GPU-based algorithms for primary vertex reconstruction at CMS

CHEP 2023,

May 8th 2023, Norfolk

Carlos Erice on behalf of the CMS Collaboration

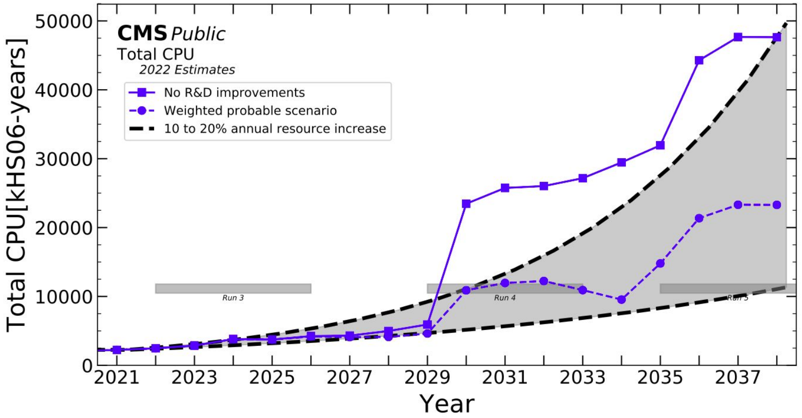
HL-LHC challenges

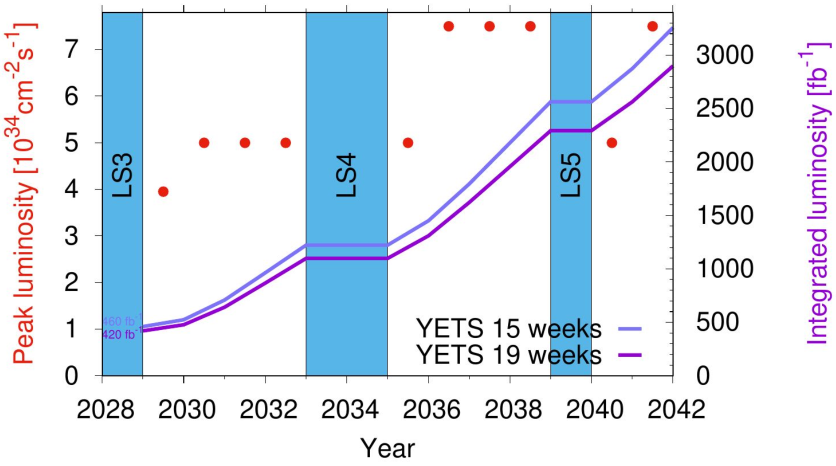
The Phase II of the LHC will lead us into the **high-luminosity regime:**

→ An instantaneous luminosity increase: more data taken per second. We will need a triggering system with a fast an efficient response to guarantee physics coverage. Challenges to the trigger/DAQ system.

→ And an increase in integrated luminosity: more data taken overall. Increasing the computational load for processing both data and -comparable amounts of- simulation. Challenges to the offline processing chain => **Focus of this talk**.

Both have a computational impact: need to do R&D to optimize our resource usage will keeping physics performance.

