 microseconds
The relative error of the approximation is 2.002595
The X dimension of the grid is 128
The number of time steps to perform is 1000
Timing - Allocating device memory.              Elasped 206513 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 476 microseconds
Timing - Initializing memory on the host.               Elasped 0 microseconds
The relative error of the approximation is 1.488118
The X dimension of the grid is 128
The number of time steps to perform is 5000
Timing - Allocating device memory.              Elasped 206603 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 477 microseconds
Timing - Initializing memory on the host.               Elasped 1 microseconds
The relative error of the approximation is 0.337648
The X dimension of the grid is 128
The number of time steps to perform is 10000
Timing - Allocating device memory.              Elasped 190143 microseconds
Timing - Initializing the sparse matrix on the host.            Elasped 482 microseconds
Timing - Initializing memory on the host.               Elasped 1 microseconds
The relative error of the approximation is 0.082647
```

With prefetching the time cost is reduced obviously.