Deep-learning-based tracking demonstrator for the HL-LHC

Jan Stark

Laboratoire des 2 Infinis – Toulouse (L2IT)

CAF users meeting, November 21st 2023

Approach of this talk

Will not give a comprehensive overview of techniques that have developed, nor of the striking the results that have been obtained. These are described in detail in other talks, e.g.:

- Charline Rougier at *Connecting the Dots 2022*, Princeton (clickable link)

- Xiangyang Yu at *CHEP 2023*, Norfolk (clickable link)

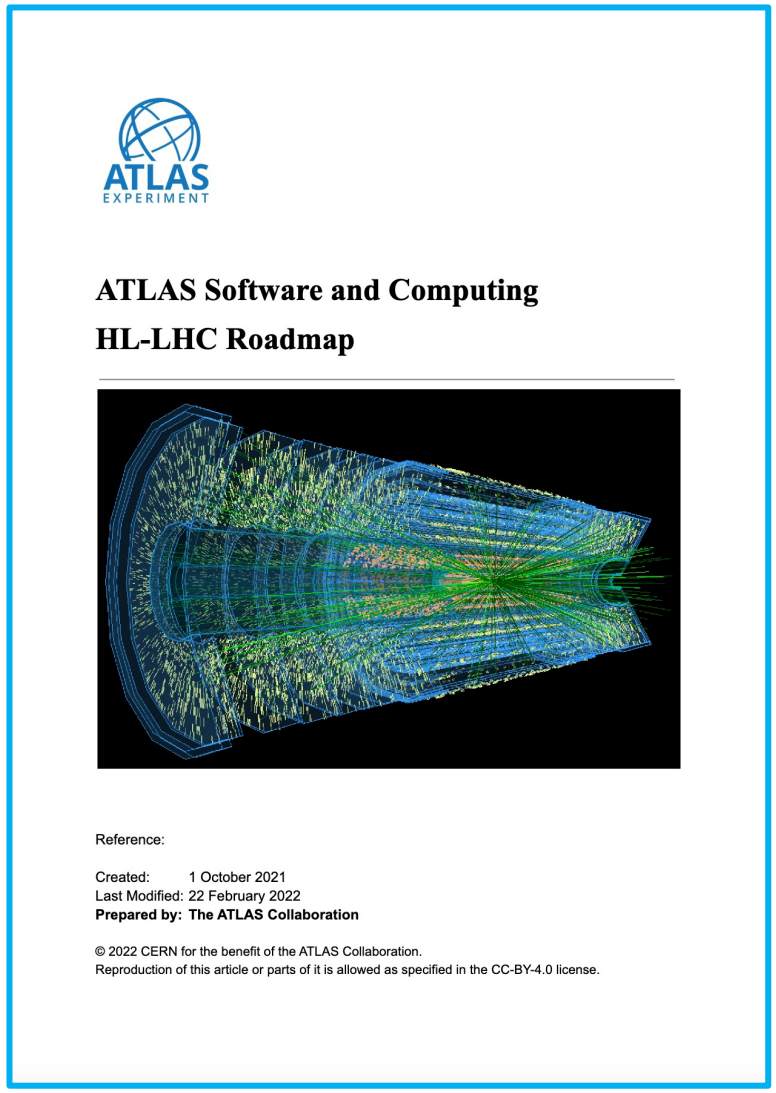
- Sylvain Caillou at *CHEP 2023*, Norfolk (clickable link)

- Heberth Torres at *Connecting the Dots 2023*, Toulouse (clickable link)

In instead, will try to describe our work in the bigger context.

