ISBN: 978-3-95450-227-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2022-WEPOMS043

UFO, A GPU CODE TAILORED TOWARD MBA LATTICE OPTIMIZATION

M. Carlà*, M. Canals, ALBA-CELLS, Barcelona, Spain

Abstract

The complexity of multi-bend achromatic optics is such that computational tools performance has become a dominant factor in the design process a last generation synchrotron light source. To relieve the problem a new code (UFO) tailored toward performance was developed to assist the design of the ALBA-II optics. Two main strategies contribute to the performance of UFO: the execution flow follows a data parallel paradigm, well suited for GPU execution; the use of a just-in-time compiler allows to simplify the computation whenever the lattice allows for it. At the core of UFO lies a parallel tracking routine structured for parallel simulation of optics which differs in some parameters, such as magnet strength or alignment, but retains the same element order, reflecting the scenario found in optimization processes, or when dealing with magnetic or alignment errors. Such an approach allows to take advantage of GPUs which yield the best performance when running thousands of parallel threads. Moreover UFO is not limited to tracking. A few modules that rely on the same tracking routine allow for the fast computation of dynamic and momentum aperture, closed orbit and linear optics.

INTRODUCTION

Single particle tracking is at the base of many optimization tasks frequently encountered during the development and tuning of lattices for light sources. Dynamic and momentum aperture optimizations are typical examples that can take advantage of a fast single particle tracking routine. Other use cases that can also benefits from a fast tracking routine are closed orbit and linear optics function computation, required respectively for orbit correction and optics matching. Traditionally these optimization tasks are carried out on large computer clusters by means of parallel optimization algorithms such as MOGA [1, 2]. However in recent time, some projects [3, 4] moved the computational burden to dedicated arithmetic processors (GPU), providing competitive performances at reduced costs. While these projects aims to provide a rather general purpose optics tool in an easy to use environment, it is possible to improve substantially the performances by tailoring the code specifically toward the optimization of electrons ring lattices, renouncing to some flexibility and ease of use by taking advantage of two main points:

- Radiation damping limits the number of turns required for stable aperture computation to around 1000 turns
- The optimization procedure (e.g. MOGA) requires to compute several variation of the same optics, that can be computed in parallel on a GPU

* mcarla@cells.es

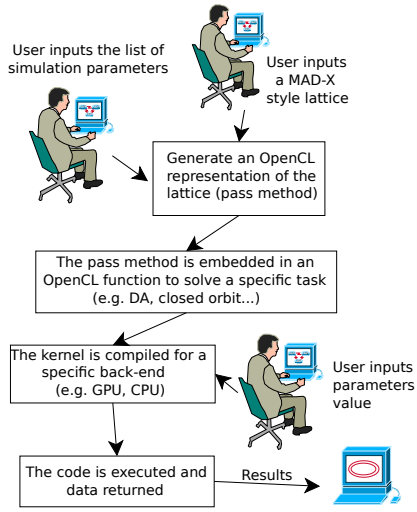

Figure 1: UFO working diagram.

Based on these assumption it was possible to develop an Utterly Fast 5D Optics code (UFO) [5] targeting electron rings optimization. The code was then characterized in terms of accuracy and performances by running simulations on a candidate multibend achromat lattice for ALBA2 [6] with alignment errors in each element.

EFFICIENT TRACKING ON GPU

The task of tracking thousands particles fits well the typical GPU architecture, where a single (or a small number) dispatch unit sends instructions to a group of computational units which therefore execute the same program in parallel. To ensure good performances code, no branching is allowed; otherwise different instruction flows would be required for each computational unit, a condition not allowed by the GPU architecture. In case of branching the different branches are executed sequentially resulting in performances penalties.

This requirement is easily met when tracking a bunch of particles through a lattice: every particle undergoes the same order of operation as any other, being the lattice identical for all of them. Also, when tracking particles through different variations of the same lattice, the condition is still met, as long as the order of the elements is maintained among the different lattice variations. The ability to simulate multiple lattices at the same time is of critical importance, in fact it would be difficult otherwise to keep a GPU with thousands of computational units under a high workload when dealing with a simulation of only 1000 particles for 1000 turns, as in the case of a typical electron dynamical aperture simulation.

