GPU Programming 2017/18

Tutorial 1

In this tutorial we will recapitulate memory management and consider how memory access patterns can affect performance on the CPU.

1.) Download the skeleton code and generate the build system using `cmake` (Under Linux and MacOS you can use a package manager (apt, brew, port,...) to install `cmake`. Then type `cmake .. in a directory ./build` on the command line to generate the make file. On Windows `cmake -G ''Visual Studio 12'' .\build` generates a Visual Studio solution).

2.) Allocate an array `data` for storing `data_size * data_size` elements of type `int` on the heap using `malloc()`. Ensure that the program correctly cleans up the memory.

3.) Initialize the memory so that the i-th element of `data` has value i.

4.) In the following we interpret the array `data` as being two-dimensional of size `data_size` $\times$ `data_size`. Implement that each element of the array is squared by traversing `data` with two nested loops

   i.) in row-major order;

   ii.) in column-major order.

   (A pragma to switch between the modes, as in the skeleton code, is sufficient.)

5.) Measure the performance for the part implemented in 3.) for arrays of size 2^j with $j = 3 \cdots 10$ for column- and row-major order, respectively. (It might be convenient to adjust the variable `k` that controls the number of repetitions of the experiment for this task). Interpret the graph.

Please finish the implementation until next week (week of 25/10/2016).

Solution: An example plot is provided in Fig. 1.

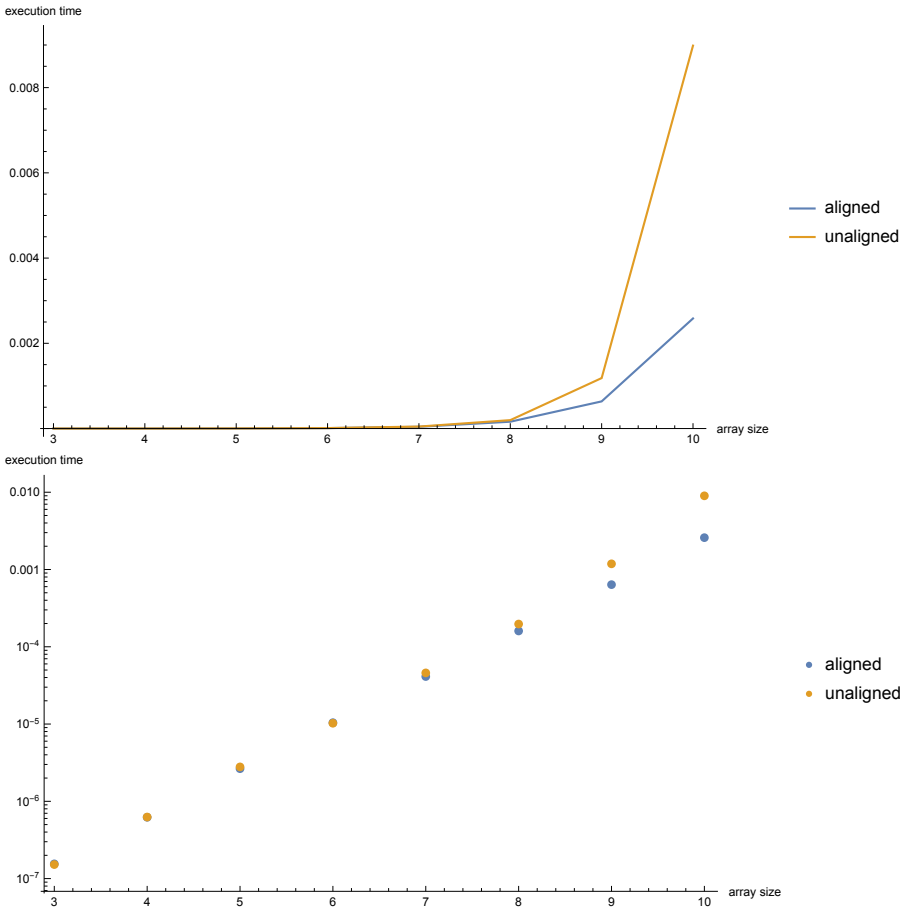

Figure 1: Execution times on an i7 processor as linear (top) and log (bottom) plot.