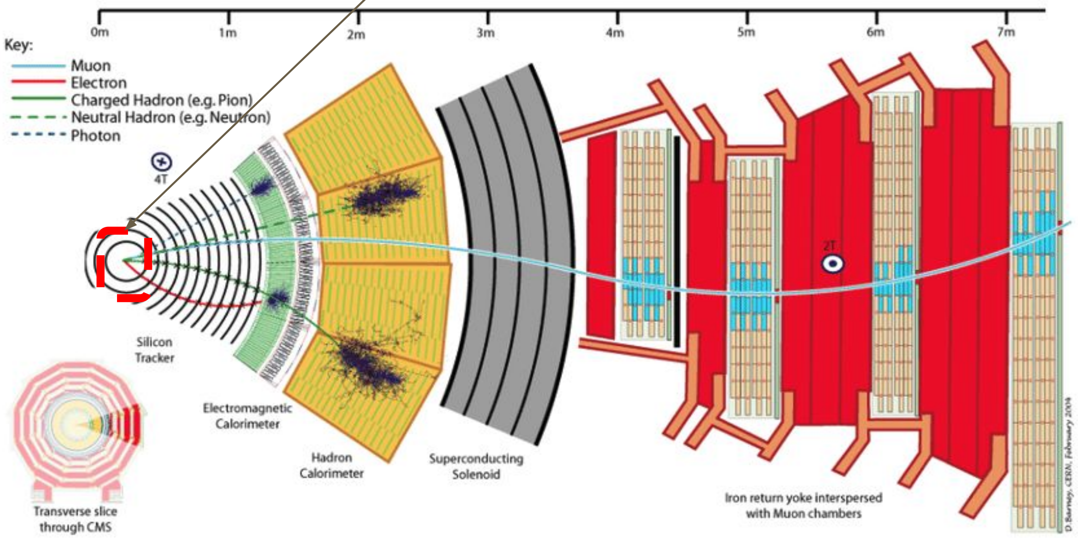


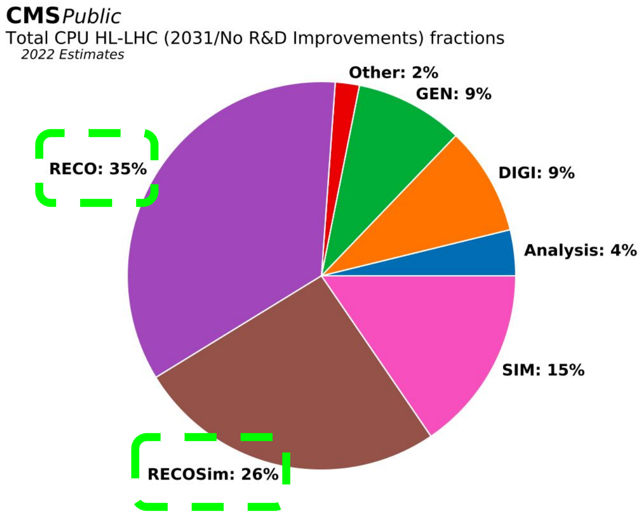


Computational challenges

→ CMS -offline- reconstruction takes ~⅔ of the pie in terms of resource usage. Optimize the reconstruction algorithms is key in the HL-LHC program.

→ Amongst it, vertex reconstruction roughly takes ~8-10% of the reconstruction time: How can we improve it?





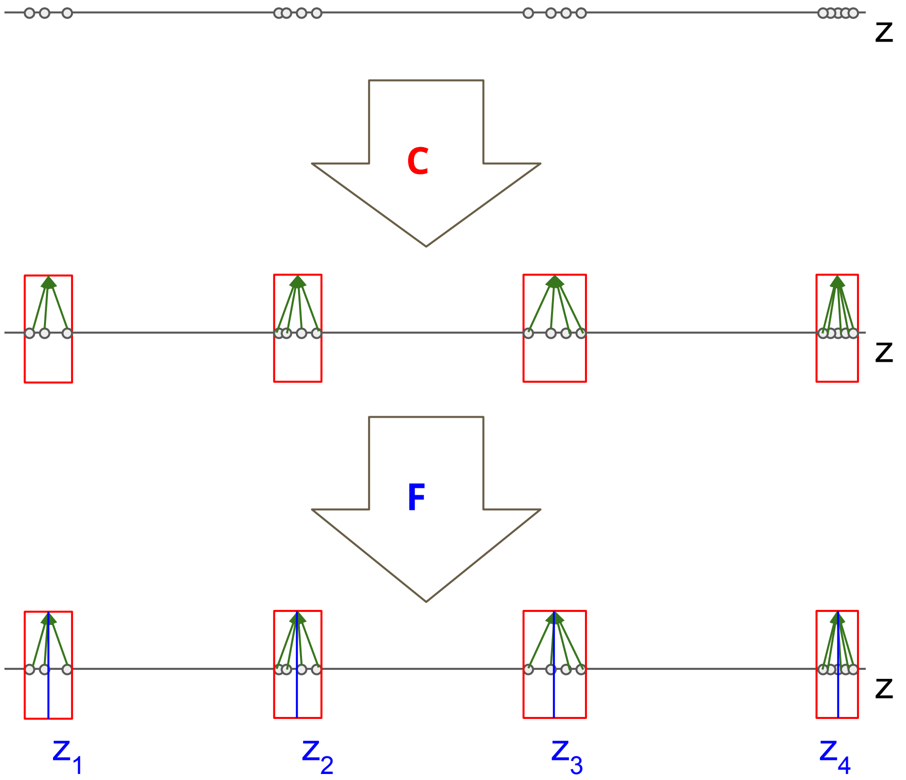
Vertexing - The overall idea

→ CMS vertexing starts from a set of tracks (~4000-8000 at Phase II of the LHC). Then proceeds into two steps:

1) **Clustering**: group together close-by tracks in cluster candidates. The algorithm used is **deterministic annealing**.
2) **Fitting**: fit vertex properties of those clusters from those of the tracks. The algorithm used is **Adaptive vertex fitting algorithm**.

