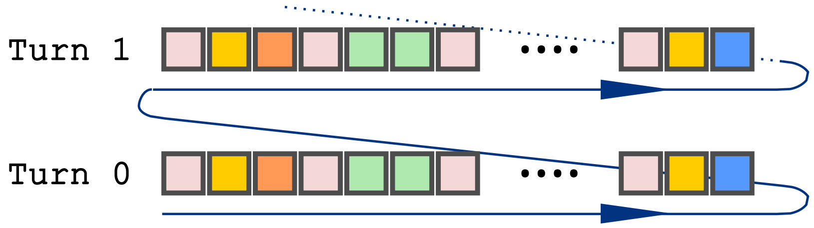
SixTrackLib: A Library for GPU Accelerated Single-Particle Tracking

Martin Schwinzerl, Riccardo De Maria, Giovanni Iadarola,
Hannes Bartosik, Konstantinos Paraschou (CERN)
Adrian Oeftiger, Vera Chetvertkova (GSI)

ABP Group Information Meeting February 2020 :: 2020/02/27

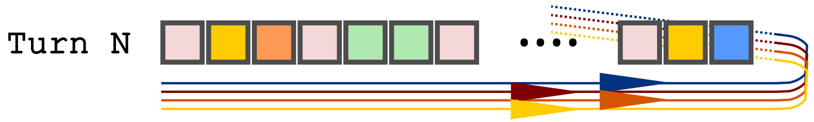
Motivation

- Tracking one particle over lattice ➡ sequential operation



- Tracking over $N_T \geq 10^4 \dots 10^8$ turns ➡ numerically expensive

- Tracking $N_P \gg 1$ non-interacting particles over lattice



➡ "embarrissingly parallel problem"

- Idea: Reimplement Core of SixTrack as a stand-alone library, suitable for (massively) parallel systems ➡ SixTrackLib

SixTrackLib: Features

- SixTrackLib: https://github.com/SixTrack/sixtracklib
- OpenCL 1.2, CUDA, SIMD Auto-Vectorisation / C, C++, Python3

- Particles: 6D Phase-Space

  $$\{x, y, p_x, p_y, \zeta, \delta\}$$

  Common Implementation (C99, header-only) for physics models accross all architectures, languages, ...

- Beam-Elements
  - Drift, DriftExact,
  - Multipole, DipoleEdge
  - Cavity, RFMultipole,
  - Limit (Rect, Ellipse, Rect+Ellipse)
  - XYShift, SRotation,
  - Beam-Fields: Coasting & Bunched Frozen Space-Charge, 6D/4D BeamBeam,..

- Related Python-Centric Projects
  - cobjects: https://github.com/SixTrack/cobjects
  - pysixtrack: https://github.com/SixTrack/pysixtrack
  - sixtracktools: https://github.com/SixTrack/sixtracktools

Example: Create Lattice, Tracking Using The CPU

In [1]:

```python
import sixtracklib as st

# Build a Fodo Lattice
lattice = st.Elements()
quad_f  = lattice.Multipole(knl=[0.0, 0.165])
drift01 = lattice.Drift(length=10.0)
quad_d  = lattice.Multipole(knl=[0.0, -0.165])
drift02 = lattice.Drift(length=10.0)

# Create a beam with a set of particles for tracking
beam = st.ParticlesSet()
particles = beam.Particles(num_particles=10, p0c=4.5e11)

# Save lattice and particle states to files for later re-use
lattice.to_file('./example_lattice.bin')
beam.to_file('./example_particles.bin')

# Create a CPU SixTrackLib TrackJob
job = st.TrackJob(lattice, beam)

status = job.track_until( 1 )

job.collect_particles()
print( f"particles at_turn: {particles.at_turn}")
```

```
particles at_turn: [1 1 1 1 1 1 1 1 1 1]
```

