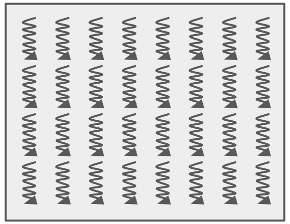
GPU-friendly surface model for Monte-Carlo detector simulations

Severin Diederichs, on behalf of the VecGeom Surface Model team
CERN, EP-SFT

severin.diederichs@cern.ch

GPUs are optimized for *throughput*

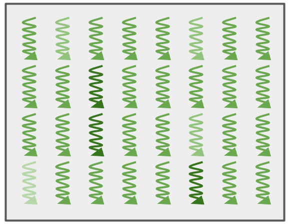
Code is executed in ***warps,*** which consist of ***32 threads***



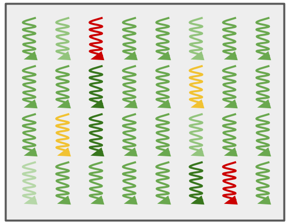
GPUs are optimized for *throughput*

Code is executed in ***warps,*** which consist of ***32 threads***

Divergence drastically reduces performance



fast



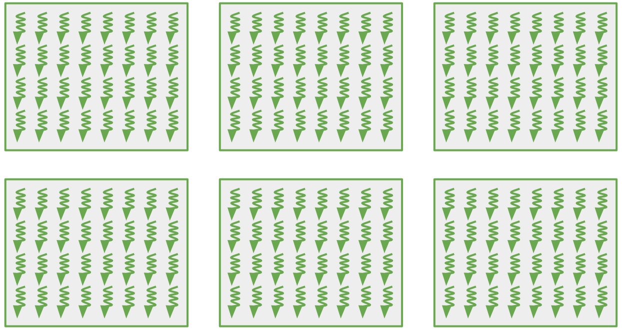
slow

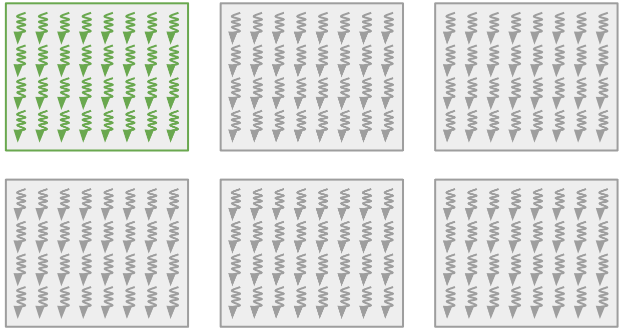
GPUs are optimized for *throughput*

Code is executed in ***warps,*** which consist of ***32 threads***

Divergence drastically reduces performance

Occupancy = # active warps / # of max possible warps
Theo. maximum limited by ***registers / thread!***


code with **32 registers / thread**


code with **255 registers / thread**

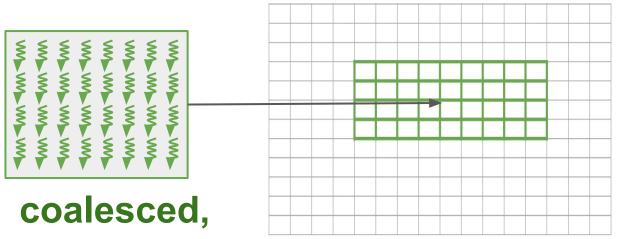
GPUs are optimized for *throughput*

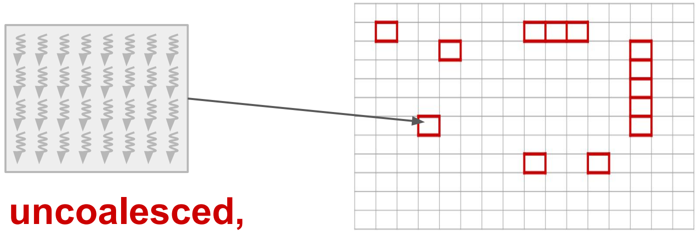
Code is executed in ***warps,*** which consist of ***32 threads***

Divergence drastically reduces performance

Occupancy = # active warps / # of max possible warps
Theo. maximum limited by ***registers / thread!***

Memory accesses are crucial

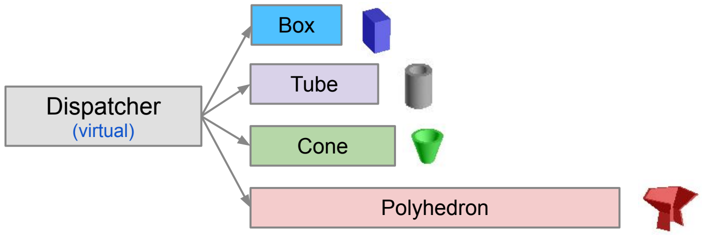

coalesced, fast


uncoalesced, slow

VecGeom solid model is a huge bottleneck on GPU

- recursive calls, virtual functions, complex algorithms ➞ high register and stack usage, low occupancy on GPU
- very different complexity per solid ➞ high divergence
- uncoalesced memory accesses ➞ high latency
- relies on small pushes for knowing in which volume one is ➞ requires double precision