ATLAS HL-LHC S&C roadmap

(clickable link)

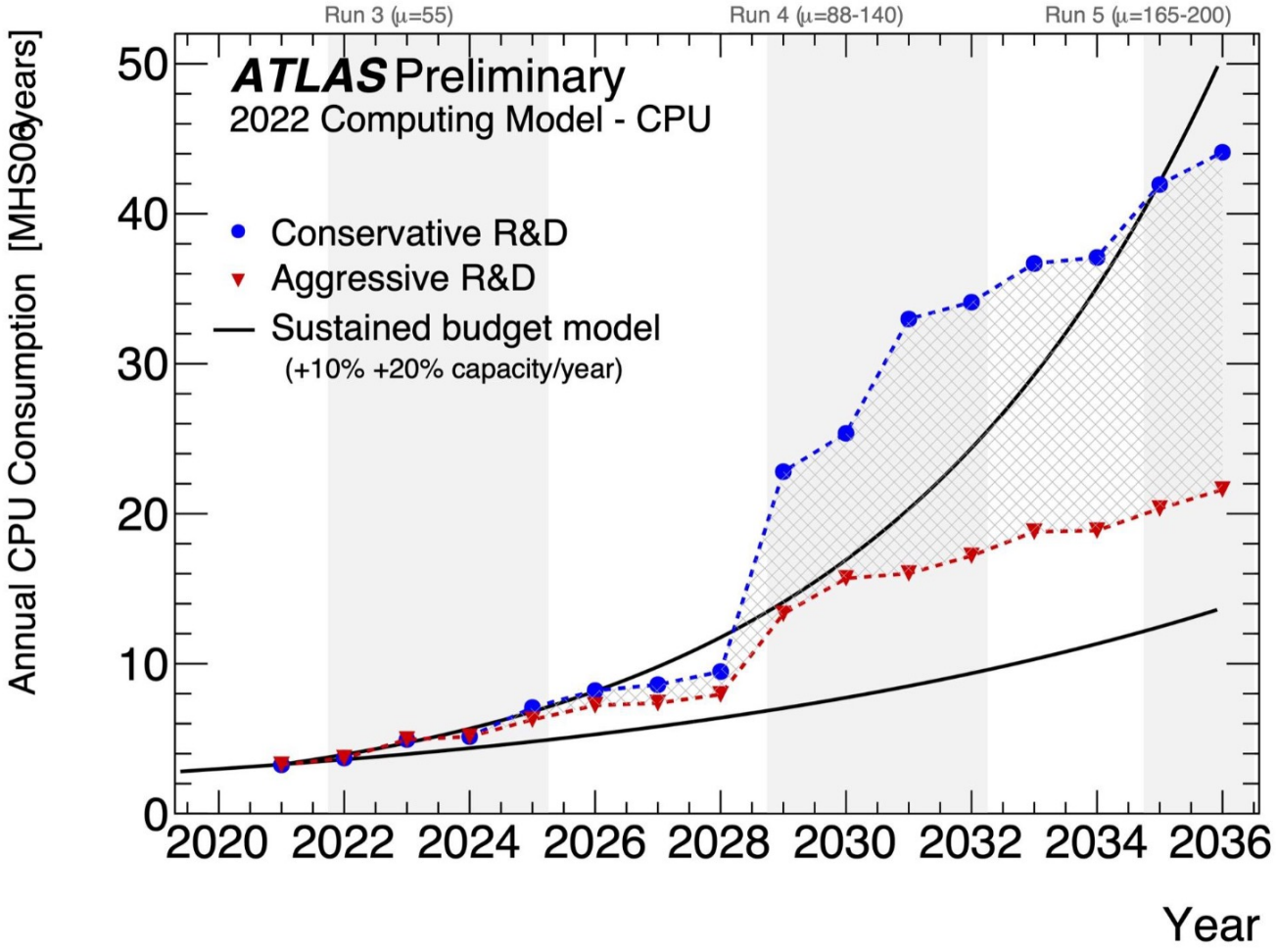


ATLAS EXPERIMENT

ATLAS Software and Computing HL-LHC Roadmap

Reference:

Created: 1 October 2021
Last Modified: 22 February 2022
Prepared by: The ATLAS Collaboration



Predictions are difficult, especially when they concern the future (George Bernard Shaw, Winston Churchill or Niels Bohr)