Example: Import Lattice From MAD-X

In [2]:

```python
# Using cpymad and pysixtrack
from cpymad.madx import Madx
import numpy as np
import pysixtrack as pyst

m_p = 938e6 # [m_p] = 1 eV
c0 = 299792458.0 # [c0] = 1 m/s

beam = st.ParticlesSet()
p0c = 450e9 # [p0c] = 1 eV
Etot = np.sqrt( p0c * p0c + m_p * m_p * c0 ** 4 ) * 1e-9 # [Etot] = 1 GeV !!!
particles = beam.Particles(num_particles=10, p0c=4.5e11)

mad = Madx()
mad.call(file="fodo.madx")
mad.command.beam(particle="proton", energy=str(Etot))
mad.use(sequence="FODO")

fodo = mad.sequence.FODO

pyst_line = pyst.Line.from_madx_sequence(fodo, exact_drift=True)
pyst_line.remove_zero_length_drifts(inplace=True)

lattice = st.Elements()
lattice.append_line( pyst_line )

# .... Continue like above ...
```

```
++++++++++++++++++++++++++++++++++++++++++++
+     MAD-X 5.05.01  (64 bit, Linux)       +
+ Support: mad@cern.ch, http://cern.ch/mad +
+ Release   date: 2019.06.07               +
+ Execution date: 2020.02.27 01:25:50      +
++++++++++++++++++++++++++++++++++++++++++++
```

Example: OpenCL Tracking (GPU or CPU)