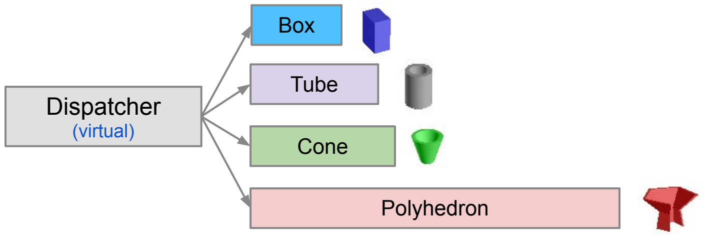
Significant divergence in the solid model



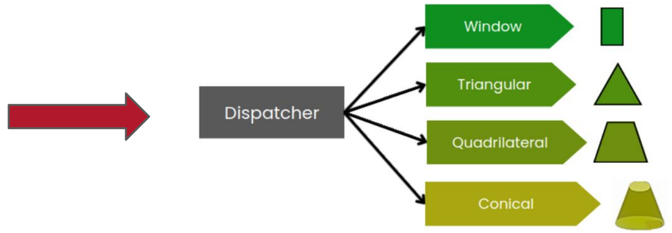
VecGeom surface model optimized for GPUs

- ~~recursive calls, virtual functions,~~ less complex algorithms → lower register and stack usage → higher occupancy on GPU?
- reduced complexity per surface → lower divergence?
- uncoalesced memory accesses → high latency (intrinsic to geometry)
- State is known by navigation, no pushes required, enables potential use of mixed precision

Significant divergence in the solid model

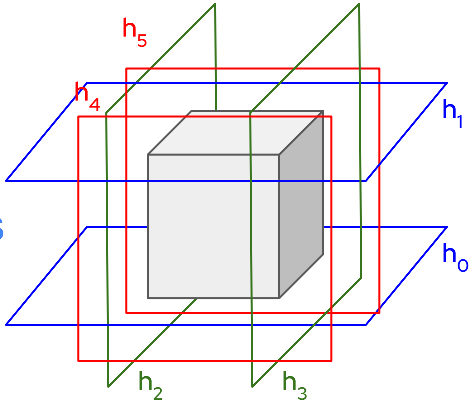


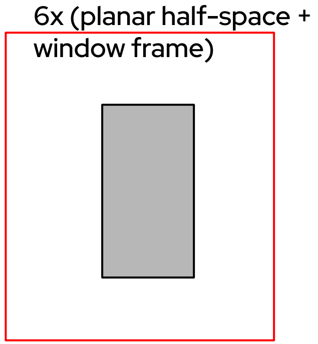
Reduced divergence using surfaces

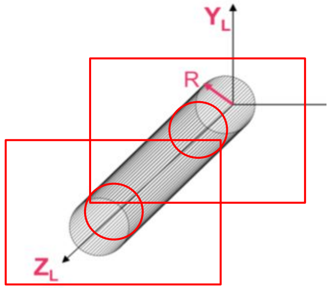


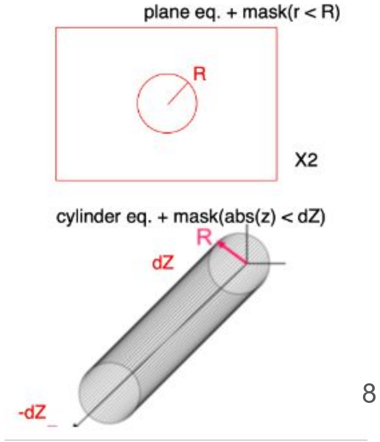
VecGeom uses a bounded surface approach

- 3D bodies represented as Boolean operation of half-space CommonSurfaces
  - First and second order, infinite
  - Just intersections for convex primitives e.g. box = h_0 & h_1 & h_2 & h_3 & h_4 & h_5
- Storing the solid imprint (frame) as a bounded FramedSurface







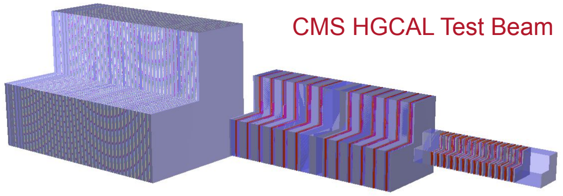


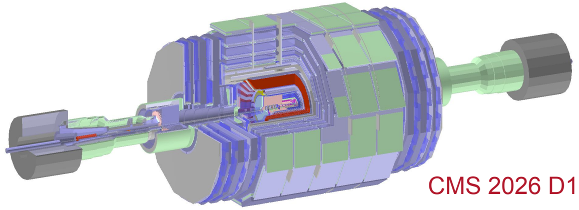
All solids supported

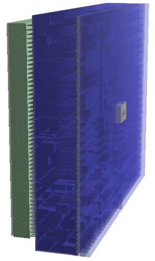
Conversion time and memory footprint under control