What Future Processors Will Look Like

2k Shares | 72 | 286 | 1.1k

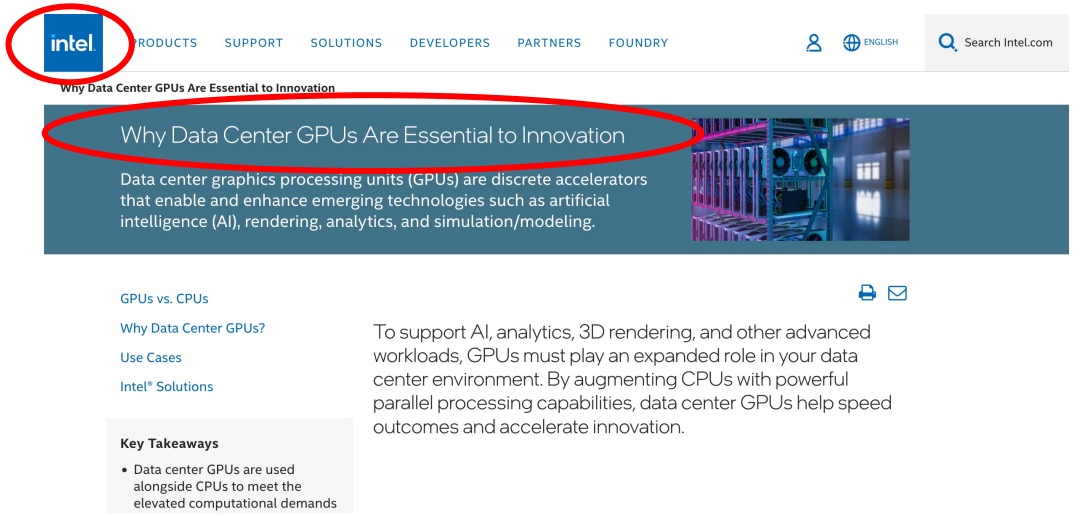
AMD CTO Mark Papermaster talks about why heterogeneous architectures will be needed to achieve improvements in PPA.

JULY 13TH, 2022 - BY: **ED SPERLING**

> Mark Papermaster, CTO at AMD, sat down with Semiconductor Engineering to talk about architectural changes that are required as the benefits of scaling decrease, including chiplets, new standards for heterogeneous integration, and different types of memory. What follows are excerpts of that conversation.

SE: What does a processor look like in five years? Is it a bunch of chips in a package? Is it a CPU and FPGA and GPU?

Papermaster: There is no question that the future of processing is heterogeneous. It's multiple compute engines working in tandem, because massive data and graphics processing is needed everywhere. It's needed in data centers and in PCs, and the explosion of data from the Internet of Things requires analysis and visualization across that whole food chain. There clearly is a requirement of domain-specific architectures. The CPU is fantastic for general processing, and there are a gazillion applications that run on x86 and on Arm. For more specialized graphics and vector processing, FPGAs or ASICs can provide very specialized computing. We saw this future at AMD well over a decade ago, and we pointed our R&D efforts toward this future. We've been in mass production for years with an APU that combines the CPU and GPU for our client and embedded markets, and now we're bringing that APU into the



