Eileen Evans
Phoebe DeVries

Verification

We wrote two verification codes for the advs CUDA kernel. These two scripts (one calls the serial advs function, the other the advs CUDA kernel) output the strains calculated using dummy inputs, just to verify that they are the same. We have also verified the python serial advs function against the original Matlab code for a range of inputs (Meade, 2007).

Once we verified the code for step 1, it was easy to verify the code for steps 2 and 3 against the code from step 1.

Performance

We exceeded our performance goals by two orders of magnitude. We thought inititially that we would be thrilled with a two-time speedup. Instead, a calculation that would take about 1 hour in serial now takes only 5 seconds with our parallelized code. **This is a speedup of about 800.**

In our CUDA kernel, all of our reads and write are coalesced, and we make sure to pad our input values to make them multiples of 32. In this way, we make sure that no warps are only half in use.

The following figures show the run times for the serial code and the three versions of the parallel code (note that the y-scale is logarithmic). The run times for the serial code for the "final" data inputs are not included; these run times would have been on the order of about 10 days, so we did not have time to run this serial code. The total run time for the CUDA kernel execution alone (and the execution time for the analogous serial code) is shown in Figure 1. The total run time for the CUDA kernel including the time to allocate and transfer data to the GPU is shown in Figure 2. And finally, the total run time of each code set is shown in Figure 3.

In all cases, the more parallelized versions of the code run faster than the less parallelized versions. Step 1 is faster than the serial version, step 2 is faster than step 1, and step 3 is faster than step 2. For reference, tables of these run time values are shown below as well for each problem size.

Our maximum speedup, going from the serial code to our fastest parallel code, step 3, for our medium input size, was 795. The speedups for the different versions of the code and the different test problem sizes are shown in Figure 4. It is clear that for our test data inputs, our speedups increase with problem size (the overhead of data transfer to the GPU becomes less and less significant as we increase our problem size).

In Figure 5, the speedups for our largest data inputs (final) are shown alongside the test data speedups. We got a speedup of 85 with this largest data set. This smaller speedup is because in our final data input, we had many more triangles (~3000 triangles) than we did in our test data sets (~100 triangles). However, a speedup of 85 is still well beyond our performance goal of a speedup of 2!

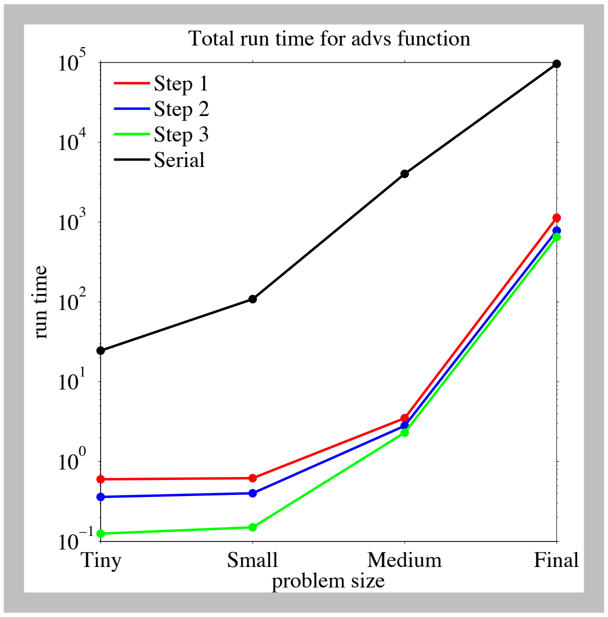

FIGURE 1. Total time spent in the advs kernel only (and the equivalent serial function).

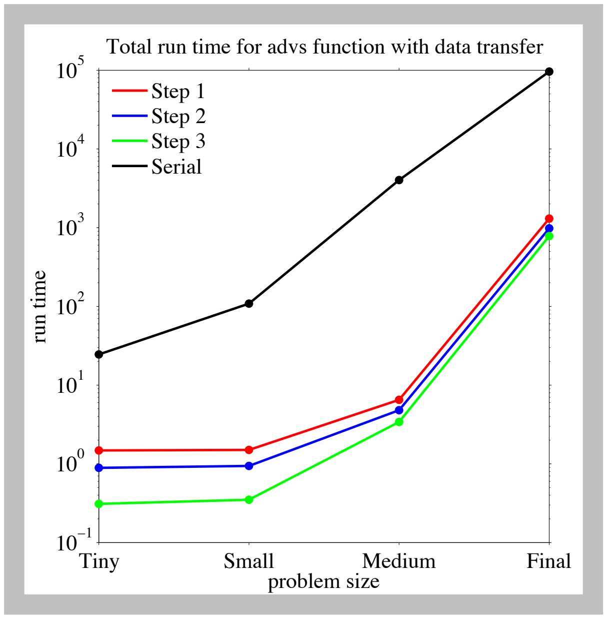

FIGURE 2. Total time spent in the advs kernel including setup and transfer of data to the GPU (and the equivalent serial function).