Both involve computations across **~1000s of tracks and ~100s of vertices**.

Can we do better than the Legacy algorithms? Can we do an heterogeneous implementation? What can we learn from it?



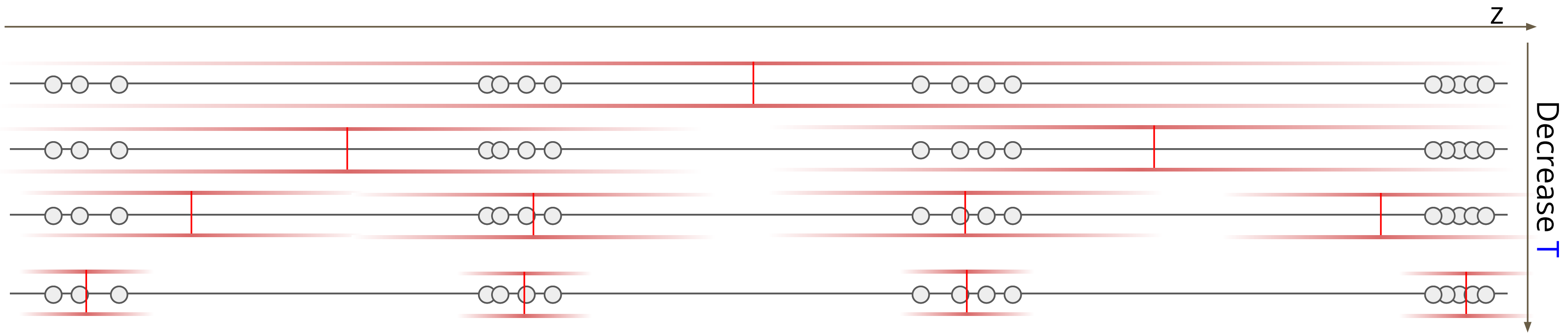
Deterministic annealing - at a glance

Deterministic annealing (DA) is based on optimizing an **energy** (assignment) function with a penalization entropy term:

Tracks

Vertices

$$E - TS = \sum_{i=1}^{I} \sum_{k=1}^{K} P_{ik} \frac{(z_i - z_k)^2}{\sigma_i^2} + T \sum_{i=1}^{I} \sum_{k=1}^{K} P_{ik} \log(P_{ik})$$



Starting at very high temperature (T) all tracks are assigned to one single cluster.
As we lower T, splitting the cluster into several (increase K) becomes beneficial.
Iteratively update assignment probabilities P_{ik} while lowering T provides a final robust estimation of the clusters.

DA - The heterogeneous architecture solution

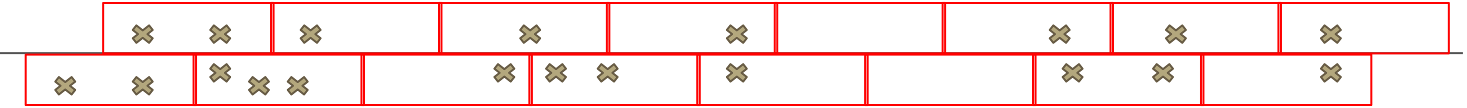
Analytical formulae for estimating P_{ij}, z_k at each iterative step exist but they are computationally intensive.