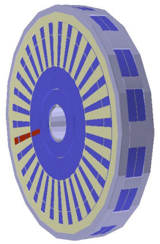
	# touchables [million]	conversion time [s]	memory [MB]
cms_2018	2.1	5.1	307
cms_TB_HGCAL	0.06	0.8	51.4
cms_2026D110	13.1	59.8	673
LHCb_Upgrade	18.5	92.8	173
LHCb_ECal_HCal	18.4	0.8	6.7
ATLAS_EMEC	0.08	1.4	132

Correctness tested with randomized raytracing


CMS HGCAL Test Beam

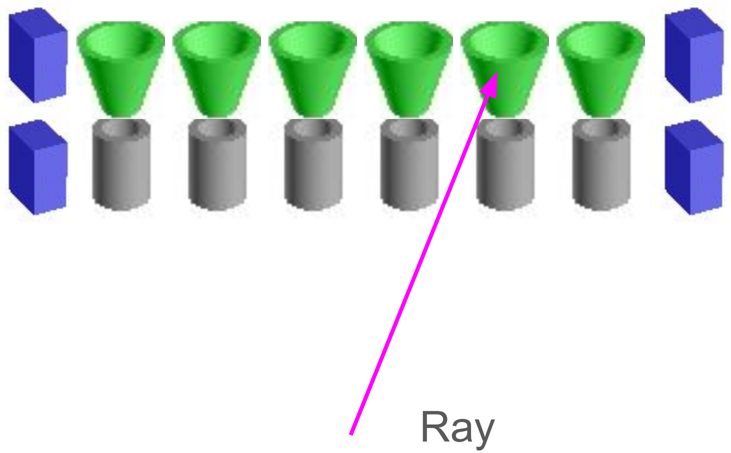

CMS 2026 D110


LHCb ECal HCal


ATLAS EMEC

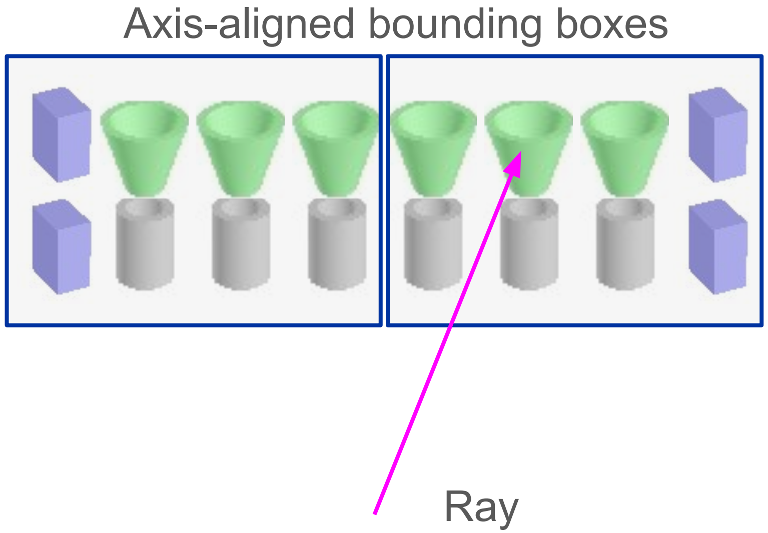
Entering surfaces can be a huge source of divergence

Naive approach: loop over all surfaces $\rightarrow$ huge divergence $\rightarrow$ **slow**



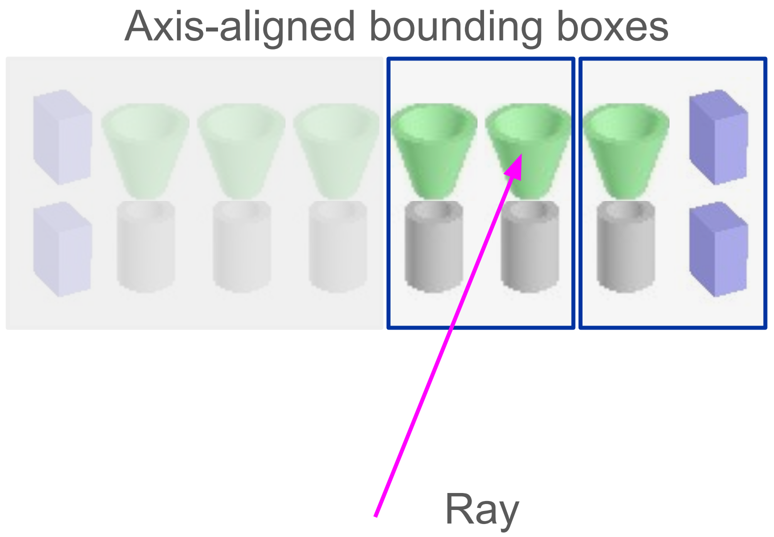
BVH acceleration structure for factors of speedup

- **Bounding Volume Hierarchies** (BVH) are used to speed up collision detection within 3D objects. Number of checks scale with **Log(n)**



