UFO, a GPU Code Tailored Toward MBA Lattice Optimization

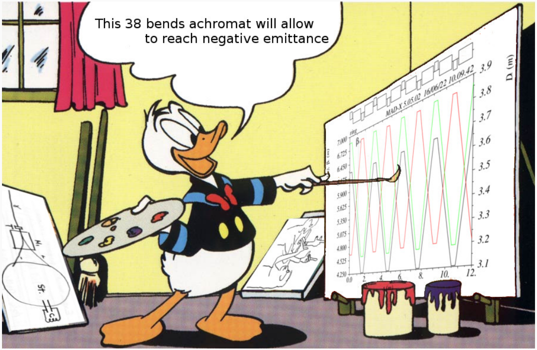
M.Carlà, M.Canals

ALBA-CELLS Synchrotron, 08290 Cerdanyola del Vallès, Spain

Jun 2022

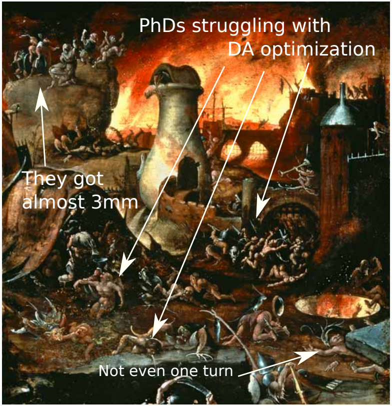
The lattice design and optimization process

Starts with a brilliant idea...



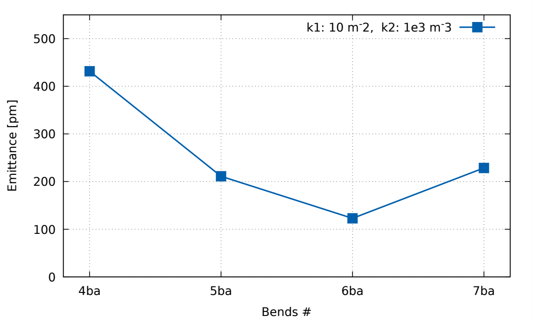
The lattice design and optimization process ...part 2

Ends with brute force optimization



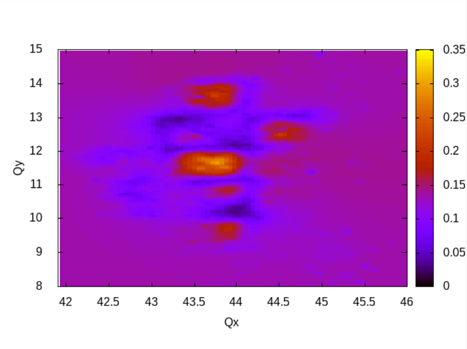
- A complicated lattice has **many parameters**
- **High dimensionality optimization problems are not human friendly!**
- The **optimization** phase can be a very **long, intensive and frustrating process...**

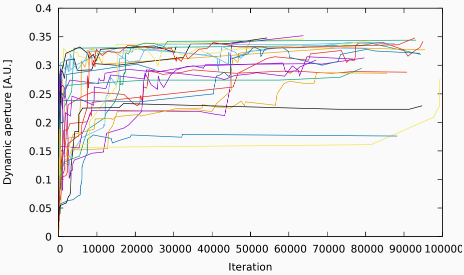
Optics optimization process



1. **Optimize arc** targeting: ϵ_x, α_c, $\psi_{x,y}$
2. Add **matching triplet** and mutate randomly the arc until closed solutions are found
3. **Optimize the entire ring** targeting: ϵ_x, **DA, lifetime,** α_c

Lattice optimization for ALBAII





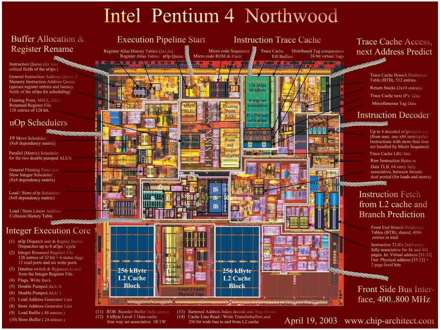
- A random walk is used to optimize the optics parameters: magnets length, strenght...
- The optimization metrics is function of D.A., β-functions at ID...
- **How to compute $\sim 10^6$ solutions in ~ 1 day?**