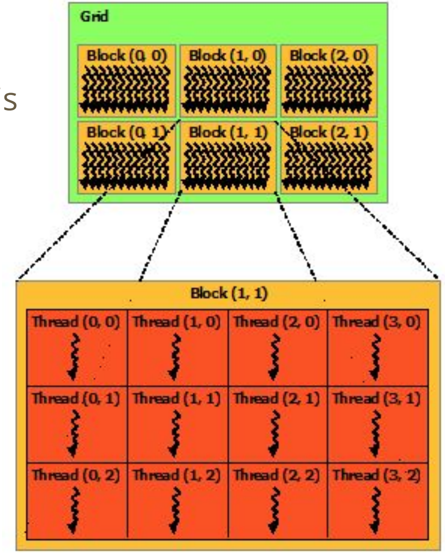
Many simple operations in parallel => A perfect place for the **usage of GPUs**.

$$p_{ij}^{(n)} = \frac{e^{-\beta\left(\frac{z_i - z_k^{(n-1)}}{\sigma_i^2}\right)^2}}{\sum_{l=1}^{K} e^{-\beta\left(\frac{z_i - z_l^{(n-1)}}{\sigma_i^2}\right)^2}}$$

Needs to be estimated ~N_{Tracks} x N_{Vertex} of times in every loop iteration!

Ideally we would use "**one GPU thread=one track**" to ensure maximum parallelization, but this would consume full resources of most commercial grade GPUs. Instead, the problem is simplified:





"Multiblock" approach: sort tracks along z, then split them in overlapping blocks of ~512 tracks:

→ Limits computational complexity significantly, allows for multithreading "per-block".

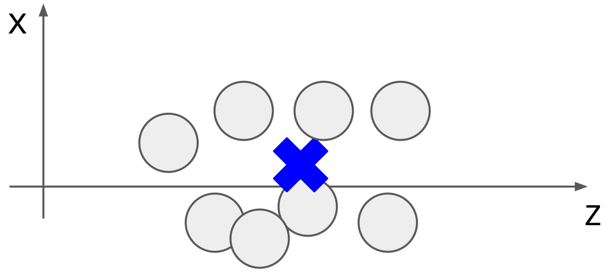
→ Blocks can run asynchronous => Better usage of device resources.

→ Matches the "block" organization of threads in GPUs!

Fitting - Weighted means fitter

Adaptive vertex fitting (Legacy Run II algorithm) is quite complex: involves iterative annealing+kalman filter-like steps.

We searched for an alternative solution that could improve performance and easily run both in CPU and GPU.



Weighted mean fitter:

→ Vertex position along i-th coordinate determined iteratively with a weighted mean of i-th coordinates of the tracks,

→ Using the error of the track's impact point along i-th coord. as weight

→ Plus outlier rejection: tracks that are 3σ away from the vertex candidate are rejected.

