GPU users meeting #2

CERN, 11 March 2016

The aim of the meeting is to discuss our experience and future plans concerning the usage of GPUs for scientific computing at CERN and to define possible forms of collaboration with NVIDIA DevTech in this framework.

Present: Gianluigi Arduini, Thomas Bradley, Xavier Buffat, Riccardo De Maria, Giovanni Iadarola, Vincenzo Innocente, Michael Jonker, Sami Kama, Kevin Li, Peter Messmer, Elias Metral, Adrian Oeftiger, Felice Pantaleo, Maurizio Pierini, Giovanni Rumolo, Sofia Vallecorsa, Jeroen Van Nugteren, Jean-Roch Vlimant.

Activities of NVIDIA DevTech

Speaker: Thomas Bradley and Peter Messmer

NVIDIA's developer technology (devtech) team research and develop GPU computing algorithms in a number of application domains, working in conjunction with key application developers to ensure best possible performance of GPU computing applications on current and next-generation architectures. They also work with NVIDIA's architecture and software teams to influence the design of the future platform to solve tomorrow's scientific problems. Application domains include geosciences, life sciences, computer-aided engineering, computational fluid dynamics, computational chemistry, computational physics, computational finance, electronic design automation, data mining, machine/deep learning, medical imaging, and many more.
Alongside devtech, the developer relations contact is available to create appropriate connections into NVIDIA to support the various GPU projects at CERN.

Magnetic field simulations

Speaker: Jeroen Van Nugteren (Twente Technical University (NL))

A GPU accelerated code is used to simulate the behavior of research magnets based on high temperature superconductors. The code simulates the evolution of the currents, voltages and temperatures in the magnet, using the multi-level fast multipole method (MLFMM) to speed-up the computation of mutual inductance contributions. A large fraction of the computing time (70%) is spent in the solution of the factorized pre-conditioner (performed using the MUMPS).
It was stressed that having the hardware already available in the form of a video card in a PC, was one of the reasons for having started this project in the first place.

Discussion

P. Messner suggested getting in touch with the research group of Prof. Barba (http://lorenabarba.com/) who has done extensive work on FMM on GPU. He also mentioned that the upcoming GPU technology conference would be a perfect

occasion to meet people from the community.
It was asked whether the code allows profiting from multiple GPU cards. **J. Van Nugteren** answered that this is not possible yet, and it would require a significant implementation effort.
T. Bradley asked whether the matrix could be treated in blocks. **J. Van Nugteren** answered that this is not possible, since there is full coupling among the coils. The FMM method, on the other hand helps reducing the memory requirements.
V. Innocente asked whether the different linear system solutions can be run in parallel. **J. Van Nugteren** answered that the time derivatives make the problem fully serial with respect to time.
R. De Maria asked whether they use an adaptive time step. **J. Van Nugteren** confirmed that this is the case.
It was discussed if the AmgX library (https://developer.nvidia.com/amgx) could be effective in this case but whether their matrix falls in the hypothesis for the usage of this algorithm still has to be investigated
G. Iadarola asked whether a direct solver (e.g. LU factorization) has been considered as an option, since this proved to be very effective for band matrices used in e-cloud field calculations. **J. Van Nugteren** answered that this was unsuccessful.
J. Van Nugteren asked whether it is possible to profile CUDA code inside Matlab. **P. Messmer** and **T. Bradley** answered that nvprof allows to do so.
The question was raised on which are reasonable criteria to compare CPU vs GPU implementation (e.g. performance/cost vs. performance/energy) and it was stated that this really depends on the needs and means of the different research groups.

Beam physics simulations: SixTrack

Speaker: Riccardo De Maria (CERN)

SixTrack is a single particle tracking code for particle accelerator studies. It is written in Fortran 77/90 and designed to be numerically portable across operating systems and compilers. It is used in the volunteer computing project LHC@Home with 200k registered users and about 20k CPUs simultaneously running. In this framework each particle is tracked twice on two independent CPUs to identify errors introduced by faulty machines. Double precision is mandatory in this kind of studies.
At the moment a standalone tracking library (SixTrackLib) is being derived from SixTrack for usage within other simulation codes (e.g. PyHEADTAIL). The code is being written in C and OpenCL in order to be portable on different architectures (CPUs and GPUs from different vendors). The speed up obtained makes GPU solutions very attractive. For this kind of application, high FP64 FLOPS counts are desired while memory bandwidth and memory size are less important. Different solutions available on the market are compared according to these criteria.