Figure 1 shows the execution flow of UFO. The user is required to input a lattice (in a MAD style format) and a list of parameters (e.g. particle coordinates, magnet length, strength...). UFO generates an internal representation of

ISBN: 978-3-95450-227-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2022-WEPOMS043

the lattice integrator in OpenCL language [7]. Then, the integrator is embedded in a function specific for the task to be carried out (e.g. dynamic aperture, closed orbit...) and compiled for a specific GPU or CPU back-end. Finally, the user invokes the execution of the task, and only at this time the value of each parameter is specified independently for each particle.

Compiling the integrator at run-time provides a chance for some optimizations that otherwise would be complicated to implement. An example is the case of a succession of linear elements (drifts and quadrupoles) that most modern compilers are able to optimize as a single linear transformation, or the precomputation of expressions that depends on constants only. This is the case when the parameters of an elements are not specified as per particle parameters and therefore part of the integrator can be calculated *a priori*. Note that while compiling the integrator is a time consuming process it is required only once.

TRADING ACCURACY FOR SPEED

Since UFO aim to provide accurate results for short time integration (i.e. 1000 turns), it is possible to trade some accuracy to speed-up even more the computation, this is achieved through two devices: by using an approximated integrator and by using a 32 bit floating point representation of variables.

Approximated Integrator

Different approaches has been developed to take into account relativistic effects. For example MAD-X/PTC approximates thick elements with a succession of thin ones interleaved by drifts that can be solved with an exact integrator. This approach allows for very accurate results at cost of performances, in fact an accurate solution is achieved only with a very fine slicing. The effect of using a purely classical integrator has been characterized by running a typical DA simulation, the results have been compared against the one provided by MAD-X/PTC with the exact relativistic Hamiltonian option switched on. As shown in Figure 2, a discrepancy between the two codes is visible but limited. About 5% of the stable particles are incorrectly tracked for the on-energy case, increasing to 10% in the off-energy case.

32- vs 64-Bit Variables Representation

GPU hardware is optimized mainly for integer and 32-bit floating point computations, therefore by strictly sticking to a 32-bit variables representation, it is possible to obtain a considerable performance gain at cost of very mild accuracy penalties. Figure 3 shows the comparison of the dynamic aperture computed with 32 and 64 bit variables, in this case 3% of the stable particles are incorrectly tracked when switching to 32 bit.

It is worth noting that while most of the CPU follows strictly the IEEE-754 floating point standard this is not the case for GPU which usually tend to use less accurate approximation in order to speed up computation, therefore some minor difference between CPU and GPU computations can be observed even when using the same variable format.