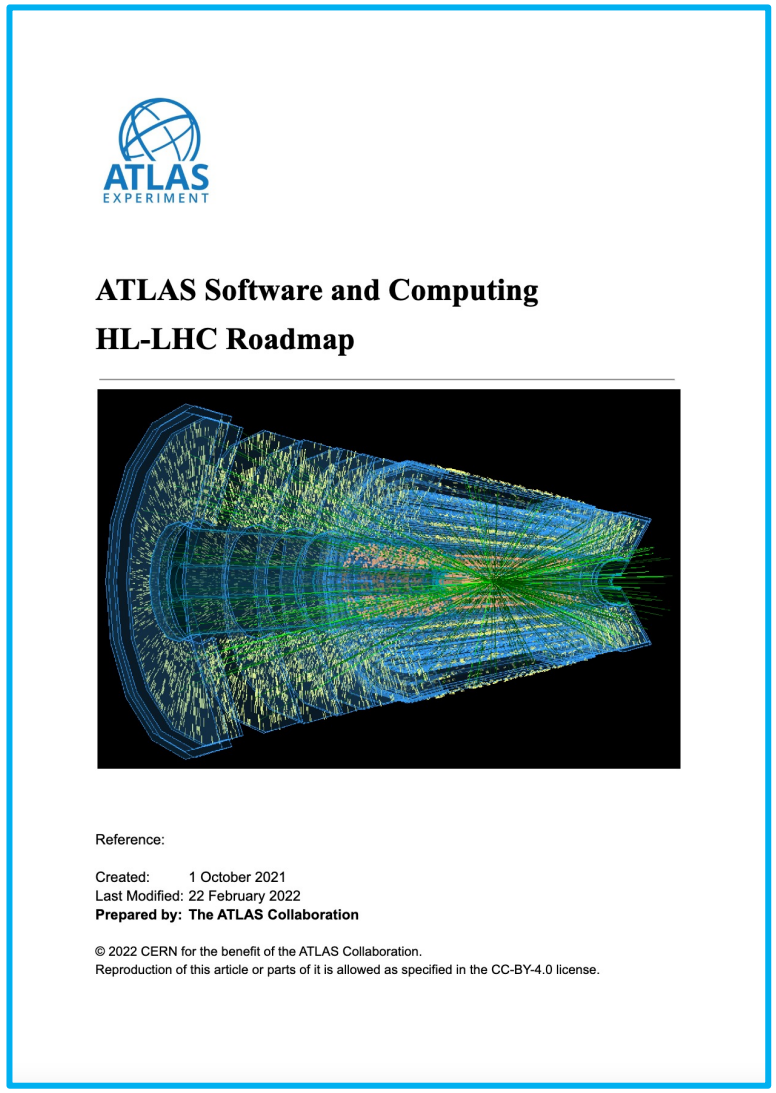
CTO = Chief Technical Officer

PPA = Power, performance, area

I don't have a crystal ball either to predict the future.
But I think that, as a field, we must be able to
run our software on the GPU-heavy heterogeneous
architectures that may well be the future.

ATLAS HL-LHC S&C roadmap

(clickable link)



ATLAS EXPERIMENT

ATLAS Software and Computing HL-LHC Roadmap

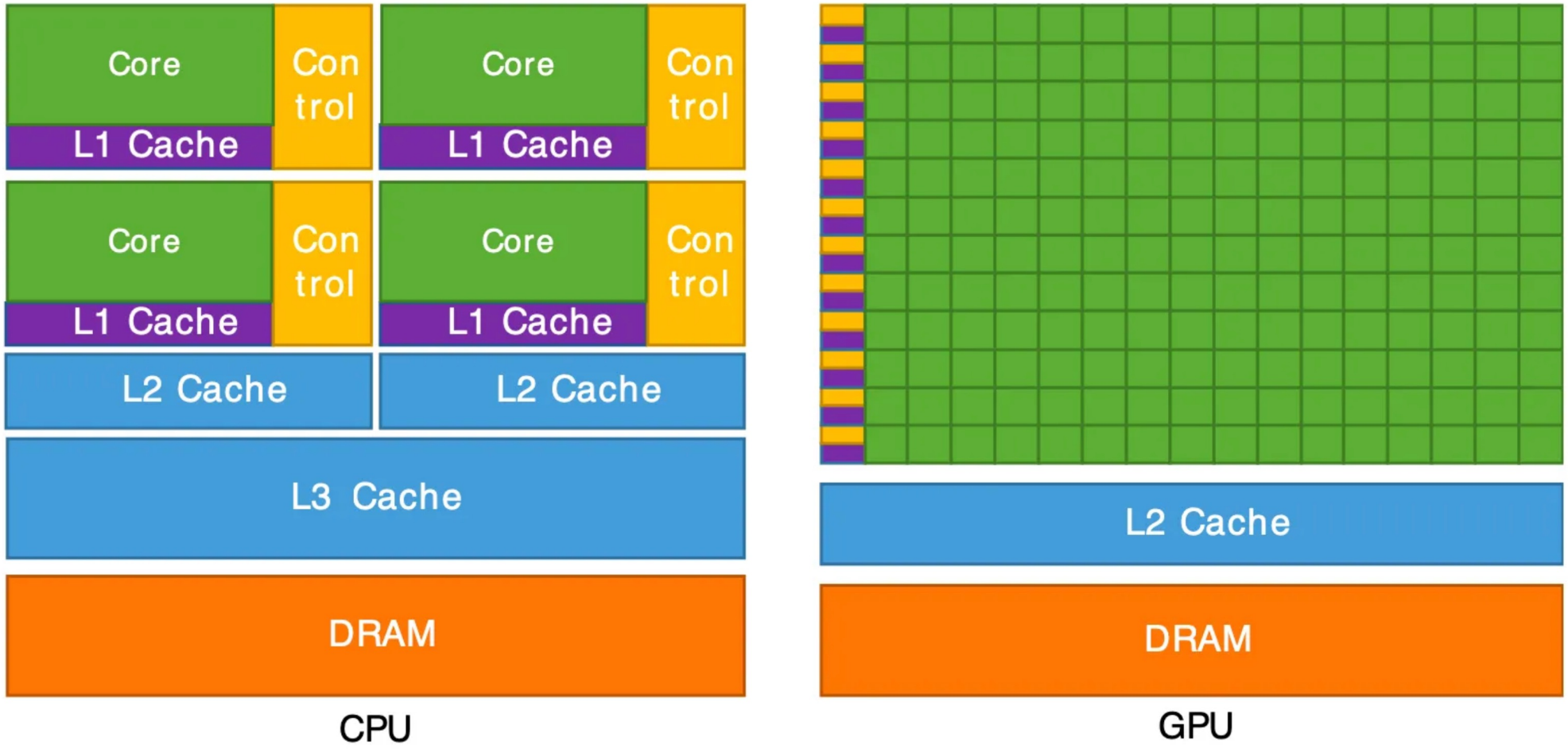
Reference:

Created: 1 October 2021
Last Modified: 22 February 2022
Prepared by: The ATLAS Collaboration

Project Organization				
	MID	**DID**	Description	Due
	PR-1		First evaluation of effort needed to deliver on HL-LHC milestones	Q2 2022
	PR-2		HL-LHC Computing TDR	Q3 2024
		2.1	R&D projects targeting Run 4 ("Run 4 projects) define scope and potential impact of their demonstrators, and a program of work with effort and risk estimates to the end of Phase 2.	Q4 2022
		2.2	Define release, datasets and platforms to be used to evaluate Run 4 performance impact of demonstrators	Q1 2024
		2.3	Run 4 projects release their demonstrators	Q2 2024
		2.4	Run 4 projects evaluate the performance impact of their R&D demonstrators and estimate the effort needed to develop fully functional prototypes	Q2 2024
	PR-3		Run 4 Release	Q2 2027
		3.1	Run 4 projects release fully functional prototypes, estimate risks and effort needed to bring to production quality	Q2 2026
		3.2	Run 4 developers tutorial	Q3 2026
		3.3	Run 4 Feature Freeze	Q2 2027
	PR-4		Ready for Run 4 Data Taking	Q2 2029
		4.1	Run 4 projects demonstrate required functionality in release	Q3 2026
		4.2	Run 4 release validated	Q1 2029