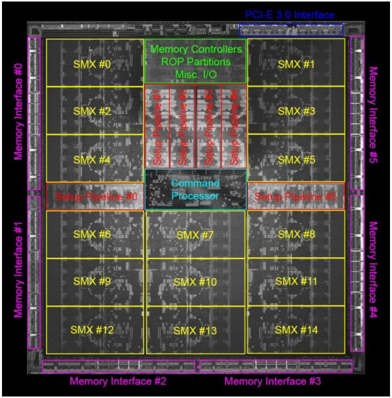
Introducing UFO

- A **few % error** in the computation of D.A. can be **tollerated**
- **GPU** fits very well single particle tracking
- Other GPU tracking codes already exists [1], but not optimized for electron ring
- UFO is not only pure **tracking** it also does **closed orbit** and some **linear optics**
- UFO is written in with a mixture of Python and OpenCL
- **UFO can run on CPU and GPU**

[1] https://github.com/SixTrack/sixtracklib

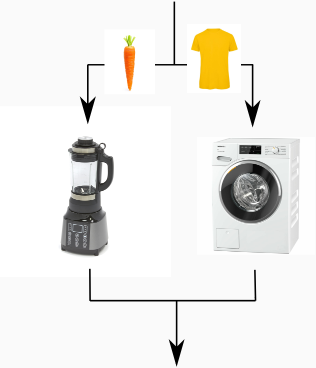
CPU & GPU



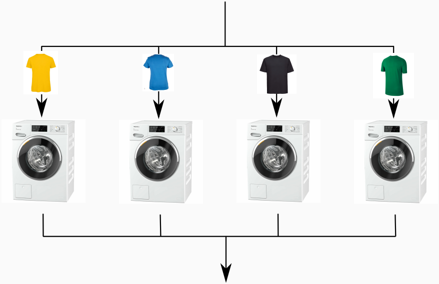


- CPU: A lot of effort into optimizing program flow execution (Branch-prediction, Out of order execution...)
- **CPU: Lots of space dedicated to stuff we don't use for tracking!**
- **GPU: Lots of small arithmetic cores** driven in parallel by a single instruction fetch/decode unit
- GPU: no branch-prediction/out of order execution. $\leftarrow$ we don't need this for tracking!!
- **GPU programs must fulfill very strict criteria to perform well!** $\leftarrow$ programming GPUs is an ...interesting experience

CPU vs GPU



- In a CPU cores are ∼independent
- Each core can run **his own program**
- Each core can operate over **different data**
- **Tens of cores at maximum**

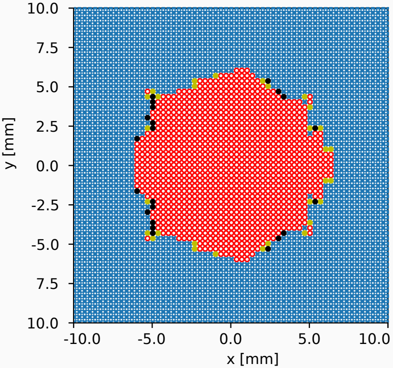


- In a GPU the **same instructions feed a group of cores**
- However, each core operate on **different data**
- A GPU can have a few to 100 groups of 64 to 128 cores
- **Thousands of cores**

Tracking through a ring fits well GPUs architecture: many particles move throught the exact same succesion of elements

UFO is not relativisticaly correct

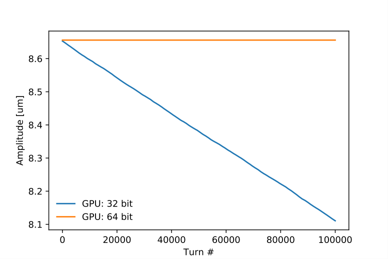
- Relativistic effects couple in a non-linear way the transverse motion
- Relativistic integrators (such as the one used in PTC) are slow!