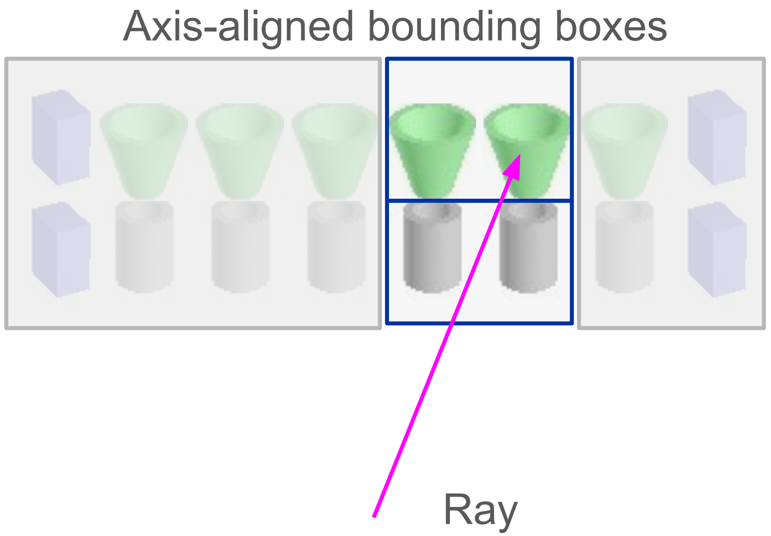
BVH acceleration structure for factors of speedup

- **Bounding Volume Hierarchies** (BVH) are used to speed up collision detection within 3D objects. Number of checks scale with **Log(n)**



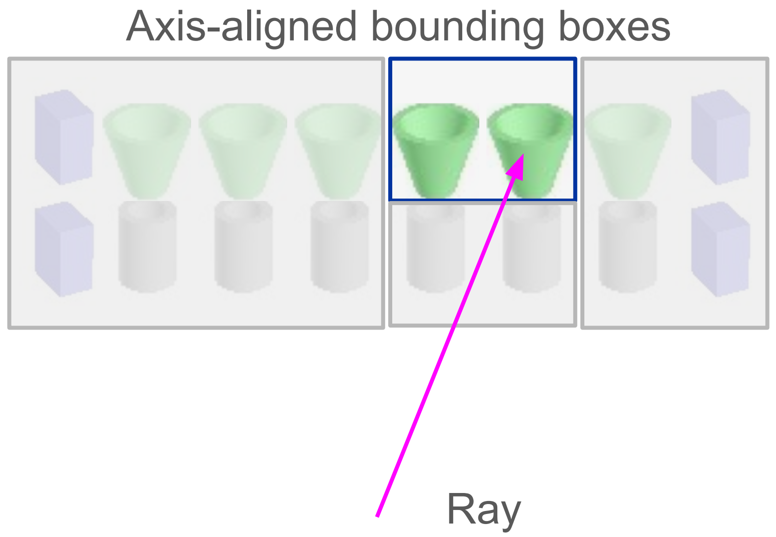
BVH acceleration structure for factors of speedup

- **Bounding Volume Hierarchies** (BVH) are used to speed up collision detection within 3D objects. Number of checks scale with **Log(n)**



BVH acceleration structure for factors of speedup

- **Bounding Volume Hierarchies** (BVH) are used to speed up collision detection within 3D objects. Number of checks scale with **Log(n)**



BVH acceleration structure for factors of speedup

- **Bounding Volume Hierarchies** (BVH) are used to speed up collision detection within 3D objects. Number of checks scale with **Log(n)**
- Solid model BVH adapted using the bounding boxes of surfaces

	HGCAL Test Beam	LHCb Calorimeters	CMS 2026 D110
Looper	1.097 s	3.007 s	26.78 s
With BVH	0.226 s	1.006 s	2.60 s
Speedup	**4.9x**	**3.0x**	**10.3x**

Run time of ray tracing with 10M rays

Surface model still slightly slower than solid model

AdePT
HGCAL Test beam: 100 primary electrons with 10 GeV

Solid model:	**2.62 s**
Surface model:	**2.77 s**

LHCb calorimeters: 8 ttbar events

Solid model:	**21.22 s**
Surface model:	**26.25 s**

Why slower after all the advertisement?

Surface model still slightly slower than solid model

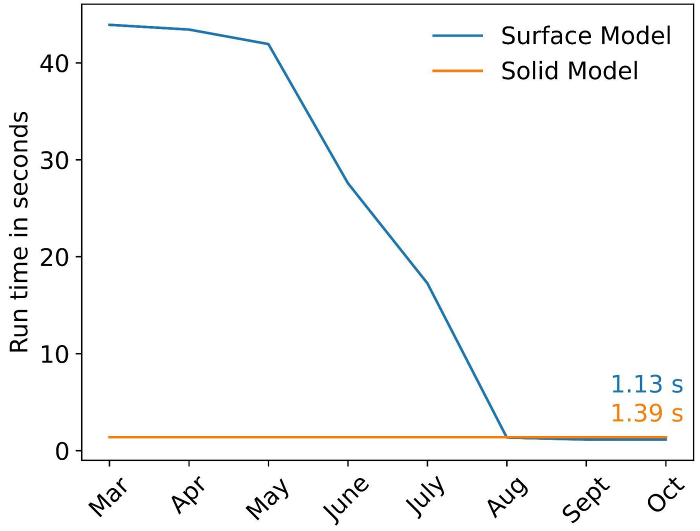
AdePT
HGCAL Test beam: 100 primary electrons with 10 GeV