A few words on SIMD and GPUs

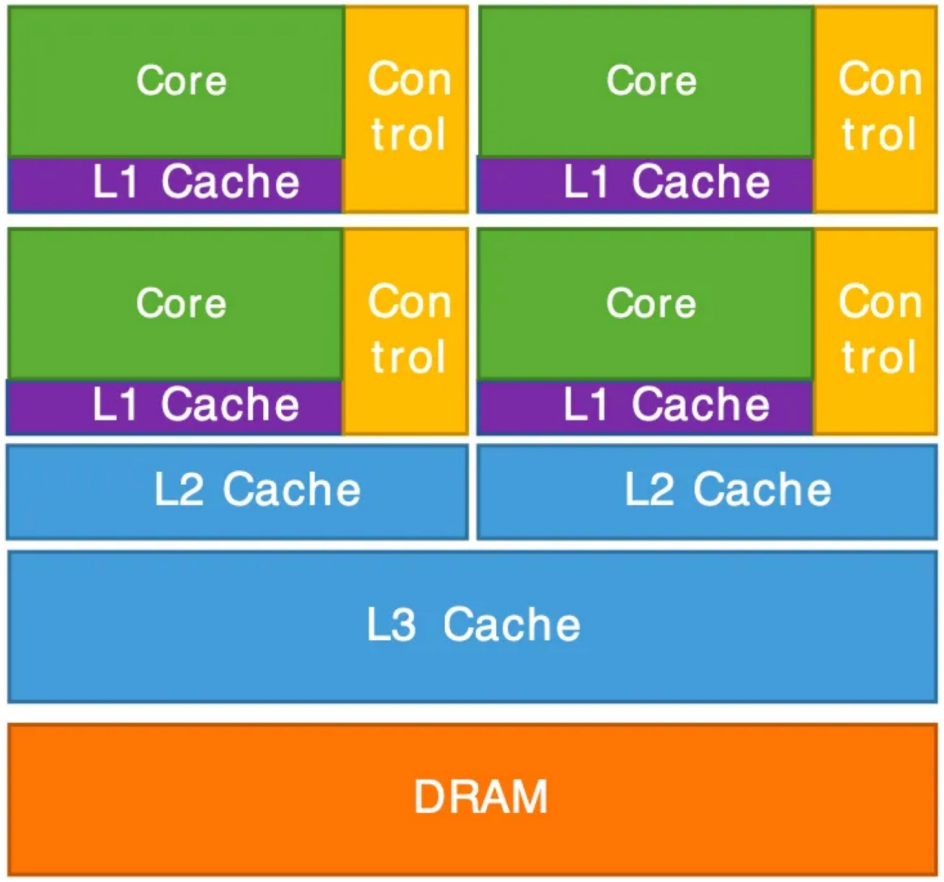
Figure taken from the CUDA C programming guide.



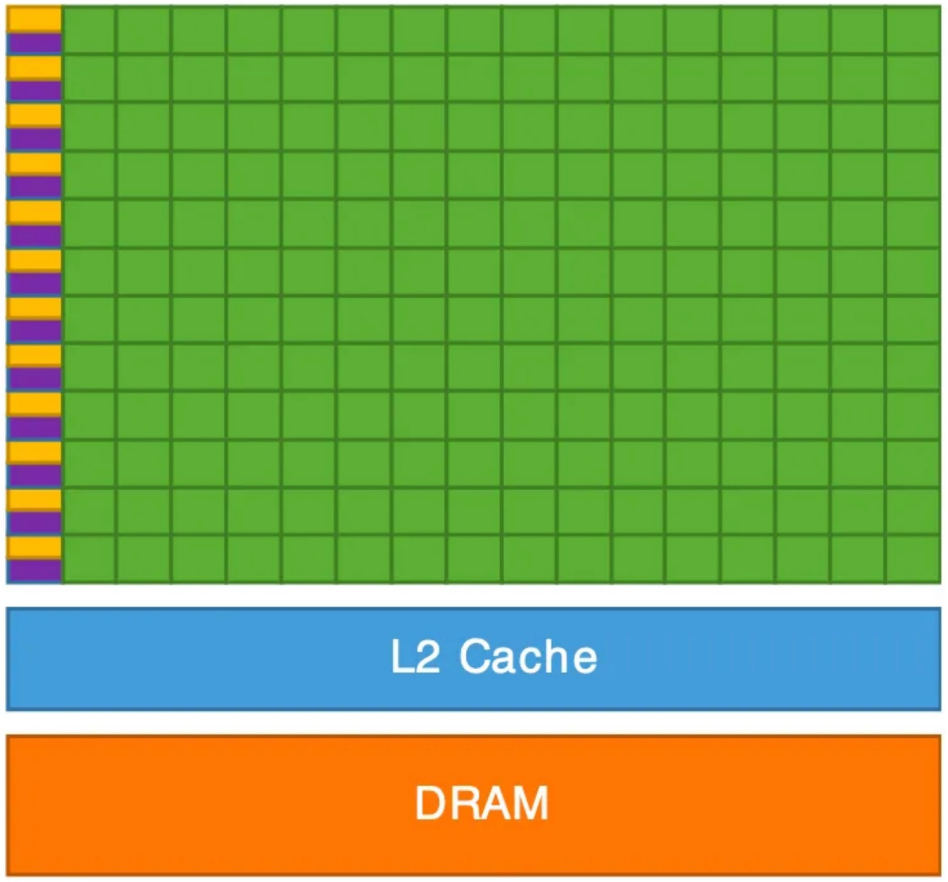
SIMD = single instruction, multiple data