$$t_{i,x}^2 = \frac{(x_i - \bar{x})^2}{(\sigma_{i,x}^2 + \sigma_{\bar{x}}^2)}$$

If $\forall j \;\; t_{i,j} < 3 \rightarrow \tilde{\omega_i} = 1$

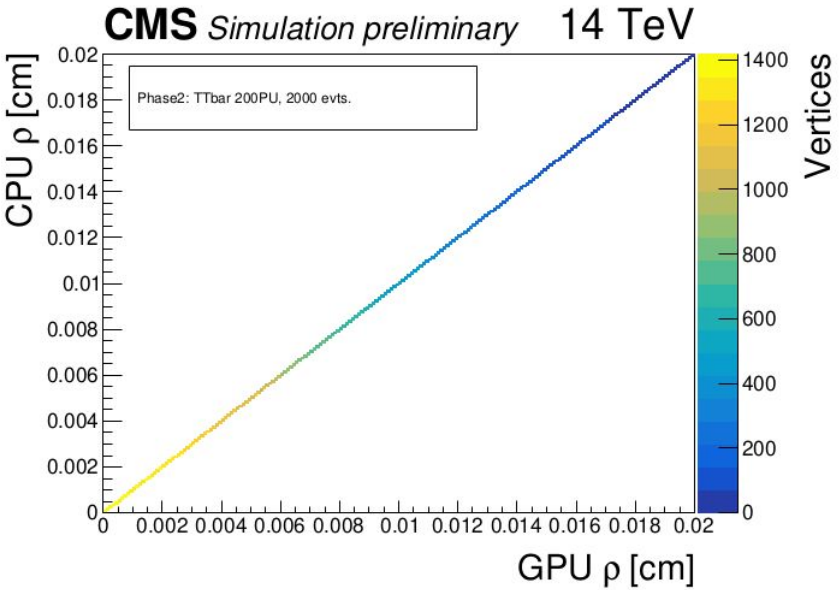
Else $\tilde{\omega_i} = 0$

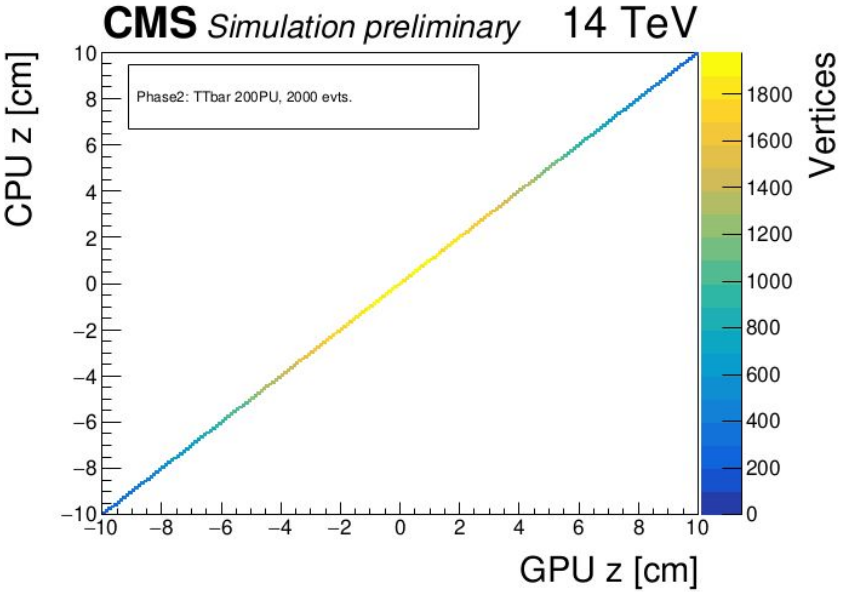
$$\omega_i = \frac{\tilde{\omega_i}}{\sigma_i^2} \qquad \bar{x} = \frac{\sum \omega_i x_i}{\sum \omega_i} \qquad \sigma_{\bar{x}}^2 = \frac{\sum \omega_i \tilde{\omega_i}}{(\sum \omega_i)^2}$$

Formulae are run iteratively on each cluster (vertex candidate) until converging to "fitted" parameters.

Implementation and consistency

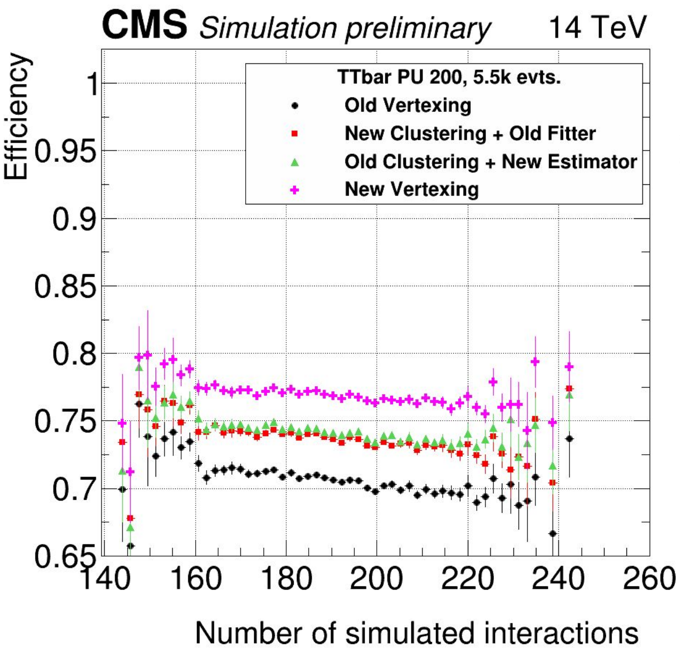
The clustering algorithm has been implemented as part of the CMS reconstruction chain for running in both CPU and using GPU acceleration based on CUDA. We find them to be **<u>fully consistent in terms of the properties of the reconstructed vertices</u>**. Here: coordinates of ~2000 events simulated at Phase II conditions using the CPU (X axis) and GPU (Y axis) implementations.



