Graphics Processing Units (GPU) acceleration on acoustic 3D simulations

Lionel Boillot

Graphics Processing Units (GPU) acceleration on acoustic 3D simulations

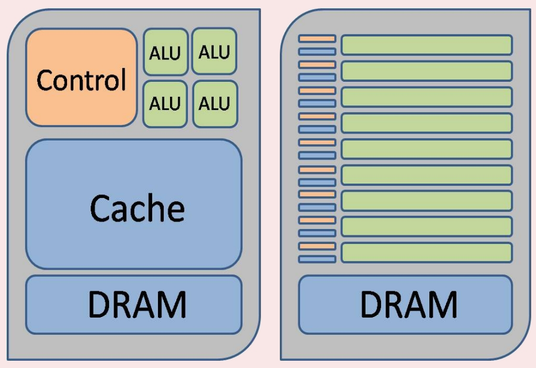
Lionel Boillot - OpenGPU project

Applied Mathematics and Systems laboratory (MAS), Ecole Centrale Paris, FRANCE

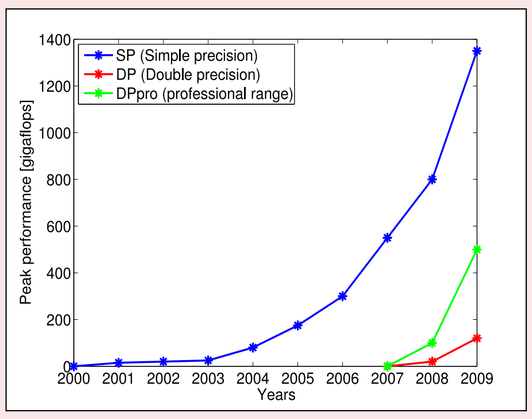
GPGPU

CPU are designed for sequential complex tasks treatments. On the contrary, GPU are built for rendering, which is comparable to massively parallel simple tasks computations: GPGPU (General-Purpose computing on GPU).

CPU & GPU architecture schemes reveal fundamental differences. On one hand CPU contain advanced control block, few ALU (Arithmetical and Logical Units), a big memory cache and a Dynamic Random-Access Memory (DRAM). On the other hand GPU are made up of basic control blocks, little local memory (manual cache), each pair associated with several ALU, and a global DRAM.



CPU GPU



GPU peak-performance evolution

During the last decade, GPU knew the same evolution than CPU's 10 years ago. The current computing peak perfomance is amazing: more than one teraflop for only one graphics card (i.e. 10^{12} floating-point operations per second). Moreover, the ratios price/performance and consumption/size between CPU and GPU are in favour of graphics cards.

Among the 5 most powerful supercomputers, 3 exploit GPU capabilities (1^{st}, 3^{rd} and 4^{th}), see the last "Top500" list at [1]. This demonstrates that GPGPU has become an essential tool for HPC (High Performance Computing), lots of examples in [2].

Acoustic simulations

Acoustic phenomena are represented by the wave propagation equation:

$$\rho \frac{\partial^2 u}{\partial t^2} - \mathrm{div}(\mu \nabla u) = f$$

where the right hand side f and the unknown u are scalar functions upon the space $\mathbb{R}^3$ and the time $\mathbb{R}^{+*}$.

Helmholtz equation is the harmonic case (i.e. time-independent):

$$-\Delta u - k^2 u = f$$

with Δ the Laplacien, u the amplitude of the wave and k the wavenumber ($k = \frac{\omega}{c}$ with ω the pulsation and c the celerity).

From physics equation, the problem is transformed to a linear system by the FEM (Finite Element Method).

$$Ax = b$$

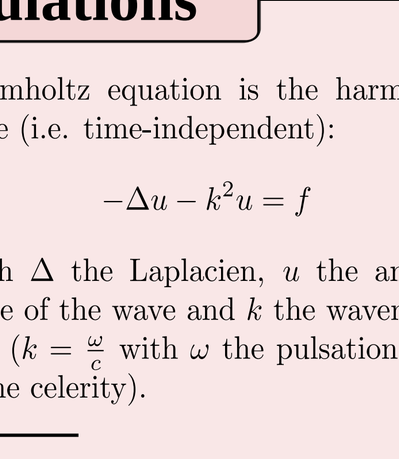
with A the matrix, b the right hand side and x the unknown vector.
Existence and unicity are ensured by correct boundary conditions on the discrete mesh.

Helmholtz problem matrix is sparse and non-symmetric. Krylov-based iterative methods are commonly used (direct LU is too expensive in 3D) based on BLAS (Basic Linear Algebra Subroutines):

$$x + \alpha y, \quad < x, y >, \quad Ax$$

Especially the matrix-vector products are the most expensive steps.