- Electrons stable or unstable with both integrators are shown as red or blue markers
- Yellow and black electrons are stable for one integrator but unstable for the other
- Difference is a few %

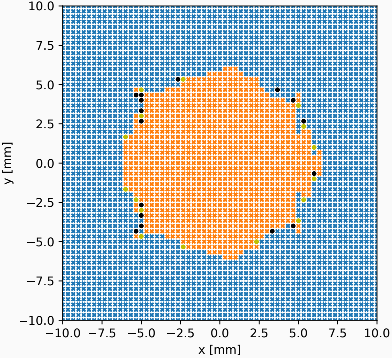
UFO use limited precision computations

- Scientific codes tend to use double precision floating point variables (64 bit)
- 64 bit variables are 2 times bigger than 32 and a few times slower



- In a long term tracking the loss of symplecticity is clearly visible for a 32 bit integrator
- 32 bit in GPU $\neq$ 32 bit in CPU: GPU does not follow ieee 754
- GPU are usually less precise...

Symplecticity loss with 32 bit variables



- Electrons stable or unstable with both integrators are shown as red or blue markers
- Yellow and black electrons are stable for one integrator but unstable for the other
- Difference is a few %

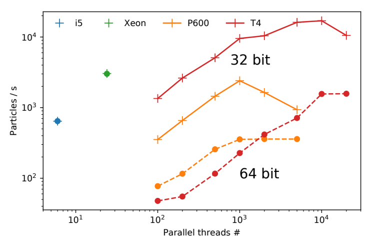
Benchmarking some hardware

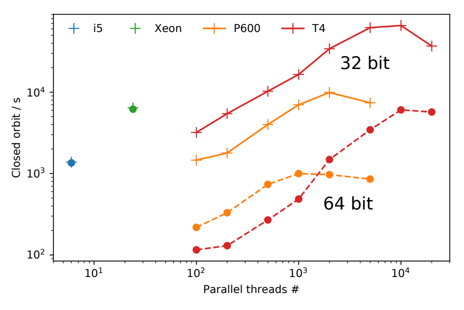
- To test performances 4 different machine were selected
- Some high and low-end GPUs and CPUs were selected

	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

- The test aims to find the optimal number of parallel threads for GPUs
- On a CPU the best performances is met when # of threads = # of cores
- OpenCL does not allow to change easily the number of thread on CPU (don't know why)

Benchmarking: Were is the gain?





- Tracking and closed orbit computation shows very similar behaviour
- To achieve good performances on GPU enough threads must be run in parallel (a few times the number of cores)
- **Performances gain respect to a CPU is ∼one order of magnitude**

Conclusions and outlook

UFO is a tracking code developed from scratch with electron ring optimization in mind:

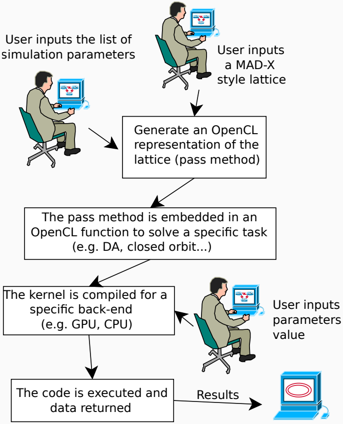
- Some physics and numerical approximation improve dramatically tracking performances
- Those approximations are acceptable in the initial design phase of an electron ring
- A single high-end GPU can achieve same performances of a small/medium class cluster

UFO is under active development:

- Fast linear optics matching
- Higher order integrators
- Full 6D simulations with RF and radiation

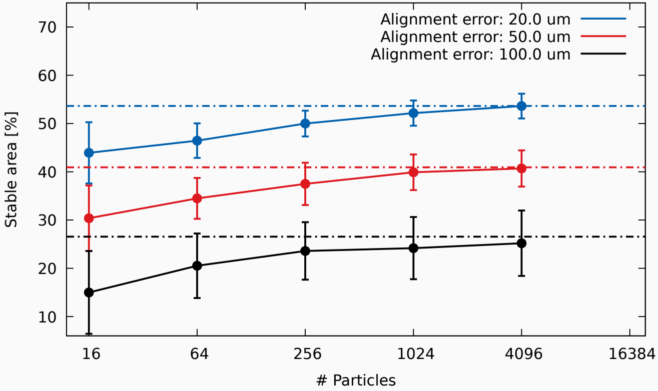
UFO has an unconventional interface...