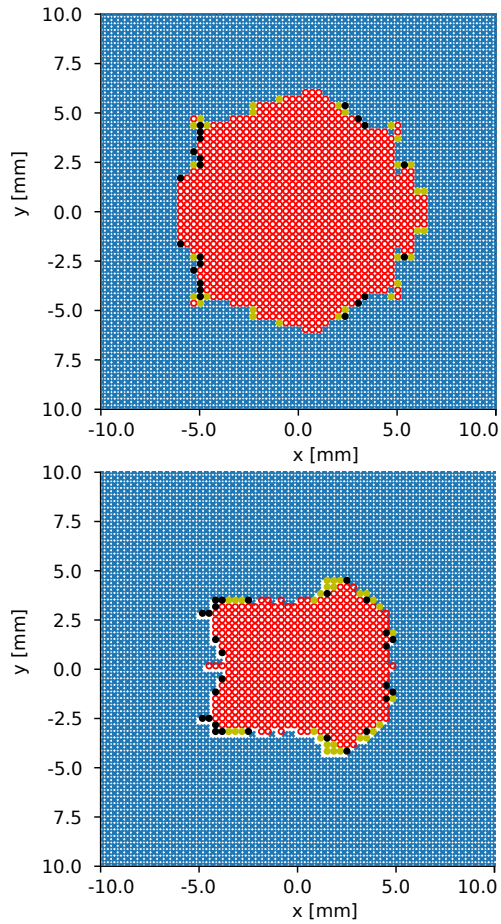

Figure 2: Comparison of on-energy (top) and 3% off-energy (bottom) dynamic aperture computed with UFO and MAD-X/PTC with exact relativistic Hamiltonian. Each marker shows the initial transverse coordinate of an electron. Electrons found stable or unstable by both codes are shown respectively as red or blue markers, electrons stable in MAD-X/PTC and unstable in UFO are shown in yellow, while black marker are the ones stable for UFO but unstable for MAD-X/PTC.

Table 1: Four OpenCL Backends Under Test

	Base clock	Cores
Intel i5-8400	2.8 GHz	6
Intel Xeon Gold 6136	3.0 GHz	24
Nvidia Quadro P600	1329 MHz	384
Nvidia Tesla T4	585 MHz	2560

BENCHMARKING

Four different hardware configurations has been used to test UFO performances, including high-end and low-end GPUs and CPUs. The main characteristics of the 4 configurations are shown in Table 1.

ISBN: 978-3-95450-227-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2022-WEPOMS043

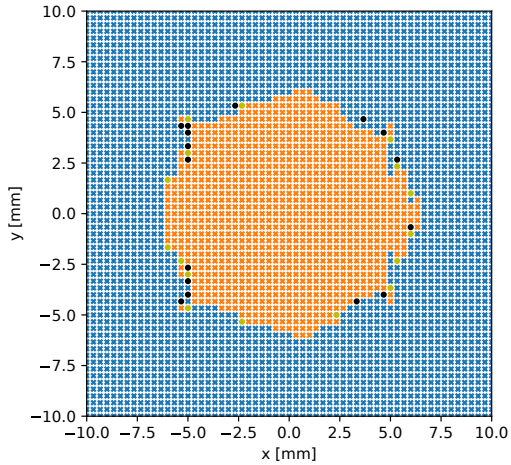

Figure 3: Comparison of dynamic aperture computed with UFO using 32 or 64 bit floating point format. Electrons found stable or unstable in both simulations are shown respectively as red or blue markers, electrons stable in the 32-bit simulation and unstable in the 64 one are shown in yellow, while black marker are the ones stable for 64 bit but unstable for 32 bit.

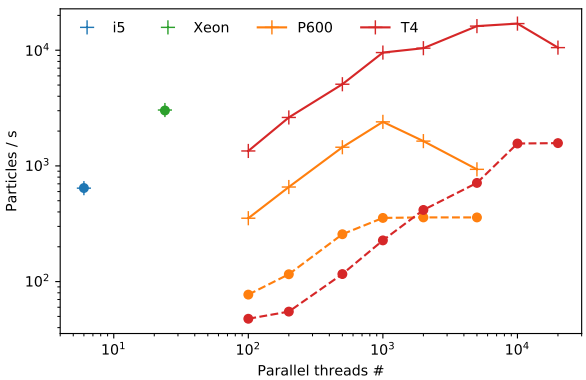

Figure 4: Dynamic aperture benchmark. Different number of particles are tracked in parallel on the 4 different hardware configurations. The test is repeated using 32 (solid lines) and 64 (dashed lines) bit variables. In the CPU tests a number of parallel threads equals to the number of CPU cores is always used, since this is the best performance condition and OpenCL does not allow to change easily this setting.

In a first benchmark, the 4 different set-up are tested for performances in a typical dynamical aperture computation. The test is repeated using 32- and 64-bit variables. As shown in Fig. 4, using 32-bit variables results in a remarkable performance improvement for the GPUs, while no difference is observed for the CPUs.