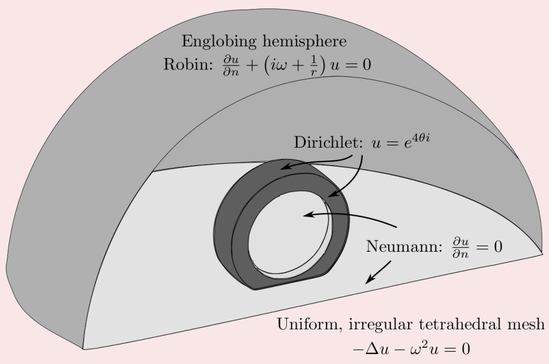
The goal is to accelerate Krylov algorithm by replacing BLAS CPU calls by BLAS GPU calls. Indeed these functions are directly parallelizable. Communications between CPU and GPU are too much expensive, so they are restricted to the minimum. Thus, the entire algorithm is run on the GPU.



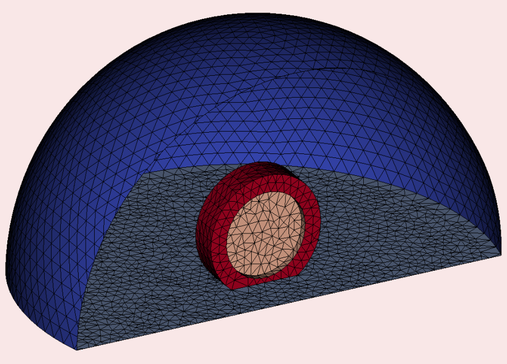
Krylov GPU flux diagram

GPU acceleration

Illustration test case is a 3D radiative problem (i.e. an exterior problem). It represents the modelisation of the air around a vibrating wheel along the ground. The hemispheric mesh is made up of tetrahedrons. This "3D tire case" was created by Cedric Venet, more details in [3].



3D tire case



Mesh skin

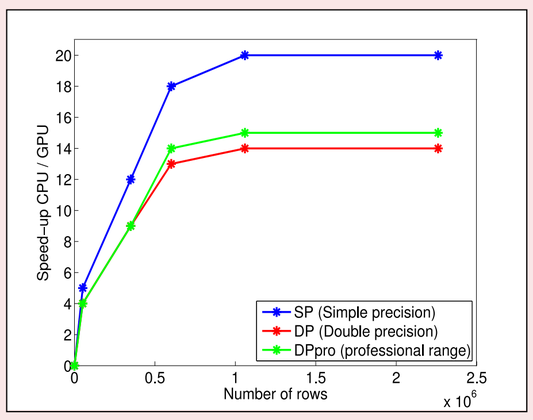
Kernels (i.e. GPU functions) are coded in CUDA (Compute Unified Device Architecture), which is the Nvidia C-based API for compiling and executing code, see [4]. Current opponent is OpenCL, specified by Khronos, see [5].

Basic kernels are very sub-optimal. GPGPU requires optimisations steps, summezised in [6]. In 2010, professional libraries released. Nevertheless, further researches have to be done to achieve better performance.

In order to measure GPU acceleration, CPU version and GPU version execution times are compared. For both architectures, compilation is "-O3" mode. Results below are obtained with the Nvidia libraries libCUBLAS and libCUSPARSE (speed-up devided by 1.5 with non-professional kernels).

Material description (2010):

- CPU: Intel Core-i7 930 (2.80 GHz) with 12 GiB of RAM
- GPU: Nvidia GTX 570 (1.5 GHz) with 1 GiB of RAM
- GPUpro: Nvidia M2050 (1.2 GHz) with 3 GiB of RAM



Speed-up CPU / GPU

Perspectives

Speed-up between CPU and GPU highly depends on the matrix structure and storage format, see [7]. Not only BLAS but most of parallelizable functions could be accelerated by GPU, and the speed-up could be very important. Moreover, double precision performance will meet simple's soon.

In few years, GPGPU has proved its usefulness in all scientific fields, see [8]. Furthermore, libraries are regularly improved, e.g. CUSP which has become the STL (Standard Template Library) in CUDA 4.0, or C++ wrappers in OpenCL 1.1, tending towards a complete C++ compatibility.

Bibliography

[1] Top500, Website, `http://top500.org`, 2011
[2] GPGPU, Website, `http://gpgpu.org`, 2011
[3] C. Venet, *Numerical methods for acoustic simulation of large-scale problems*, PhD thesis, Ecole Centrale Paris, 2011
[4] CUDA, Website, `http://nvidia.com/object/cuda_home.html`, 2011
[5] OpenCL, Website, `http://www.khronos.org/opencl/`, 2011
[6] P. Micikevicius, *CUDA Optimization*, Conference, 2009
[7] N. Bell & M. Garland, *Efficient Sparse Matrix-Vector Multiplication on CUDA*, NVIDIA Technical Report, 2008
[8] Nvidia GTC (GPU Technology Conference), 2009-2010