- To achieve good performance the GPU must be under constant load
- Tracking 1000 particles is not enough to saturate a GPU



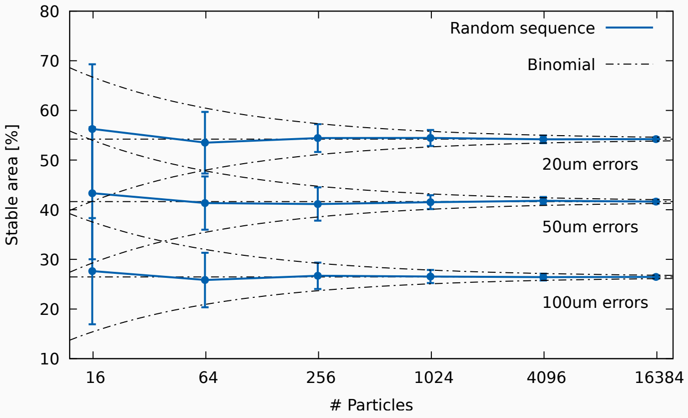
- Different optics variations can be simulated in parallel
- Optics parameters (field strength, element lengths...) can be specified per-particle
- Element order must be kept the same

Dynamic aperture on a grid



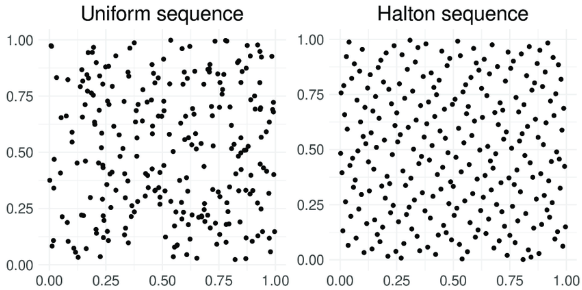
- A grid of N particles is tracked through the ring with a given set of errors (1000 turns)
- The error are chenged and the tracking repeated 100 times
- Points are the average + std of the 100 DA evaluations
- Strong systematic error for small number of particles

Dynamic aperture with random sampling



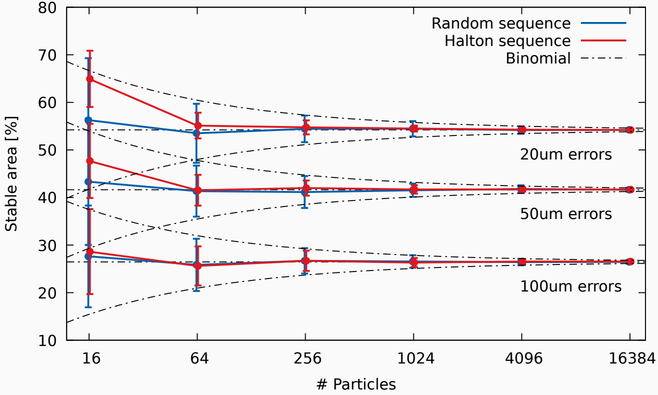
- Particles coordinates are generated randomly (uniform distribution)
- Each particle has a different set of errors (each particle → different machine)
- No systematic errors, but the error bars are bigger (noise in particles initial conditions)
- 1024 particles, 100um errors → Average DA error: 1.32%

Dynamic aperture with halton sampling



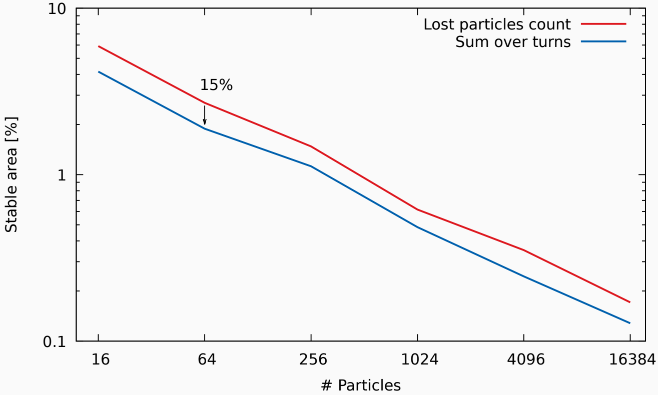
- Random initial conditions increase the overall estimator noise
- A Halton sequence is uniformly distributed and kind of random ...but not too random

Halton sampling (Red curve)



- Limited systematic error and much smaller random fluctuations
- 1024 particles, 100um errors $\rightarrow$ Average DA error: 0.94%

Counting turns instead of lost particles



- Up to now DA was evaluated by looking at the number of survived particles
- Similar information is obtained by summing the total amount of turns
- A 15% reduction of the DA evaluation error is obtained
- 1024 particles, 100um errors → Average DA error: 0.80%