

Practical 4: reduction operation

The main objectives in this practical are to learn about:

- how to use dynamically-sized **shared** memory
- the importance of thread synchronisation
- how to implement global reduction, a key requirement for many applications
- how to use shuffle instructions

What you are to do is as follows:

1. Read through the `reduction.cu` source file and note the following:
 - The main code computes the results using both the CPU and the GPU. The CPU code is very simple, whereas the GPU code is much more complex.
 - Try to understand the `reduction` kernel completely.
 - The kernel uses dynamically allocated shared memory; the size is a third argument in the `<<< >>>` brackets.

You don't need to comment on any of this in your report.

2. Compile and run the executable `reduction`, and check that it gets the correct result. Put the output in your report and explain why this shows the result is correct, and how the code has performed the required check.
3. The code currently assumes the number of threads is a power of 2. Modify the code to handle cases in which it is not a power of 2, and test it with 192 threads.

To help you with this, on the course webpage I have provided a little program which demonstrates 3 different ways of rounding `n` up to the nearest power of 2.
4. The code currently performs the reduction operation for a single thread block. Modify the code to perform reduction using 1000 blocks each with 512 threads, with each block working with a different section of an input array of size 512000.

As explained in Lecture 4, there are two ways in which the partial sums from each block can be summed:

- each block puts its partial sum into a different element of the output array, and then these are transferred to the host and summed there;
- an atomic addition is used to safely increment a single global sum.

Do one of these, and in your report include your code, and results to show that the calculation has been carried out successfully.

5. Modify the block-level reduction to use shuffle instructions as described in Lecture 4. Again your report should include your code, and results to show that the calculation has been carried out successfully.