Solid model:	**2.62 s**
Surface model:	**2.77 s**

LHCb calorimeters: 8 ttbar events

Solid model:	**21.22 s**
Surface model:	**26.25 s**

We've come a long way!
Raytracing in CMS TB
~ **40x** speed-up in 6 month



Surface model indeed uses less registers

	FindNextVolume	Relocation	Theo. Max. Occupancy
Solids	256	220	16%
Surfaces (double)	146	142	25%
Surfaces (mixed)	123	122	33%

… but single kernel in AdePT prevents this to have an effect

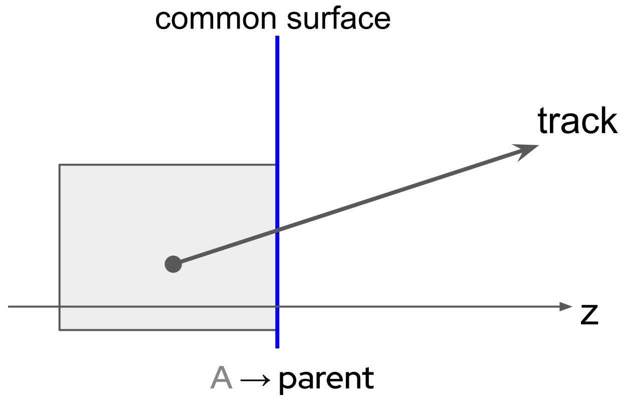
	FindNextVolume	Relocation	Theo. Max. Occupancy
Solids	256	220	16%
Surfaces (double)	146	142	25%
Surfaces (mixed)	123	122	33%

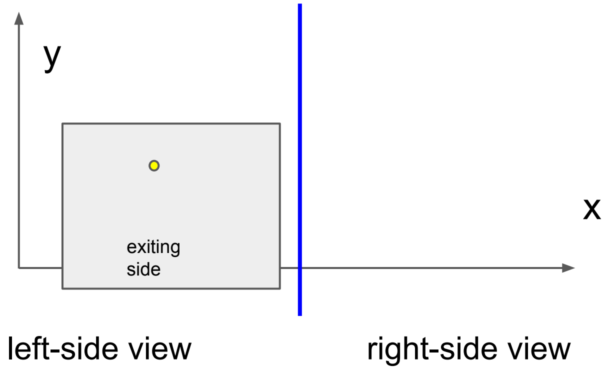
AdePT is using a *single kernel* which is still at the **maximum registers / thread**

→ Kernel must be split to benefit from lower registers (separate physics, geometry, magnetic field etc)

→ achieved occupancy must be limited by max. occupancy for this to have an effect