Furthermore, it is evident how to achieve good performance it is crucial to feed the GPU with enough parallel computation to keep all the core saturated, the experiment shows that the best condition is met when the number of parallel tracked particle is a few times the number of GPU cores. The cause is attributed to the inability to keep the cores saturated when running only one program at a time, in fact certain operations can require several instruction cycles and therefore can be convenient to interleave a few simulations, that is achieved when simulating a number of particle multiple of the number of cores. Testing has shown that for best performances, the number of simulated electrons should be chosen between 3 and 5 times the number of cores depending on the specific hardware.

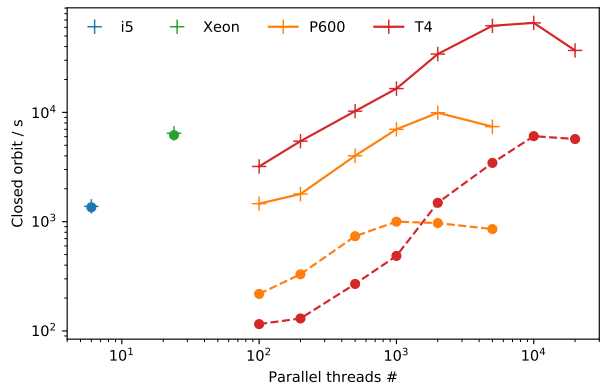

Figure 5: Closed orbit benchmark. Different number of particles are tracked in parallel on the 4 different hardware configurations. The test is repeated using 32- (solid lines) and 64- (dashed lines) bit variables. In the CPU tests a number of parallel threads equals to the number of CPU cores is always used, since this is the best performance condition and OpenCL does not allow to change easily this setting.

The results of a second benchmark to test the performances for closed orbit computations are shown in Fig. 5. Also in this case results similar to the dynamic aperture benchmark are observed.

CONCLUSION

UFO is a tracking code developed from scratch with electron ring optimization in mind. The here presented benchmarks show how a considerable performance boost can be achieved by introducing some physics and numerical approximation, which however are considered acceptable especially in the context of the initial design phase of an electron ring. Furthermore it was shown that by running UFO on a single high-end GPU it is possible to achieve a level of performances equivalent to the one provided by a small/medium class computer cluster. UFO is under active development: higher order integrator and full 6D simulations are currently under test.

ACKNOWLEDGEMENTS

The authors would like to thank Pau Carnicer Heras and Jordi Salabert Quintana from the ALBA computing section for setting up the GPUs used in this test, Zeus Martí and Gabriele Benedetti from the ALBA beam dynamics group for the fruitful discussions and for supporting the idea.

ISBN: 978-3-95450-227-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2022-WEPOMS043

REFERENCES

[1] L. Yang, Y. Li, W. Guo, and S. Krinsky, "Multiobjective optimization of dynamic aperture," *Phys. Rev. ST Accel. Beams* , vol. 14, p. 054001, 2011. doi:10.1103/PhysRevSTAB.14.054001

[2] M. P. Ehrlichman, "Genetic algorithm for chromaticity correction in diffraction limited storage rings," *Phys. Rev. Accel. Beams*, vol. 19, no.4, p. 044001, 2016. doi:10.1103/PhysRevAccelBeams.19.044001

[3] R. De Maria *et al.*, "SixTrack project: Status, runtime environment, and new developments," in *Proc. ICAP'18*, Key West, FL, USA, May 2018, pp. 172-178. doi:10.18429/JACoW-ICAP2018-TUPAF02

[4] J. King, I. Pogorelov, K. Amyx, M. Borland, and R. Soliday, "Current Status of the GPU-accelerated ELEGANT," in *Proc. IPAC'15*, Richmond, VA, USA, May 2015, pp. 623-625. doi:10.18429/JACoW-IPAC2015-MOPMA035

[5] UFO github: https://github.com/mcarla/ufo

[6] G. Benedetti *et al.*, "A distributed sextupoles lattice for the ALBA low emittance upgrade" in *Proc. IPAC'21*, Campinas, SP, Brazil, May 2021, pp. 2762–2765. doi:10.18429/JACoW-IPAC2021-WEPAB074

[7] Khronos Group - OpenCL: https://www.khronos.org/opencl/