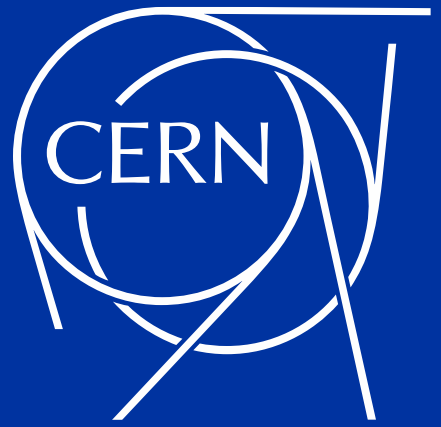


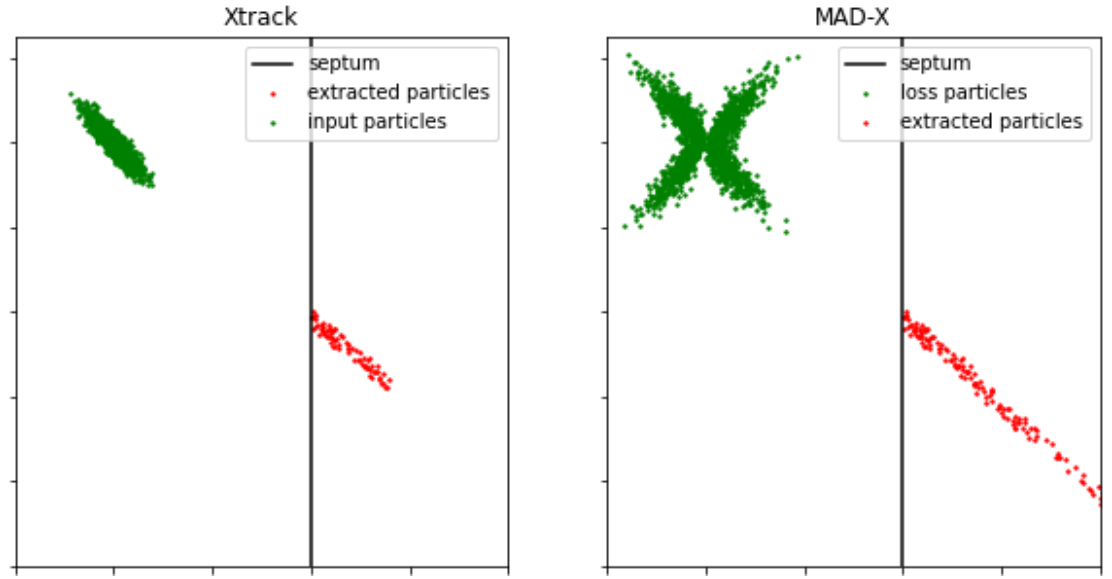
MAD-X and Xtrack Benchmarking on CERN SWAN

Thomas Bass
CERN SY-ABT-BTP (Royal Holloway Univ. London)
2022-10-27

iFAST 5.3

Benchmarking Background

- ***SWAN* – CERN's Jupyter Notebook service – provides CUDA GPU Acceleration**
- **Xsuite's *Xtrack* module can take advantage of this acceleration**
  - *context = xo.ContextCupy()*
- **Tests run using *SWAN's* new Kubernetes infrastructure swan-k8s.cern.ch**
  - Software stack: 102 Cuda 11.2 (GPU)
  - Platform: CentOS 7 with gcc8
  - Each SWAN GPU session is assigned a Tesla T4
  - *Approximately equivalent to an RTX 2070/2080*
- **Benchmarking with SPS Slow Extraction**
  - MAD-X using cpymad
  - Xtrack using CPU and CUPY contexts

