GeAccKL: Toward a GPU numerical kernels library for geosciences

Alan Richardson and Chris Hill
Earth, Atmospheric, and Planetary Sciences, MIT
alan_r@mit.edu, cnh@mit.edu

Introduction

Geoscience codes often contain similar elements as they usually solve comparable equations, frequently using the same methods.

Examples of such common building blocks include:

Transport equation $\frac{\partial \varphi}{\partial t} = \vec{u} \nabla \varphi + \varphi \nabla . \vec{u}$

Wave equation $\frac{\partial^2 \varphi}{\partial t^2} - c^2 \nabla^2 \varphi = F$

Diffusion equation $\frac{\partial \varphi}{\partial t} = \nabla \cdot k \nabla \varphi$

Population equation $\frac{dP}{dt} = uIP \frac{N}{N + K_s} - gPZ - mP$

Solving these systems of time dependent differential equations, such as by the use of iterative Krylov subspace and explicit time-stepping methods, is potentially very amenable to GPU acceleration as it involves parallelizable operations and a relatively high number of computations per data element transferred from the CPU to GPU.

This is being exploited to develop a library, GeAccKL (Geoscience Accelerated Kernels Library), that runs such kernels on GPU hardware. The library will allow geoscience developers to benefit from the improved performance obtainable by using GPUs with minimal effort.

In our presentation we will look at features of the library, an example kernel, and an example of use in a geoscience application.

Features of the library

The library will contain kernels that form core building blocks of geoscience codes. This comprises implicit solvers such as the Conjugate Gradient method, as well as basic operators like $\nabla \varphi$ and $\nabla \cdot \varphi$.

Mixed precision algorithms will be used to maximize GPU performance. Before the release of the 'Fermi' architecture this year, NVIDIA's GPUs that were capable of double precision calculations ('compute capability 1.3' devices) had eight times more single precision units than double precision, therefore performing as many of the operations as possible in single precision was critical for obtaining good performance.

Even on 'Fermi' GPUs there can still be a significant advantage in maximizing the use of single precision as geoscience codes are typically memory-bound. Since single precision data requires 32 bits compared to 64 bits for double precision, a performance improvement of up to two times is expected even on the 'Fermi' architecture when double precision operations are substituted with single precision. This effect can be seen in Fig. 1.

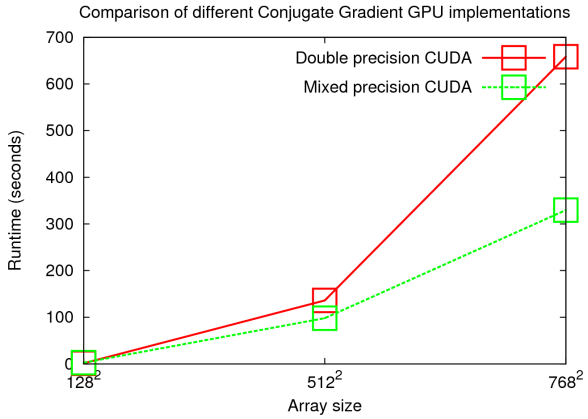


Figure 1: Performance comparison between a mixed and double precision Conjugate Gradient algorithm on a GTX 295 GPU

Due to the recent growth of interest in using GPUs, FPGAs, CELL processors, and other accelerators, which are typically optimized for single precision calculations, there have been several recent demonstrations of mixed precision algorithms, that is methods that perform many of their operations in single precision but still obtain a similar answer to methods that run entirely in double precision. Such algorithms are therefore likely to be well suited to GPU execution. This needs to be verified experimentally for each algorithm, however, as mixed precision methods frequently perform a larger number of operations than double precision algorithms.

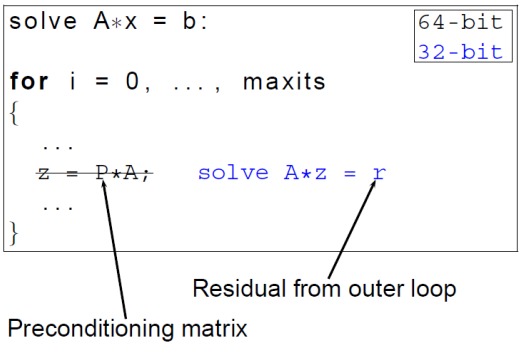
Another focus during library development is creating interfaces that are easy to use while also maintaining enough flexibility that the library can be employed in a diverse variety of codes. This target will be met with metaprogramming which allows dynamic code modification in order to suit the situation. This enables kernels to be optimized based on the particular circumstances in which they are being used. For example, if the A matrix in the CG algorithm is Toeplitz, then it is only necessary to transfer one element per non-zero diagonal. This reduces CPU-GPU transfer time and also the consumption of memory bandwidth on the GPU. It is envisaged metaprogramming will be implemented by designing the high level structure of the algorithms and creating a Python script that from this will generate code tailored to user inputs.