Discussion:

P. Messmer asked whether it makes sense, instead of comparing two numerical results exactly, to do statistics with three or more results. **R. De Maria** answered that this is not really the case since they need to distinguish between chaotic behaviors coming from the physics and, for example, random memory bit flips.
R. De Maria clarified that redundancy has to be implemented in any case, so the ideal solution for them are low cost (gaming) cards, without Error-Correcting Code (ECC) memory and with a good double precision performance. **T. Bradley** answered that the trend from NVIDIA is to have double precision performance only on the professional cards (Tesla), which also do not suffer from overheating. For the next nVidia GPU generation they expect double precision performance comparable to their direct competitors.
P. Messmer asked whether it would make sense to assemble the accelerator model at compile time by using code generation. **R. De Maria** answered that in principle it would be possible but some flexibility for dynamically evolving elements would be lost.

Beam physics simulations: Particle-In-Cell

Speaker: Adrian Oeftiger (Ecole Polytechnique Federale de Lausanne (CH))

Particle-In-Cell (PIC) codes are used in the PyHEADTAIL and PyECLOUD simulation codes for the simulation of space charge and electron cloud effects in particle accelerators.
In space charge simulations the electromagnetic forces within the beam are calculated in free space. In these cases the Poisson equation can be solved via FFTs using the Hockney's algorithm. This case has been successfully ported on GPU using the cuFFT library as described here. For the particle-to-mesh deposition, sorted charge deposition has been implemented with a 3x speed-up over double precision atomics. Mesh-to-particle interpolation is performed using a custom implemented kernel.
For e-cloud simulations the presence of the conducting boundary of the beam pipe needs to be taken into account. In this case a finite difference algorithm is used to solve the Poisson equation and the Shortley-Weller approximation is used to minimize artifacts at the borders due to the curved boundary. The solution of the linear system is performed on CPU using the KLU library. The solution on GPU using the cuSOLVER library has been tested but no speed-up was observed. Obtaining a significant speed-up on this calculation would significantly ease the study of the LHC beam dynamics at high energy in the presence of e-cloud in the arcs. A sample package with a typical matrix (and the KLU solution algorithm for comparison) is available here.

Discussion
P. Messner commented that the Particle-To-Mesh and Mesh-To-Particles operations would have a significant speed-up if performed in single precision, due to the larger throughput and to the availability of atomics implemented in hardware.
A. Oeftiger pointed out that when he started working on this activity the

documentation of the cuSOLVER and cuSPARSE libraries was quite poor. **T. Bradley** answered that these libraries were included in CUDA very recently and the situation should sensibly improve with the next version.
T. Bradley asked whether numba was tested as an option. **A. Oeftiger** answered that numba has still been in a too early stage of development to be considered as a reliable solution for a production code at the time of the testing two years ago.
G. Iadarola asked whether pyCUDA will be supported and developed in the coming years. **T. Bradley** and **P. Messmer** answered that it is a very active open source development project.

LHC event filter applications

Speaker: Felice Pantaleo (CERN - Universität Hamburg)

F. Pantaleo illustrated the present and potential usage of GPUs for processing data from the LHC experiments. At the moment the main interest is focused on the use of GPUs at trigger level. The experience and needs of the different LHC experiments were discussed individually.

ALICE

GPUs are already extensively used by the ALICE experiment within their High Level Trigger (HLT). Their implementation aims at being independent from the particular hardware. Therefore the tracker is written in C++ and wrappers are used to execute the same code on different hardware platforms (AMD and NVIDIA GPUs but also conventional CPUs and Xeon Phi).

Discussion

F. Pantaleo asked, on behalf of ALICE, which is the roadmap of NVIDIA with respect to OpenCL and C++11. **T. Bradley** replied that OpenCL 1.2 is supported by NVIDIA and this will not change in the immediate future, the timescale for the introduction of OpenCL 2.0 is not defined yet. He added that C++ 11 is already supported by CUDA (e.g. lambda expressions can be used). In general he stressed that they are interested in knowing which are the features that developers perceive to be missing in their tools.
V. Innocente pointed out that they have a large amount of legacy CPU code with low performance requirements. They would be interested in running it on the GPU just as it is, even in a not optimized way, to avoid heavy data transfers from host to device. This is an example for which more advanced C++11 support is desirable.