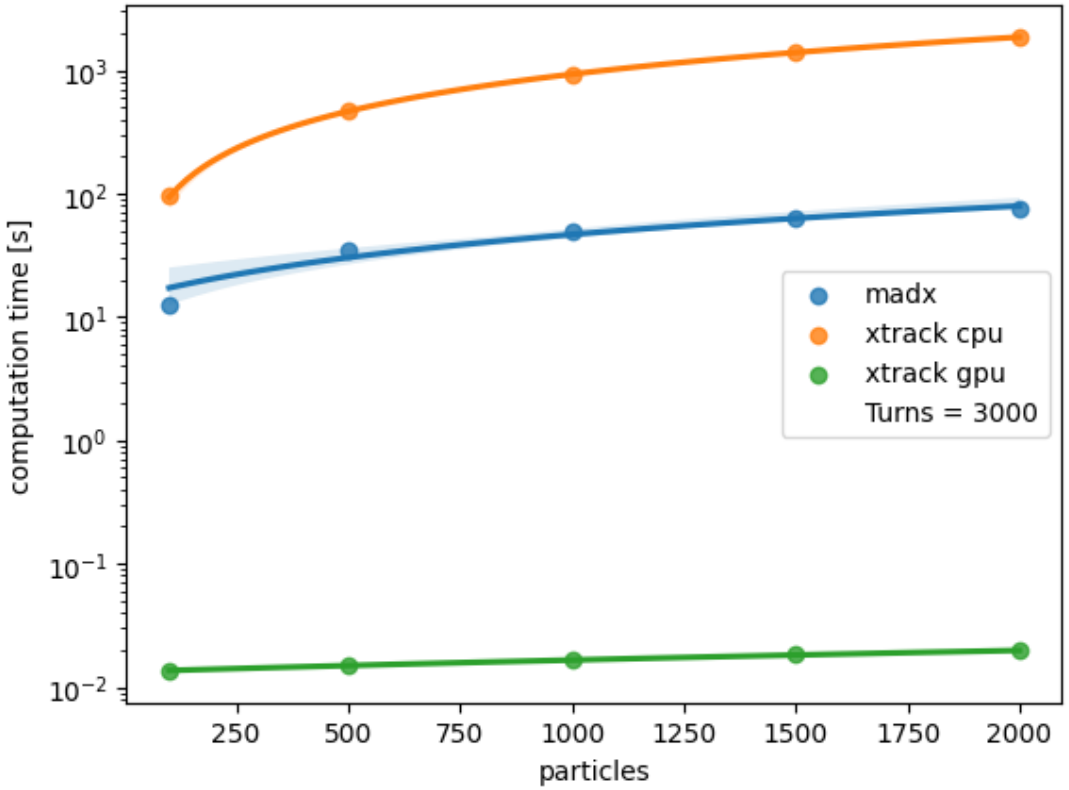


Simulation Details

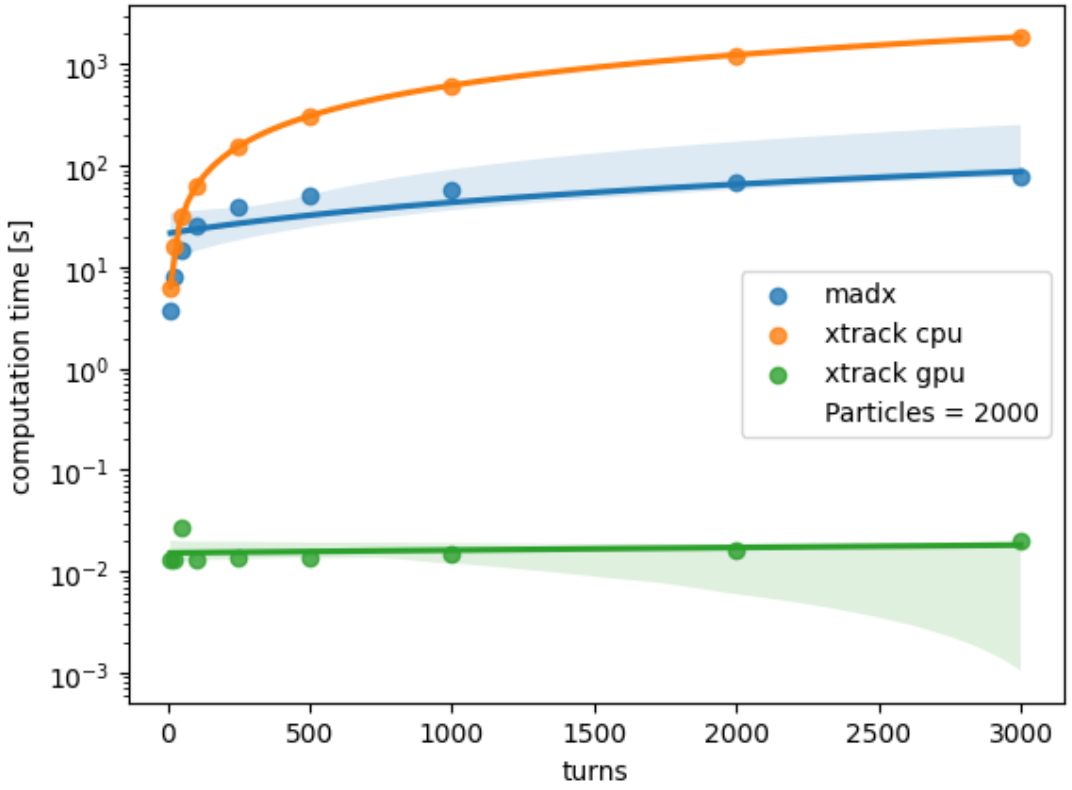
- **SPS machine slow extraction**
- **Using ft_q26_extr optics**
  - Fixed target beam
  - Integer tune of 26
  - Just before extraction
- **Using *Henontrack* from Pablo**
  - gitlab.cern.ch/parrutia/henontrack
- **Using Xsuite's *Xtrack* module**
  - github.com/xsuite