A few words on SIMD and GPUs

Figure taken from the CUDA C programming guide.



CPU



GPU

Cache prefetching

1 language

Article Talk Tools

From Wikipedia, the free encyclopedia

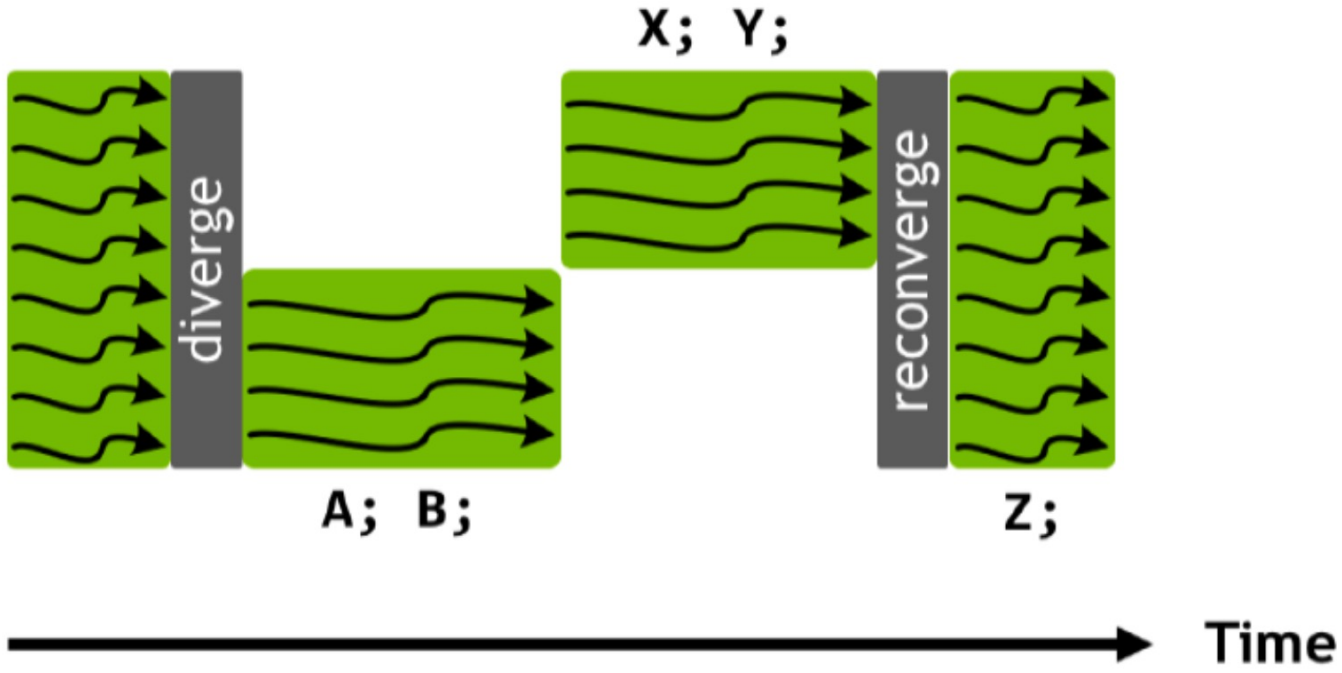
Cache prefetching is a technique used by computer processors to boost execution performance by fetching instructions or data from their original storage in slower memory to a faster local memory before it is actually needed (hence the term 'prefetch').