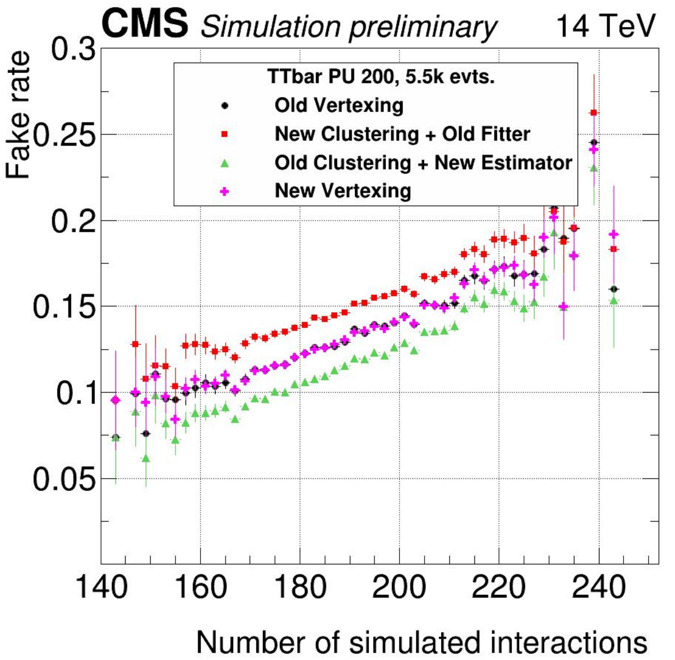


A first version of the fitting, currently running on CPU hardware, has also been implemented.

Physics performance



Efficiency: proportion of simulated vertex that are matched to a fully reconstructed one



Fake-rate: proportion of reconstructed vertex not matched to a simulated one.

Performance measurements on samples simulated on Phase II conditions show the overall improvements obtained by the new algorithms:

→ Overall ~5-6% in efficiency, as there are now ~twice the opportunities of reconstructing a vertexing due to the multiblock overlap

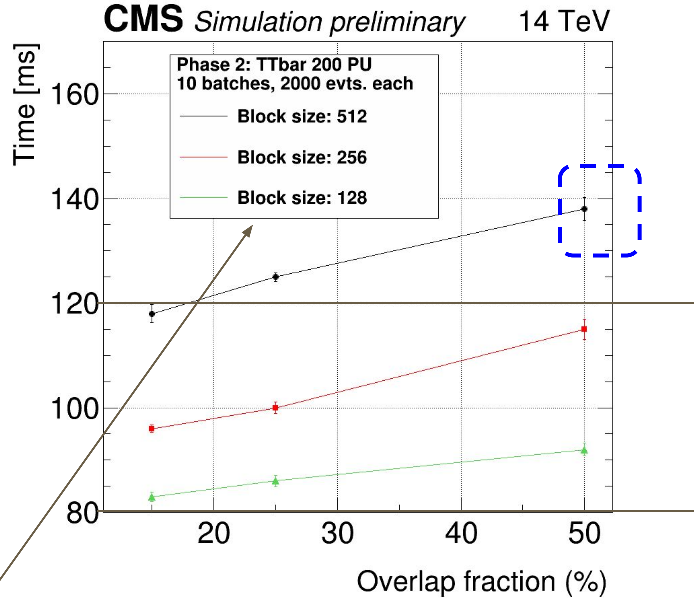
→ The corresponding increase in fake-rates is mitigated due to the additional rejection power provided by the new fitting step.

Note: new fitting slightly larger errors that leads to an increased amount of gen-reco matching

Why do we gain even in CPU?

$$p_{ij}^{(n)} = \frac{e^{-\beta\left(\frac{z_i - z_k^{(n-1)}}{\sigma_i^2}\right)^2}}{\sum_{l=1}^{K} e^{-\beta\left(\frac{z_i - z_l^{(n-1)}}{\sigma_i^2}\right)^2}}$$

CPU clusterizer + CPU fitter

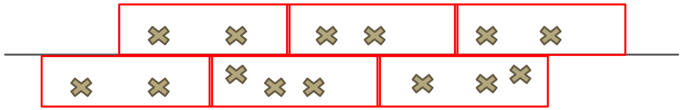


How many tracks per block

How much blocks overlap

Performance increases already in the CPU due to the decrease in the complexity of the algorithm as we dramatically decrease the number of track-vertex association needed:

Single block:
~200 vertices x ~10000 tracks => $2 \cdot 10^6$ P_{ij} values



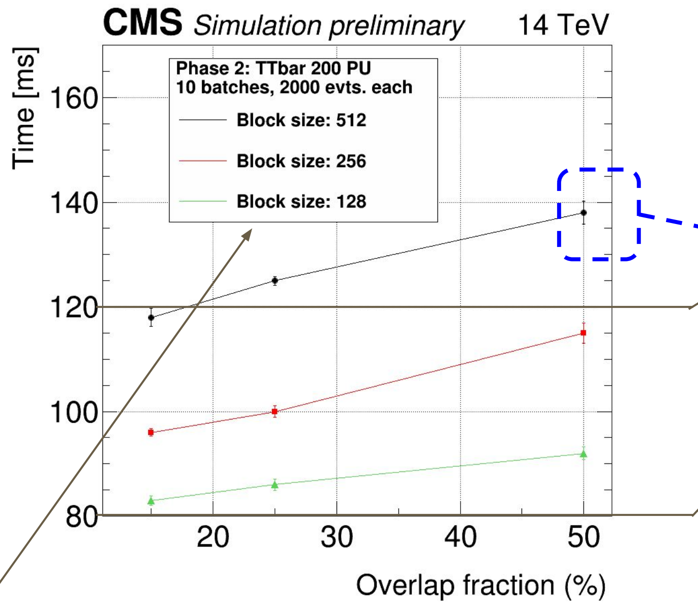
512 track block: split the 10000 tracks in 40 overlapping blocks
40 blocks x 10 vertices x 500 tracks => $2 \cdot 10^5$ P_{ij} values

- Effectively we are **transforming the problem of clustering at <PU> ~ 200 into 40 overlapping problems of clustering at <PU> ~ 10**.

Timing performance (I)

The main motivation of the updated algorithms, we want to improve the **~900 ms/evt** of the current algorithms.
→ CPU measured in a Intel Skylake Gold CPU with a single process:
→ GPU measured with nvProf in a Tesla T4 running CUDA:

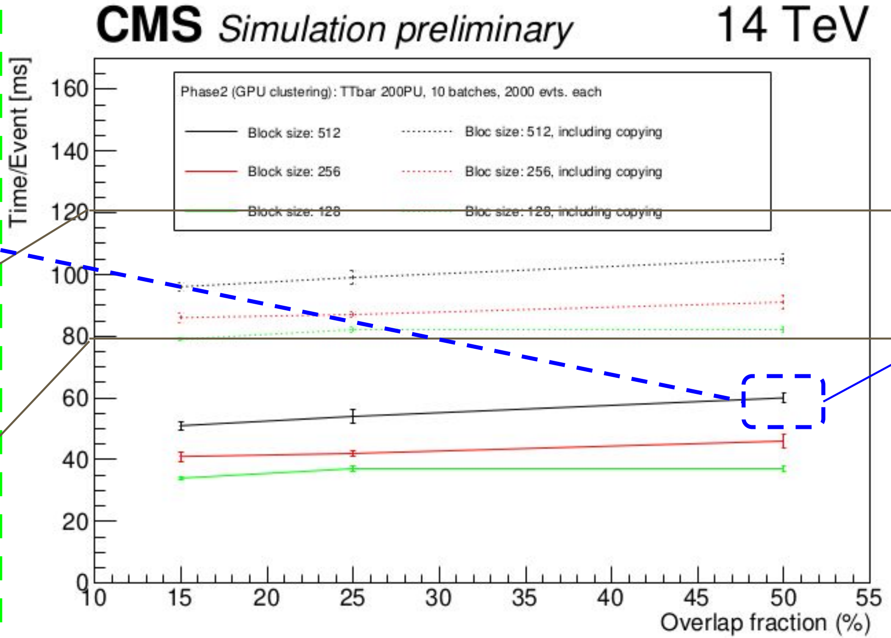
CPU clusterizer + CPU fitter



How many tracks per block

How much blocks overlap

GPU clusterizer + CPU fitter



Working point
Used in eff/fakerate plots

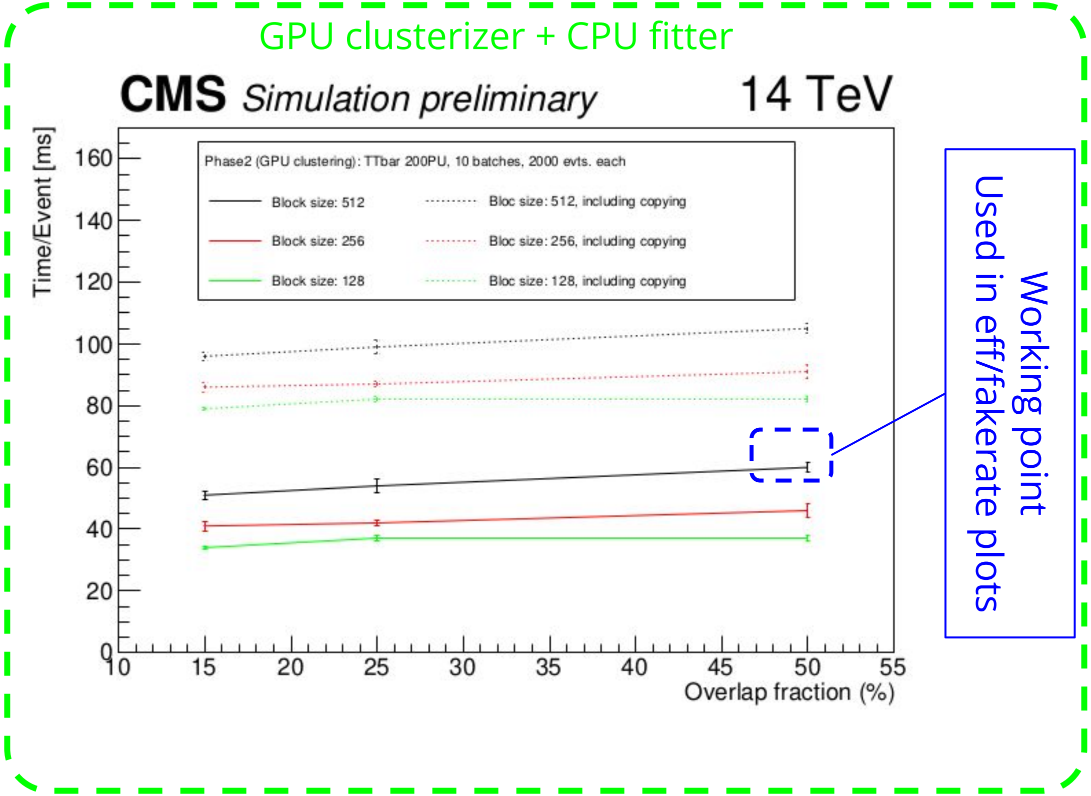
Up to **~17x speed increase** with respect to Legacy algorithm

Timing performance - To copy or not to copy

Clear distinction into the two measurements quoted in the GPU case:

1) **“Pure computational time” (full lines):** includes all the time the GPU/CPU are actually doing computations.
2) **“Including copying” (dotted lines):** includes time spent copying information between the GPU and CPU hardware (i.e. input to the clustering and output from clustering to fitting).

We include both for completeness, but in a realistic Phase II setup the whole CMS reconstruction chain would run in GPU => Just depend on the pure



Summary

- Presented an optimized algorithm designed for running offline vertexing in the CMS experiment during the Phase II of the LHC.

- A design based on compatibility with heterogeneous architectures shows improvements on both physics performance and in timing. Leading to up to ~6.3x faster algorithm.
  - Greater improvement if previous/posterior steps of the CMS reconstruction chain are offloaded to heterogeneous architectures.
  - Reduced if one takes into account pricing differences between CPU and GPU hardware.
- Several plans to provide further improvement towards Phase II:
  - Include timing information from the -new- MTD detector.
  - Usage of portability libraries (Alpaka, see Andrea’s talk) to profit from other non-nVidia hardware.

References

[1] Chabanat, E; Estre, N; Deterministic Annealing for Vertex Finding at CMS, 2005, 10.5170/CERN-2005-002.287

[2] Frühwirth, R ; Waltenberger, W. ; Vanlaer, P.; Adaptive Vertex Fitting, 2007, CMS-NOTE-2007-008

[3] CMS Collaboration; Primary Vertex Reconstruction for Heterogeneous Architecture at CMS, 2022, CERN-CMS-DP-2022-052

Backup