Final Project: Flexible-order finite difference computations

Ph.D. Course 2010: Scientific GPU Computing

This projects aims at familiarizing yourself with the parallel computing using CUDA on a heterogenous CPU-GPU system by a practical implementation of finite difference approximations of derivatives of a function.

Consider the general formula for flexible-order finite difference approximations of the q'th derivative of a function $f(x)$ in one space dimension

$$\frac{\partial^q f}{\partial x^q} \approx \sum_{n=-\alpha}^{\beta} c_n f(x_{i+n}) \tag{1}$$

where c_n is finite difference coefficients which can be computed using the supplied C function `fdcoeffF.c` and the function $f(x)$ is evaluated at a discrete grid $x_i = hi$, $i = 0, 1, ..., N-1$, with uniform spacing between grid points of size $h = \frac{1}{N-1}$. α and β are integer values indicating the number of points, respectively, to the left and right of the expansion point x_i. Take $\alpha = \beta$ for all interior points sufficiently far from the boundaries. Near the domain boundaries at x_0 and x_{N-1} the stencils will need to be off-centered.

- Familiarize yourself with the supplied sequential code for computing approximations of the q'th derivative on the discrete grid with N spatial points in one space dimension on a CPU.
- Do an analysis of how you can balance data-transfer and thread execution to maximize throughput performance for a given number of grid points N.
- Write a parallel version of the sequential code using the CUDA programming model to investigate the potential for speeding up the computations.
- Carry out performance tests to test and demonstrate how throughput can be maximized for various sizes of stencils with rank $r = \alpha + \beta + 1$ for sizes $r = 3, 5, 7, ...$ For example, choose $N = 640000$ in your tests. Reports timings (speedup), throughput (GFLOPS/s) and other interesting performance indicators.
- If time permits, extend the parallel code to be able to compute the same one-dimensional derivative approximation for each point on a grid in two space dimensions $(x_i, y_j) = (ih, jh)$, $j = 0, 1, ..., N$.

The answers should be given in a short report containing your analysis, results and conclusions. The final code should be included in an appendix and include sufficient comments to understand your program code.