Preliminary benchmarking results

Varying number of particles



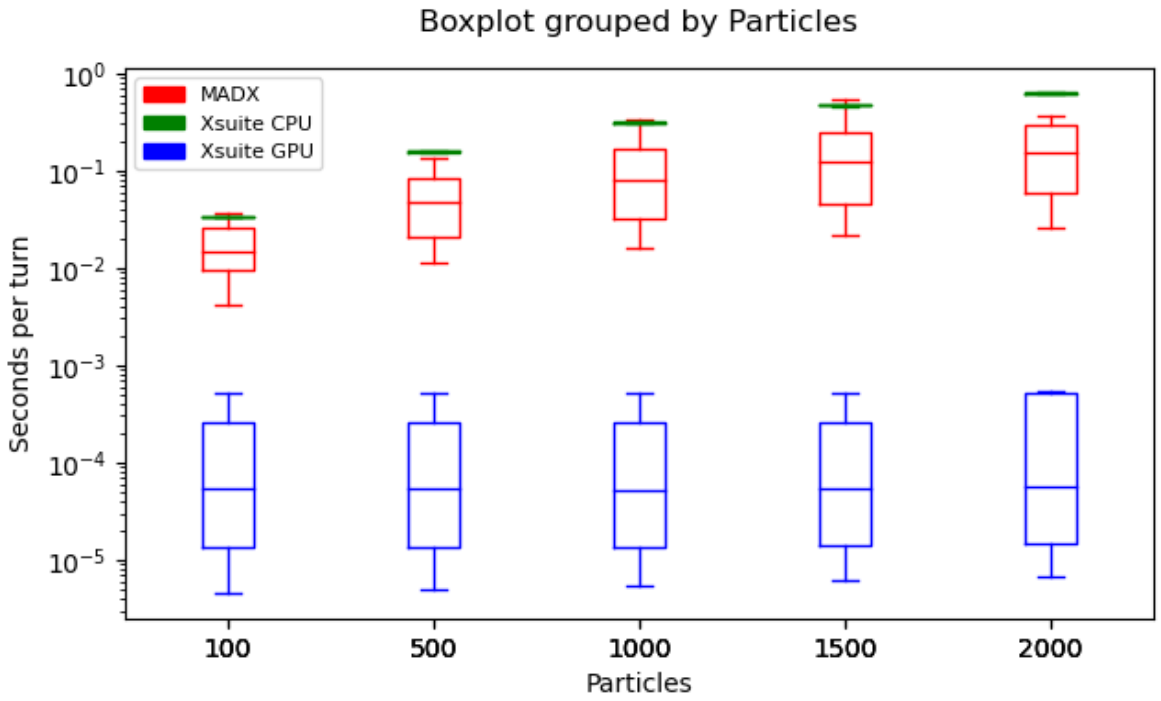
Varying number of turns



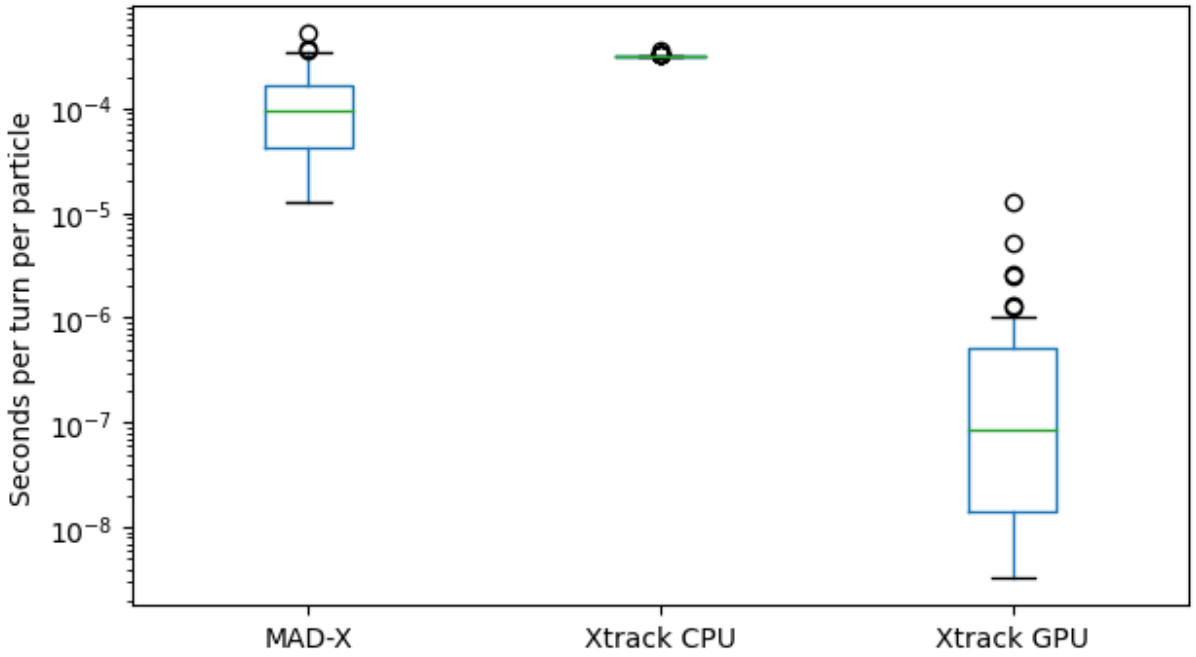
Graphs show that Xtrack GPU's computation times do not change considerably with increasing numbers of particles or turns up to 2,000 particles and 3,000 turns

Other perspectives

Seconds per turn at multiple particle counts



Seconds per turn per particle



Boxplot demonstrates that Xtrack is bounded by a constant-time process when using CPU up to 3,000 turns and 2,000 particles

Conclusions

- **Xtrack with GPU acceleration is clearly the fastest at 10^3 order of magnitude**
- **Xtrack+GPU overall simulation runtime appears to be almost constant**
- **Xtrack+CPU is *slower* than MAD-X**
  - So in CPU-limited environments, MAD-X may be faster
- **Xtrack+CPU appears to have some constant limiting factor for each particle/turn**
  - Seconds per turn per particle appears to be constant
- ***However – results can only be considered in relation to each other, not as objective values***
  - *During the course of each simulation, particles are lost and not tracked*
  - *This is not currently considered when calculating time per particle / turn*