In [3]:

```
# Same lattice and beam definition, but now track on an OpenCL device
# First: find out which devices are available
!clinfo -l
```

```
Platform #0: Portable Computing Language
 `-- Device #0: pthread-Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
Platform #1: Intel(R) OpenCL
 +-- Device #0: Intel(R) HD Graphics
 `-- Device #1: Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
Platform #2: Experimental OpenCL 2.1 CPU Only Platform
 `-- Device #0: Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
```

In [4]:

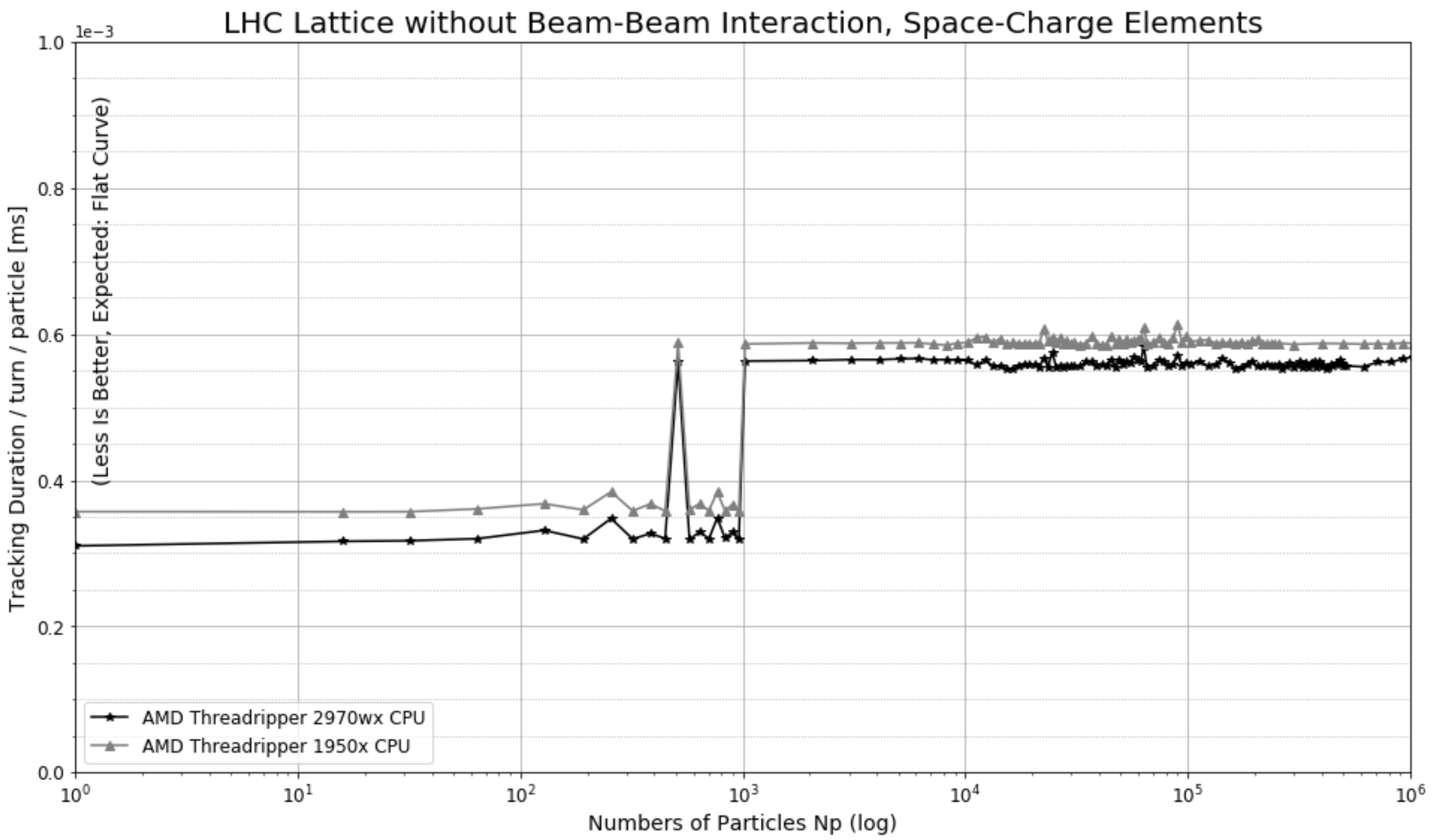
```
# Then create an OpenCL track-job using one of the devices
lattice = st.Elements.fromfile('./example_lattice.bin')
beam = st.ParticlesSet.fromfile( './example_particles.bin')
cl_job = st.TrackJob(lattice, beam, device="opencl:2.0")

# The rest is like before
status = cl_job.track_until( 2 )
cl_job.collect_particles()

particles = beam.cbuffer.get_object(0)
print( f"particles at_turn: {particles.at_turn}")
```

```
particles at_turn: [2 2 2 2 2 2 2 2 2 2]
```