The target application areas include reservoir simulation,

ocean and atmosphere models, seismology, and geomorphology codes.

Example of a kernel: Conjugate Gradient

The Conjugate Gradient (CG) algorithm is a method of solving a system of linear equations of the form $Ax=b$ in $O(N)$ steps, where A is an $N \times N$ positive definite, symmetric matrix. A mixed precision version of the algorithm, based on the idea of iterative refinement, was developed by Göddeke [2] and separately by Buttari, Dongarra, et al. [1]. In this version of the algorithm an outer loop runs in double precision. The preconditioning step is replaced by a call to another CG solver that runs entirely in single precision and solves the equation $Az=r$, where r is the residual from the outer loop.



On one of the hardware configurations tested (an NVIDIA GTX 295 GPU and a 2 GHz Intel Xeon CPU) a performance improvement of up to 12 times was seen for the GeAccKL CG code compared to a CPU version (Fig. 2). As this GPU has a maximum power consumption of 289W, while the CPU requires 65W, this corresponds to a 2.6 times improvement in power efficiency.

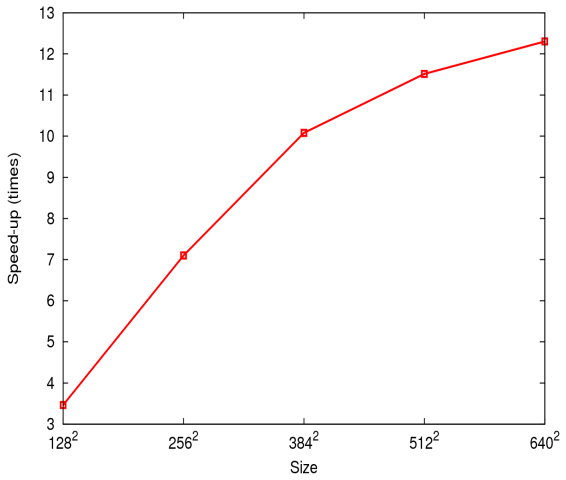


Figure 2: Speed-up of Conjugate Gradient method on GPU compared to CPU

Example of an application: MITgcm

The MITgcm (MIT General Circulation Model) is a numerical ocean, atmosphere, and climate model. It contains both 2D and 3D CG solvers. We have applied the GeAccKL CG solver to this code for a configuration where the CPU CG solver is responsible for 45% of the total runtime.

Accelerating the code using the CG method in the GPU library required integrating with a large legacy code. This will be typical of the situations in which the GeAccKL library is expected to be used.

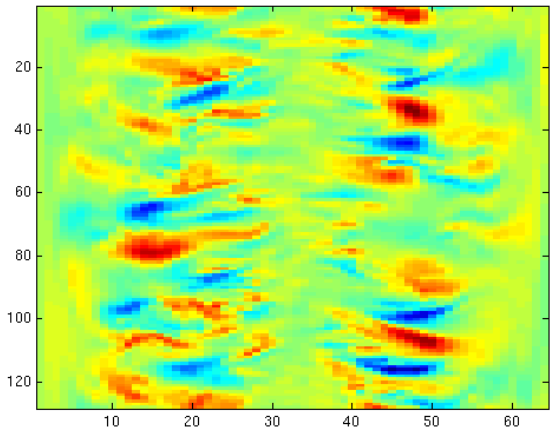


Figure 3: The figure shows instantaneous surface pressure patterns from a 4 month global atmospheric simulation using the GeAccKL conjugate gradient solver

The impact on the overall performance of the code for different problem sizes and on different GPUs will be presented.

Future Developments

The new NVIDIA 'Fermi' architecture, which was recently released, contains a number of interesting features that will be relevant to the GeAccKL library. The most notable of these is the eight-fold increase in the number of double precision computation units compared to the previous generation, so that there are now an equal number of single and double precision units. As many geoscience GPU kernels are memory-bound, such an increase is unlikely to result in an eight times speed-up, however. The new GPUs also receive an increase in memory bandwidth, which, together with new caches that will reduce memory bandwidth demand, will certainly lead to improved performance. Results from Fermi GPUs will be presented.

References

[1] A. Buttari, J. Dongarra, et al. “Mixed precision iterative refinement techniques for the solution of dense linear systems.” *International Journal of High Performance Computing Applications*, 21(4):457, 2007.

[2] D. Göddeke, R. Strzodka, and S. Turek. “Performance and accuracy of hardware-oriented native-, emulated-and mixed-precision solvers in FEM simulations.” *International Journal of Parallel, Emergent and Distributed Systems*, 22(4):221–256, 2007.