An Easy Guide to Poor Performance

Mentor: Brent Leback

CAPSL: José Monsalve, Jaime Arteaga, Stéphane Zuckerman

What We Started From

- 5-point stencil code written for the GPU (Cuda)
- Uses a source and a destination pointer
- Swaps the source and destination pointers on the host (on purpose)

```
nvprof ./StencilCUDA 1000 1000 30
==7706== NVPROF is profiling process 7706, command: ./StencilCUDA 1000 1000 30
0.18940    ==7706== Profiling application: ./StencilCUDA 1000 1000 30
==7706== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
91.18%  1.70521s       600  2.8420ms  2.8386ms  2.8457ms  gpu_stencil2D_4pt
 4.58%  85.660ms        40  2.1415ms  1.9252ms  3.6531ms  [CUDA memcpy HtoD]
 4.23%  79.187ms        40  1.9797ms  1.5725ms  3.9795ms  [CUDA memcpy DtoH]
```

What We Started From

```
// Copy to block shared memory
for ( i = 0 ; i < TILE_SIZE ; i++ ) {
    for ( j = 0 ; j < TILE_SIZE ; j++ ) {
        int globalIndex = HALO*N + blockIdx.x*blockDim.x*TILE_SIZE*N
                  +  threadIdx.x*TILE_SIZE*N + i*N+blockIdx.y*blockDim.
                     y*TILE_SIZE
                  +  threadIdx.y*TILE_SIZE + j + HALO;

        int shMemIndex = HALO*smColDim + threadIdx.x*smColDim*TILE_SIZE
                  +  i*smColDim + HALO + threadIdx.y*TILE_SIZE + j;
        shSrc[shMemIndex] = src[globalIndex];
    }
}
```

Wait! There’s more!

```
if (threadIdx.x == 0 && threadIdx.y == 0 ) {
     int indexTopHalo, indexBottomHalo, indexLeftHalo, indexRightHalo;
     for ( i = 0 ; i < HALO ; i++ ) {
          for ( j = 0 ; j < smColDim ; j++ ) {
               indexTopHalo = (blockIdx.x*blockDim.x*TILE_SIZE+i)*N
                              + (blockIdx.y*blockDim.y*TILE_SIZE) + j;
               indexBottomHalo = (HALO + (blockIdx.x+1)*blockDim.x*TILE_SIZE)*N
                              + (blockIdx.y*blockDim.y*TILE_SIZE)+j;

               shSrc[i*smColDim+j] = src[indexTopHalo];
               shSrc[(HALO+blockDim.x*TILE_SIZE+i)*smColDim + j] = src[indexBottomHalo];
     }}
     for ( i = 0 ; i < HALO ; i++ ) {
          for ( j = 0 ; j < smRowDim-HALO*2; j ++ ) {
               indexLeftHalo = (HALO+blockIdx.x*blockDim.x*TILE_SIZE+j)*N
                              + (blockIdx.y*blockDim.y*TILE_SIZE)+i;
               indexRightHalo = (HALO+blockIdx.x*blockDim.x*TILE_SIZE+j)*N
                              + ((blockIdx.y+1)*blockDim.y*TILE_SIZE)+HALO+i;
               shSrc[(HALO+j)*smColDim+i] = src[indexLeftHalo];
               shSrc[(HALO+j+1)*smColDim-HALO+i] = src[indexRightHalo];
}}}
```

Most Glaring Problems

- Copy from memory to block shared memory: need to restructure the code to make have unit strides
- Processing the halo: need to spread the work to *all* threads (not just thread 0)