Performance Analysis: LHC Lattice (no BB, no SC)

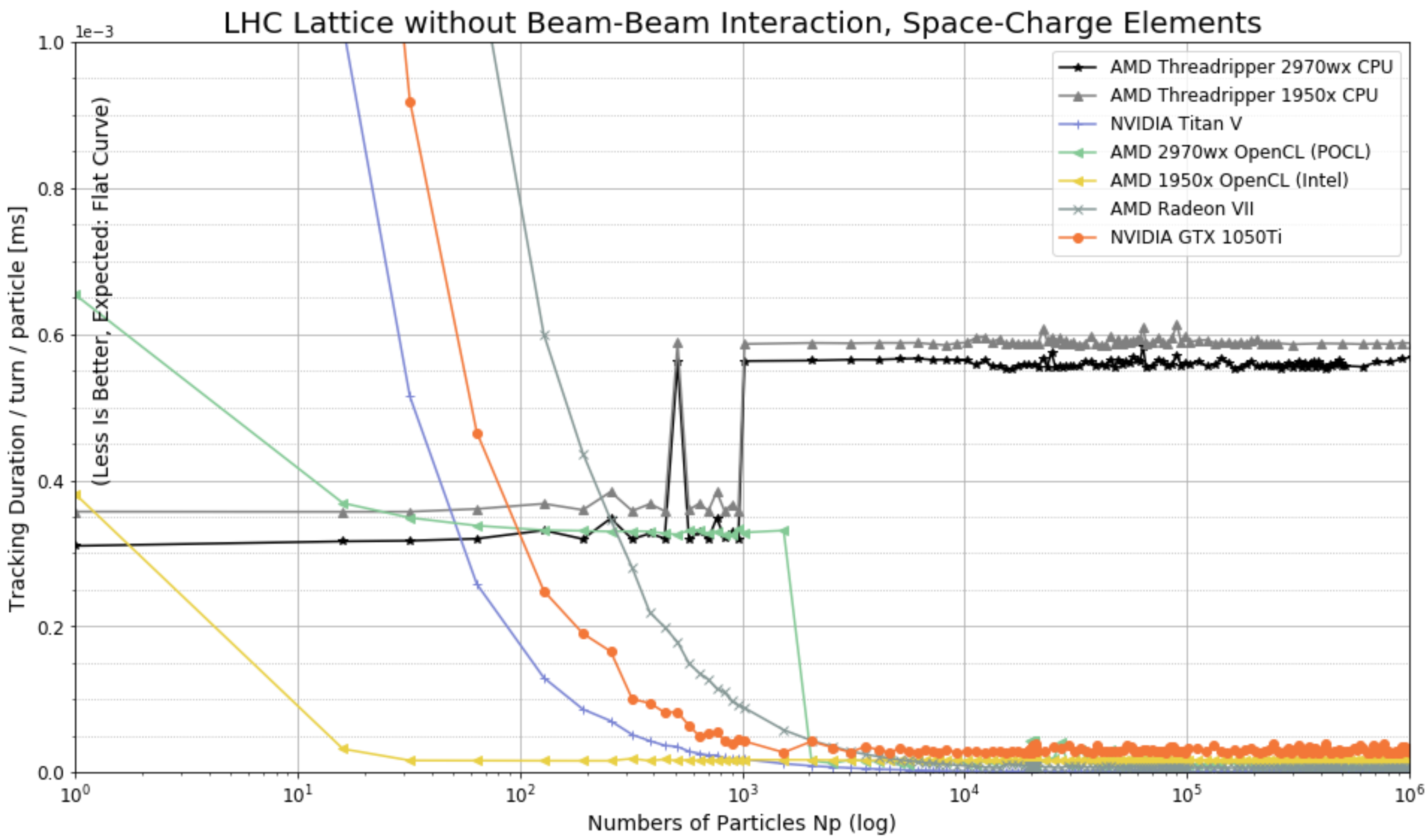
In [8]:

```
plt = plot_cpu_comparison(plt, files)
plt.rcParams["figure.figsize"]=(16,9)
```



