Mandelbrot CUDA Optimizations

By Benjamin Tiffany, Richard Qin, Jiawei Zheng

Serial/OpenMP

- 512x512, maxIter=1000
- Pixels near the center tend to take longer to render → dynamic scheduling
- Collapse nested loop into 1 loop
- gfx functions can have race conditions

```
# pragma omp parallel for num_threads(thread_count) \
        schedule(dynamic) collapse(2)
for(j=0;j<height;j++) {
        for(i=0;i<width;i++) {

# pragma omp critical
gfx_color(gray,gray,gray);

// Plot the point on the screen.
# pragma omp critical
gfx_point(i,j);
```

```
double start = omp_get_wtime();
// Display the fractal image
compute_image(xmin,xmax,ymin,yma
double stop = omp_get_wtime();
```

CUDA (GeForce RTX 2060)

```
Blocks: 512     Threads per Block: 512  Size:512x512     Depth: 1000     Time: 0.008425
```

CUDA Optimizations

- Approach 1 (Baseline): 1 block, 1 thread per block (N=1, M=1).
  - Simulate serial version of CUDA mandelbrot
- Approach 2 (Extreme): 1 block, 512x512 threads per block.
  - Each thread is assigned one pixel to render.
- Approach 3 (Extreme): 512x512 blocks, 1 thread per block.
  - Each block is assigned one pixel to render.

```
Blocks: 1       Threads per Block: 1    Size:512x512    Depth: 1000     Time: 7.793414
  Blocks: 1       Threads per Block: 1024 Size:512x512    Depth: 1000     Time: 0.338495
  Blocks: 262144  Threads per Block: 1    Size:512x512    Depth: 1000     Time: 0.187520
```

CUDA Optimizations

- Warp Size: 32
  - When threads%32!=0, have to create a new warp for leftover threads.
- Approach 4: vary number of threads per block in multiples of 32, keeping blocks fixed.
- Approach 5: vary pixels per thread with larger resolutions

CUDA Optimizations

- https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#device-side-kernel-launch
- Approach 6: Shared Memory
  - kernel_name<<< Dg, Db, Ns, S >>>([kernel arguments]);
  - “Ns is of type size_t and specifies the number of bytes of shared memory that is dynamically allocated per thread block”.
- Approach 7: Multiple Kernels
  - Separate line “mandel_kernel<<<n, m>>>(dev_counts, xmin , ymin, step, max_iter, dim, colors);” into n kernel calls and n separate streams.
- Approach 8: math_functions.h, device_functions.h
  - CUDA math functions
  - More speed for slightly lower accuracy.