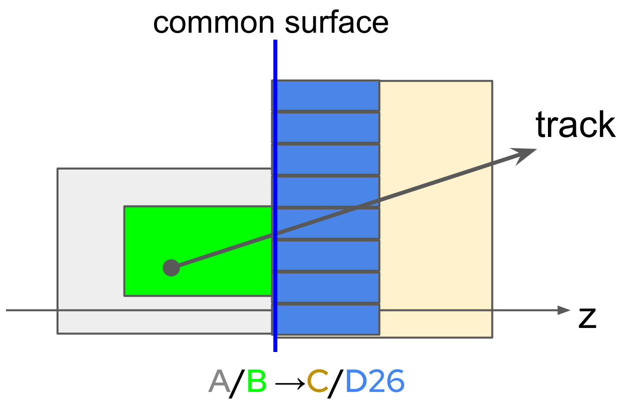
Relocation also source of divergence

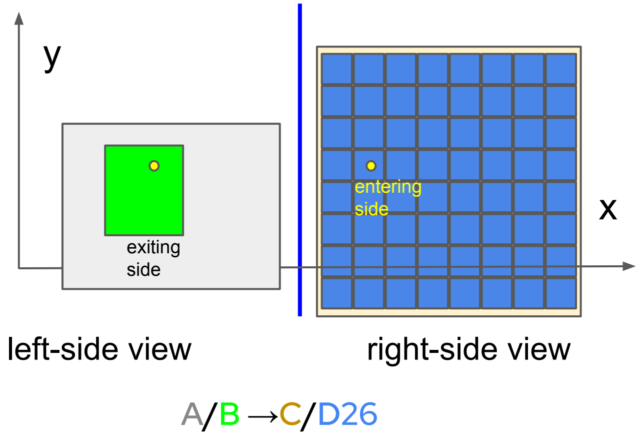




Exiting frames to check: 0
Entering frames to check: 0

Relocation also source of divergence

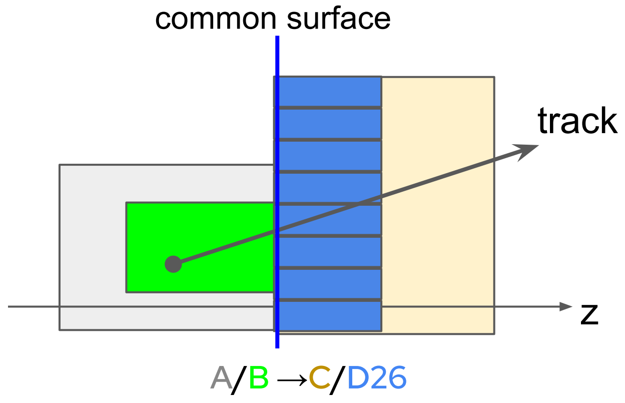


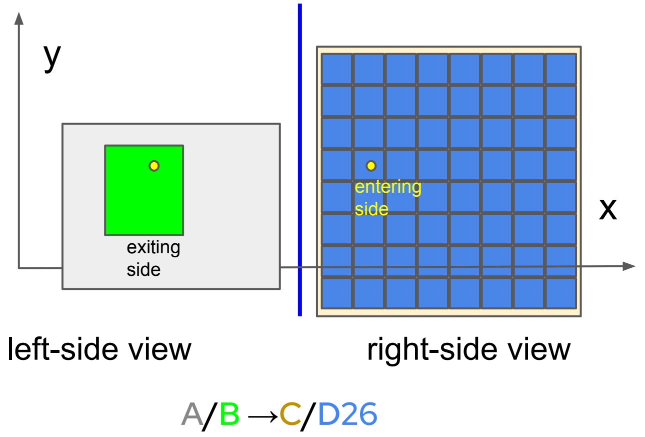


Exiting frames to check: 1
Entering frames to check: 64

Real life case CMS HGCAL TestBeam:
Entering frames to check: 800

Relocation also source of divergence





Exiting frames to check: 1
Entering frames to check: 64

Real life case CMS HGCAL TestBeam:
Entering frames to check: 800

After 2D grid optimization: ~ 8 frames to check
Further optimization ongoing

Memory access pattern need to be improved

Surface model seems to do worse than solid model:

- Memory access random but solid model seems to do more compute per access
- Surface model uses only a small fraction of the 32 bytes per memory transaction

Low level solutions:

- improve memory read per memory transaction (optimizing data structures),
- improve compute per memory read (recomputing over storing data)

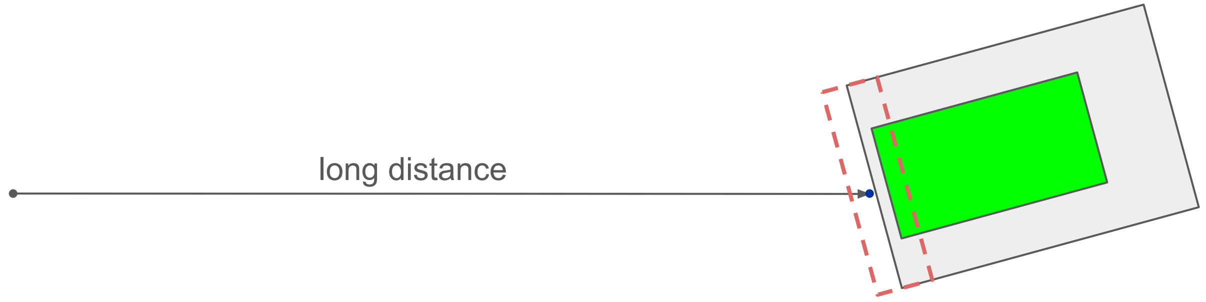
Mixed precision potentially enables significant speedup

GPUs are made for single precision:

- HPC GPUs: SP / DP flops **2:1**, consumer grade GPUs: SP / DP flops **32:1**
- lower register usage
- less memory to fetch

But: challenging due to ***different length scales***

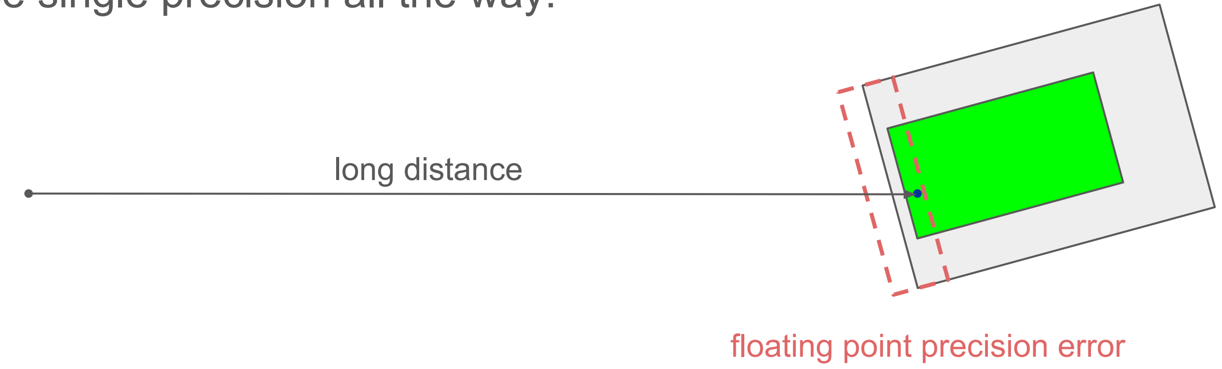
… but single precision must be used with great care