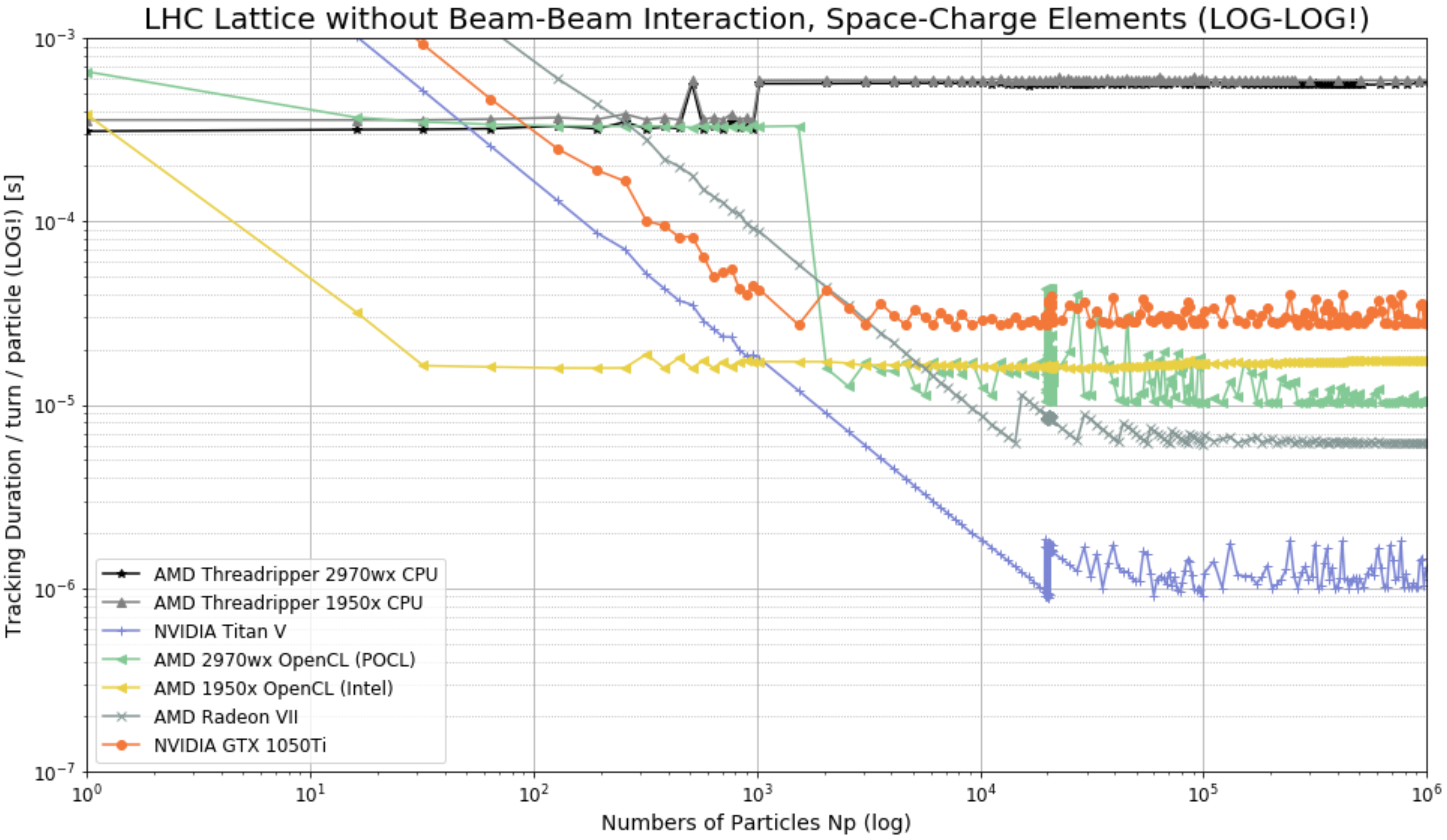
Performance Analysis: LHC Lattice (no BB, no SC)

In [10]:
```
plt = plot_cpu_vs_ocl_all_enabled(plt, files)
plt.rcParams["figure.figsize"]=(16,9)
```