SWAN's Limitations

- **Limited to 16GB RAM**
- **Xtrack's limits still beyond SWAN's limitations**
- **Simulations with nparticles >50,000 causes memory allocation issues**

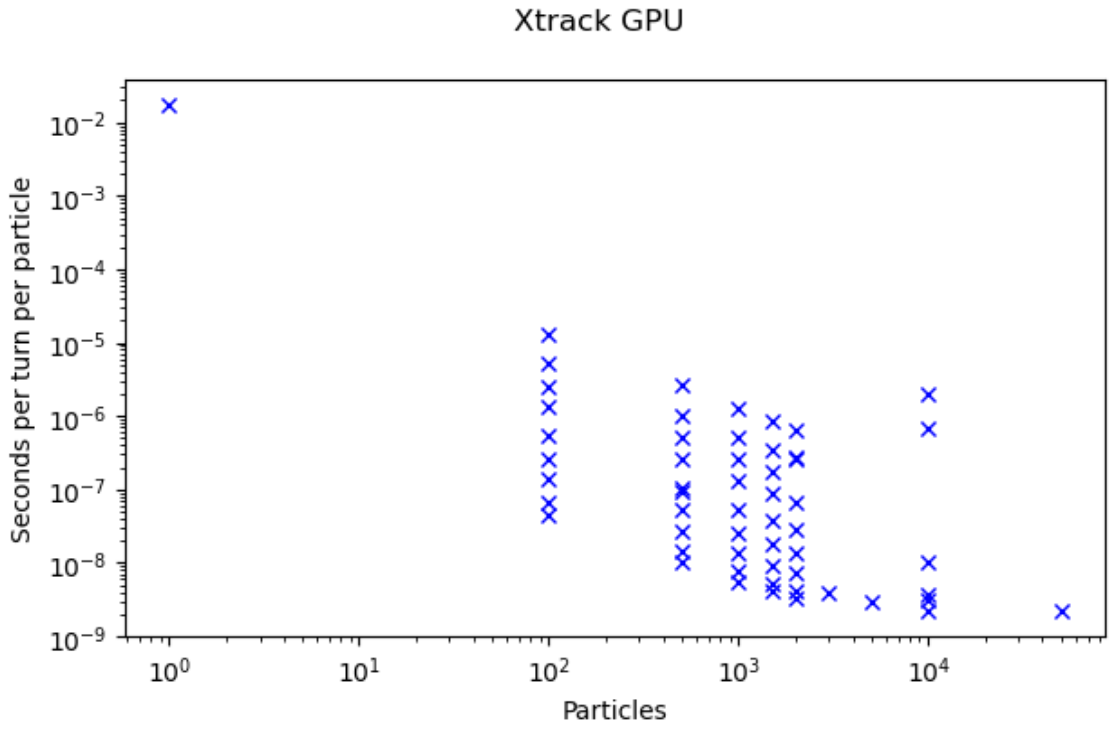
```
cupy/cuda/memory.pyx in cupy.cuda.memory.SingleDeviceMemoryPool.malloc
()

cupy/cuda/memory.pyx in cupy.cuda.memory.SingleDeviceMemoryPool._malloc
()

cupy/cuda/memory.pyx in cupy.cuda.memory.SingleDeviceMemoryPool._try_ma
lloc()

OutOfMemoryError: Out of memory allocating 18,400,001,024 bytes (alloca
ted so far: 51,291,648 bytes).
```

- **Close to 50,000 approaches the time/turn/particle limit**



Next Steps

- **Record the number of particles tracked during each turn for better analysis**
- ***SWAN* is a simple and quick method of GPU acceleration**
- ***HTCondor* provides a more robust platform, and supports GPUs**
  - Explore *HTCondor* to repeat and extend preliminary benchmarking
  - Create profiling tools to identify potential optimisations
  - Develop packages to automate *HTCondor* batch submission for large simulation tasks
- **Does *cpymad* impact performance of MAD-X?**
- **How does twiss behave? Slicing?**

home.cern