floating point precision error

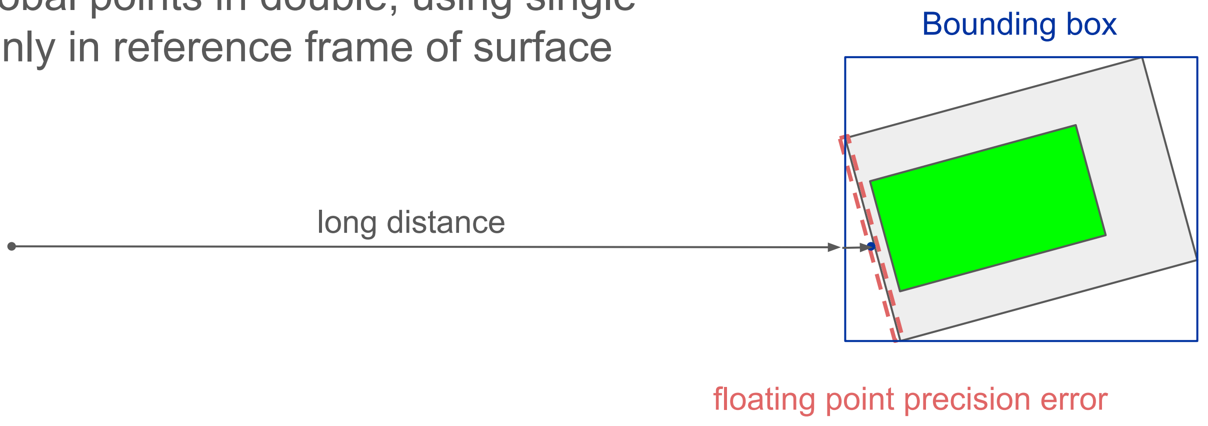
… but single precision must be used with great care

Rounding error leads to **missing** **inner Box**
Cannot use single precision all the way!



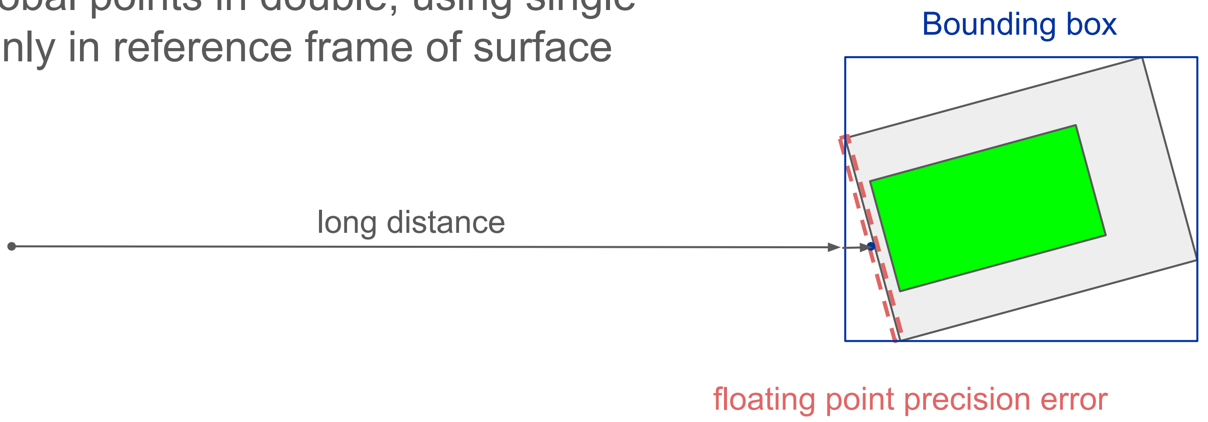
… but single precision must be used with great care

Keeping global points in double, using single precision only in reference frame of surface



… but single precision must be used with great care

Keeping global points in double, using single precision only in reference frame of surface



Many other challenges!

Status: 1/10 mio rays fail in CMS TestBeam geometry

1.5x speedup for raytracing in comparison to solid model

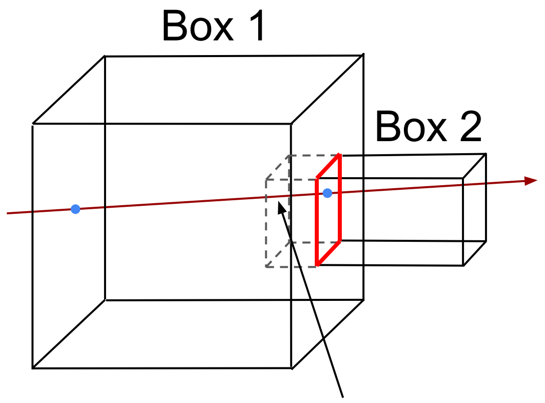
Summary and Outlook

- Tremendous progress on the surface model
- Still slightly behind solid model in full AdePT simulations
- Further optimization ongoing, many things to work on
- Promising avenue with mixed precision, but challenging