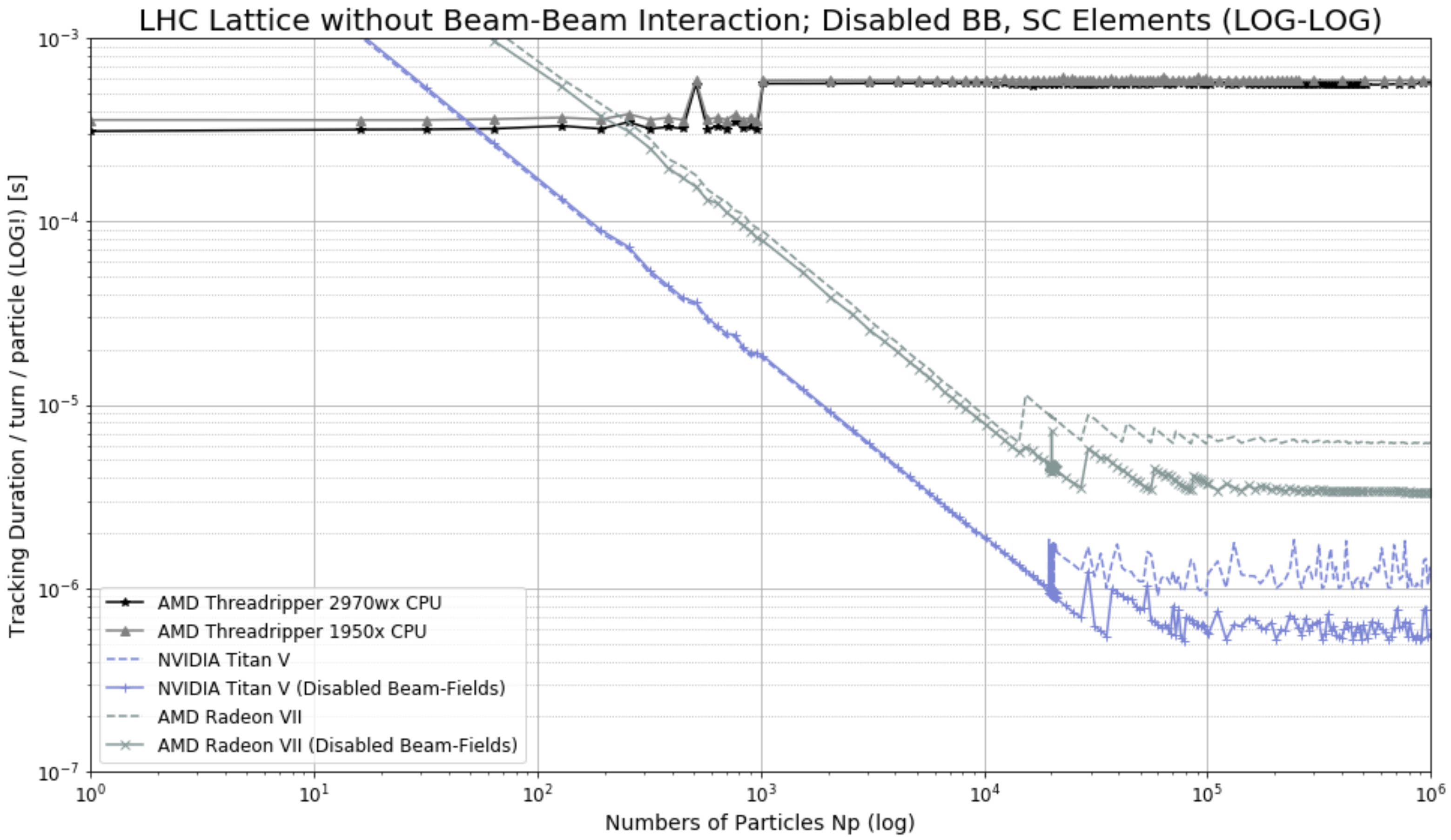
Performance Analysis: LHC Lattice (no BB, no SC)

In [12]:
```
plt = plot_cpu_vs_ocl_all_enabled_log_log(plt, files)
plt.rcParams["figure.figsize"]=(16,9)
```



Performance Analysis: Overhead Of Beam-Field Elem.

In [14]:
```
plt = plot_cpu_vs_ocl_all_none_enabled_log_log(plt, files)
plt.rcParams["figure.figsize"]=(16,9)
```



Status, Conclusion & Outlook

- SixTrackLib: 2-3 Ord. of Mag. Speedup (Depending on HW, Np, Lattice,...)
- Currently used as a "building-block" in dedicated studies
- Focus on testing, physics benchmarking, optimization
- API and ABI might still change / break ➡ "Selected Experienced Users"
- Goals:
  - Use as an (optional) tracking backend for SixTrack
  - Prepare & Test use within the Framework of BOINC (LHC@Home)
  - C++ Bindings are fully templated ➡ MultiPrec, SIMD, evt. TPSA

Thank You For Your Attention!

https://www.github.com/SixTrack/sixtracklib

Related Studies and Präsentations (Selection)

- A. Oeftiger "GPU accelerated space charge simulations using SixTrackLib and PyHEADTAIL" (4th ICFA Mini-Workshop on Space Charge 2019, CERN) https://indico.cern.ch/event/828559/contributions/3528454/ (https://indico.cern.ch/event/828559/contributions/3528454/)
- H. Bartosik "Studies on tune ripple" (4th ICFA Mini-Workshop on Space Charge 2019, CERN) https://indico.cern.ch/event/828559/contributions/3528378/ (https://indico.cern.ch/event/828559/contributions/3528378/)
- K. Paraschou "Symplectic kicks from an electron cloud pinch" (ABP Group Info Meeting, January 2020, CERN) https://indico.cern.ch/event/880340/ (https://indico.cern.ch/event/880340/)