ATLAS

At the moment GPUs are not used within the ATLAS event reconstruction software since the software runs on grid, with huge hardware diversity and almost no GPU available at the moment. However there are ongoing studies to offload part of the tasks to GPUs, using a server-client architecture to be able to operate within the grid. A “Trigger GPU demonstrator” is being developed.

Discussion

F. Pantaleo stressed that they could profit from NVIDIA support in defining strategies to increase the performance. **T. Bradley** and **P. Messmer** replied that they would be available to provide this kind of support.

F. Pantaleo reported a series of questions from the ATLAS developers:

- ATLAS would be strongly interested in using GPU within Virtual Machines. **T. Bradley** replied that this feature is presently not available.
- Can Maxwell cards be used for scientific computing? **T. Bradley** replied that they can be used and there are also professional versions (Tesla) with this architecture. However double precision performance is quite limited.
- Is there a public roadmap form scientific GPU cards? **T. Bradley** replied that NVIDIA's roadmap for new hardware should be defined at the upcoming GPU Technology Conference.
- OpenACC, when is it coming to GCC/llvm? Also the point was raised whether this tool is expected to be maintained in the long term. **T. Bradley** replied that it could happen at some point in 2016. Of course the evolution is hard to predict. On the other hand as long as R&D activities are concerned, the goal is rather to test the effectiveness of a strategy, more than make definitive choices on the tools to be employed. OpenACC will allow testing if parallelization through directives is suitable for a specific problem even if the implementation tool might have to be redefined in the future.
- Is there a way to configure Nsight for remote development? **T. Bradley** and **P. Messmer** answered that they can provide a script to do the configuration.
- Can we test pre-release hardware? **T. Bradley** replied that there is an early access program. In particular they have a small cluster where they provide pre-release access.

CMS

The CMS online farm consists of 16k Xeon cores and, at the moment, the tracks are not reconstructed for all events at the High Level Trigger. The situation will become even more difficult with larger pile-up. For this reason CMS is planning to replace half of the CPUs with GPUs to gain performance.

Growing interest is developing for the usage of Deep Neural Networks (DNN) to take fast decisions at trigger level (keep or reject the event) and for jet identification. DNN training naturally happens on GPUs, therefore the possibility of creating a O(50) GPU cluster at CERN is presently being investigated.

LHCb

For the LHC Run 3 starting in 2020, the LHCb experiment wants to operate with a full software trigger. For this purpose at the moment different architectures are being evaluated. The choice of the technology is planned in a one-year time, and by that time demonstrators will have to be built for the different parts of the high level trigger. Different projects in this direction are making use of GPUs. The final choice of the technology will be based not only on efficiency but also on integration,

scalability, maintainability and reproducibility. Their needs at the moment include mainly evaluation on state-of-the-art hardware, training, meetings with other GPU users.

GEANT V experience

Speaker: Philippe Canal (Fermi National Accelerator Lab. (US))

The Geant code is used for detailed simulation of subatomic particle transport and interactions in detector geometries. The Geant V project has the goal of improving the performance of Geant4 between 2 and 5 times with solutions that are portable on different architectures (GPUs and Xeon Phi's). The main tasks in the computation are the geometry calculations, the particle transport, and the particle physics processes.
GPUs have been used to speed up the geometry module (VecGeom) with very promising results. Preliminary results for the physics calculations on Xeon Phi have also been presented.
In the GPU implementation a "broker " adapts baskets of tracks to the coprocessors. Each thread is processing one track. The cost of data transfer is mitigated byoverlapping kernel execution and data transfer. At the moment a working broker is available for "geometry only". The next steps is to incorporate the physics code into the CUDA kernel to be able to run a full GPU accelerated prototype and identify performance issues and latency limitations.

Discussion
P. Canal raised the point that in the past the CUDA profiler was providing only information on the kernel as a whole, making quite difficult to identify bottlenecks in large kernels. **T. Bradley** answered that is already possible to have granular information within kernels and that these capabilities are getting more and more sophisticated with the most recent hardware.
P. Messner suggested investigating the impact of basket size on performance.
T. Bradley suggested using concurrent kernels to improve performance. **P. Canal** answered that this is already widely exploited.
P. Messmer asked whether the VecGeom library is available and can run outside Geant. **P. Canal** answered that this is the case.
It was asked how the computing load is distributed within the code. **P. Canal** answered that in a typical simulation they experience 30% geometry, 30% propagation, 30% physics.

Reported by G. Iadarola