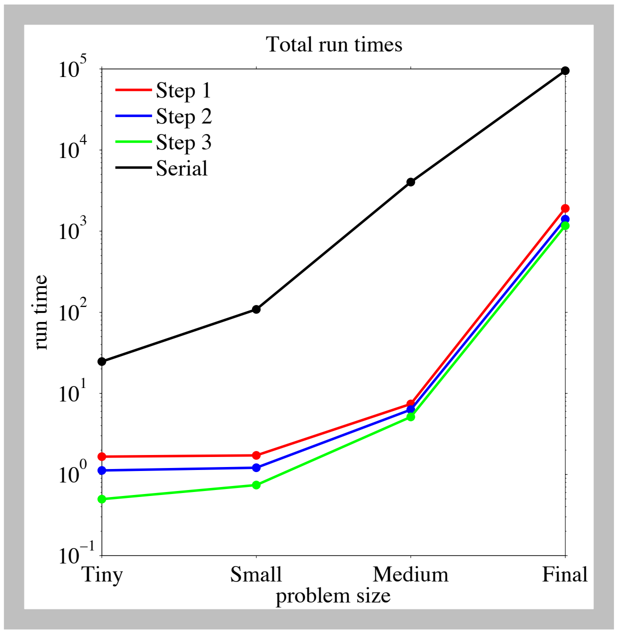

FIGURE 3. Total runtimes of sets of codes.

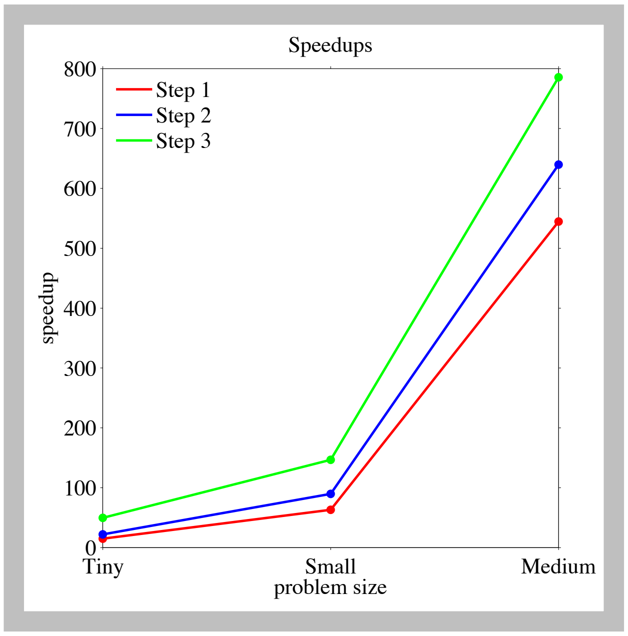

FIGURE 4. Speedups of the three parallel codes for each input test data set. Our fastest speedup was ~800, with our largest test data set. The speedups increase with problem size as the overhead of transfering data to and from the GPU becomes less and less significant.

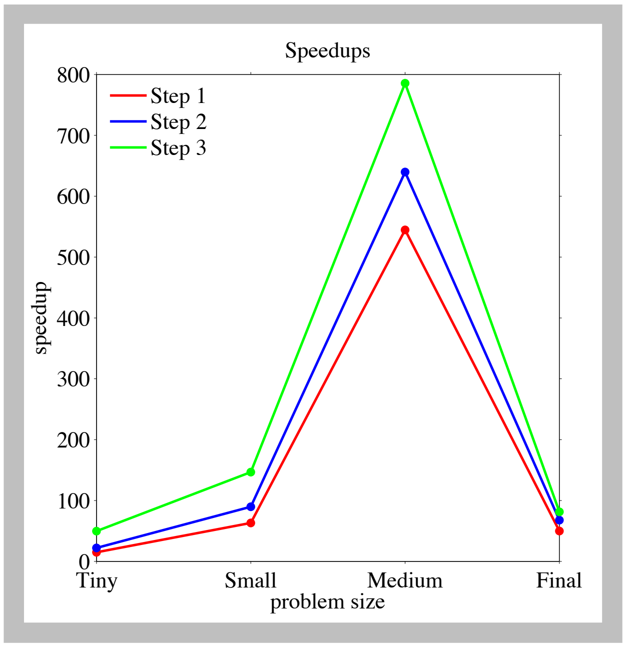

FIGURE 5. Speedups of the three parallel codes for each input data set. Our final data set has a slower speedup than our medium data set because it has many more triangles. However, a speedup of 85 for our final data set still exceeds our performance goals.

TABLES OF RUNTIMES FOR EACH INPUT DATA SIZE

RUNTIMES FOR TINY DATA INPUTS (G = [60 x 111])

	Kernel only	Kernel + Setup	Total run time
Serial	24.54 s	N/A	24.7 s
Step 1	0.6	1.48	1.66
Step 2	0.36	0.89	1.12
Step 3	0.125	0.31	0.53

RUNTIMES FOR SMALL DATA INPUTS (G = [24000 x 111])

	Kernel only	Kernel + Setup	Total run time
Serial	108.41 s	N/A	108.9 s
Step 1	0.62 s	1.5 s	1.72 s
Step 2	0.4 s	0.94 s	1.21 s
Step 3	0.15 s	0.35 s	0.74 s

RUNTIMES FOR MEDIUM DATA INPUTS (G = [300000 x 330])

	Kernel only	Kernel + Setup	Total run time
Serial	$4x10^3$ s (11 hours)	N/A	$4x10^3$ s (11 hours)
Step 1	3.5 s	6.5 s	7.4 s
Step 2	2.8 s	4.8 s	6.3 s
Step 3	2.3 s	3.4 s	5.13 s

RUNTIMES FOR FINAL DATA INPUTS (G = [300000 x 7863])

	Kernel only	Kernel + Setup	Total run time
Serial	27 hours ($9.5x10^4$s)	27 hours	27 hours
Step 1	1135.4 s	1302.9 s	1905.6 s
Step 2	780.4 s	980.5 s	1403.4 s
Step 3	651.7 s	781.6 s	1166.7 s