Overlaps in geometries more problematic for surface model

- **Overlaps** in the geometry lead to wrong results
- Correctness achieved with overlap detection + relocation
- Relocation expensive & source of divergence

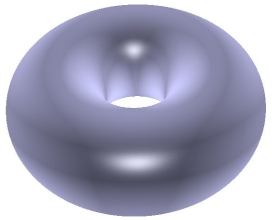
  Needs separate kernel launch

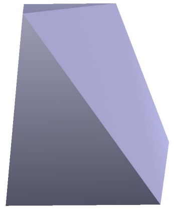


Missing the overlapping entering surface leads to missing Box 2 entirely

Supporting generality makes the code complex

making torus + parabolic surfaces compile-time option



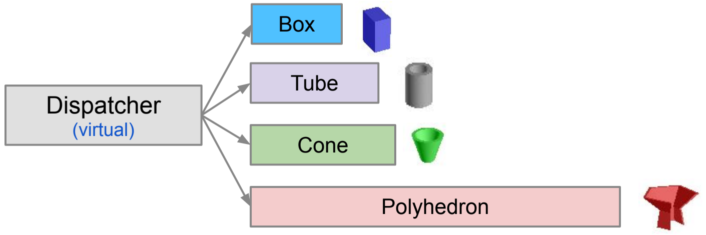


also elliptical tube

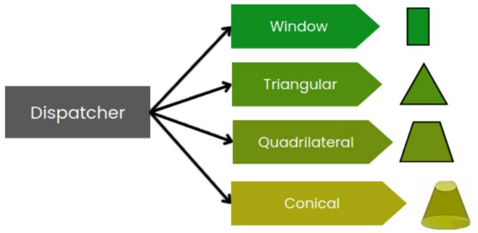
	FindNextVolume	Relocation	Theo. Max. Occupancy
Solids	256	220	16%
Surfaces (double)	146 **123** **115**	142	25% **33%** **33%**
Surfaces (mixed)	123 **86** **84**	122	33% **50%** **50%**

Different surfaces still generate divergence

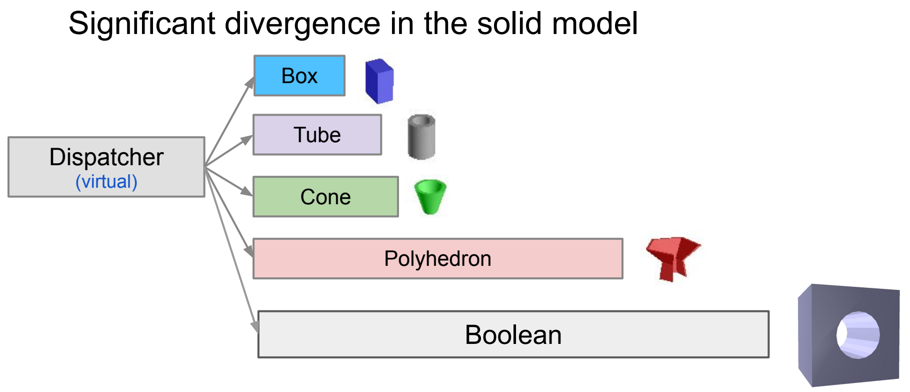
Significant divergence in the solid model

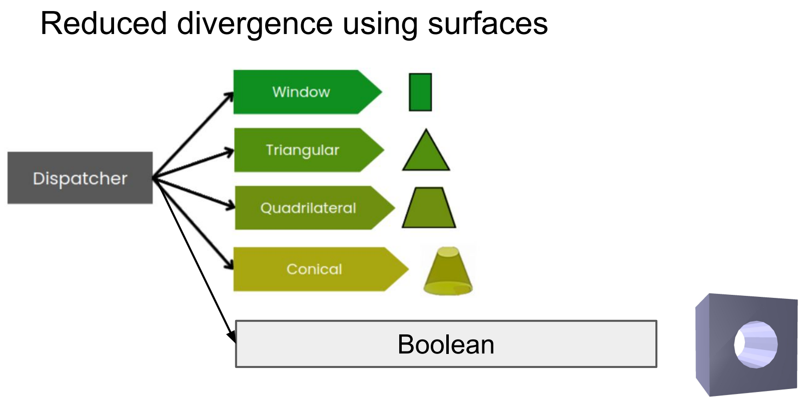


Reduced divergence using surfaces



Different surfaces still generate divergence





Boolean solids can have virtual surfaces → require full logic evaluation of all surfaces → expensive!

Solution: reduce number of boolean hits by marking "safe" surfaces at construction

Mixed precision works well for safety calculations

10 mio random points in CMS2026D110

Safety	surfaces (DP)	surfaces (MP)	solids
run time (s)	**5.84**	**0.76**	0.31
register / thread	**117**	**74**	196
Comp throughput	86 %	61 %	83 %
Mem throughput	13 %	61 %	16 %
Occupancy	**22 % / 33 %**	**31 % / 50 %**	12 % / 16 %

Not optimized! Stricter calculations than solid model (i.e., larger safeties), BVH does not yet improve performance on GPU