CPUs and the memory chips that they control spend a big part of their electricity to move around data that are never used.

SIMD = single instruction, multiple data

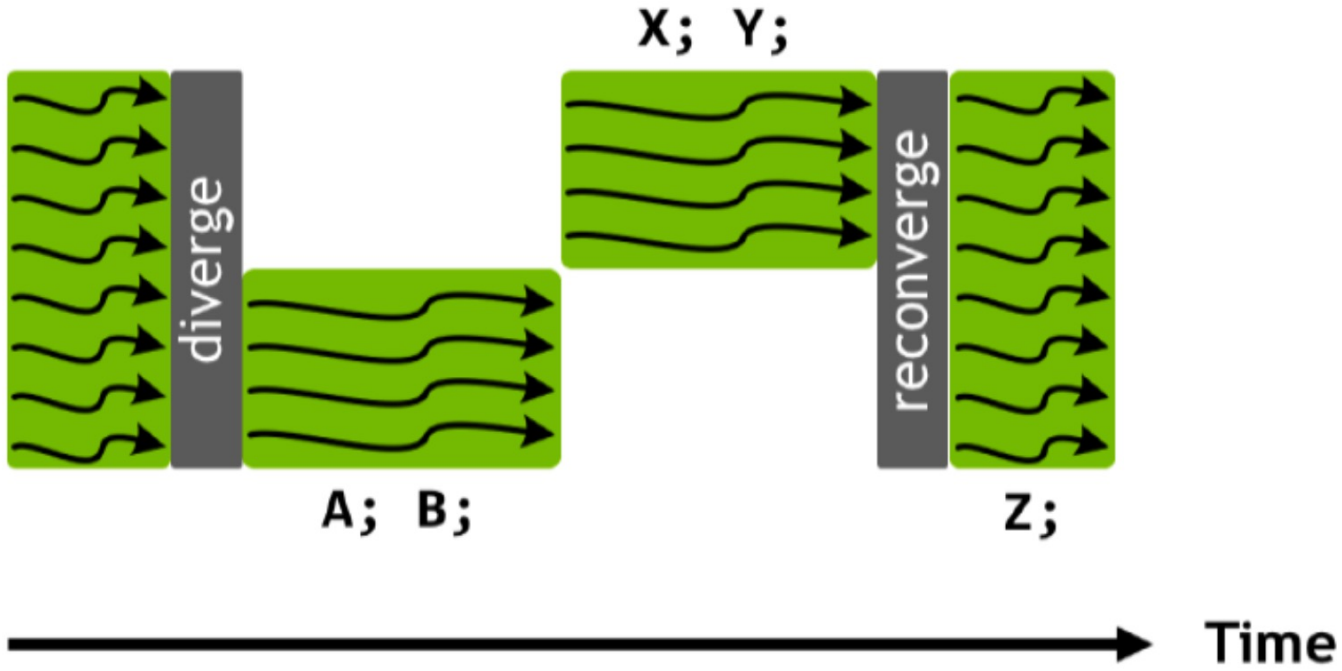
Consequence of SIMD: warp divergence in GPUs

```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```



Consequence of SIMD: warp divergence in GPUs

```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```



And this is just one "*if*". Imagine nested *if* statements … the GPU quickly becomes idle.

Need to learn to design algorithms (almost) without *if* statements … otherwise GPUs are useless.

A few words on GPU programming

Taken from the CUDA C programming guide.

5.1. Overall Performance Optimization Strategies

Performance optimization revolves around three basic strategies:

- Maximize parallel execution to achieve maximum utilization;
- Optimize memory usage to achieve maximum memory throughput;
- Optimize instruction usage to achieve maximum instruction throughput.

Which strategies will yield the best performance gain for a particular portion of an application depends on the performance limiters for that portion; optimizing instruction usage of a kernel that is mostly limited by memory accesses will not yield any significant performance gain, for example. Optimization efforts should therefore be constantly directed by measuring and monitoring the performance limiters, for example using the CUDA profiler. Also, comparing the floating-point operation throughput or memory throughput - whichever makes more sense - of a particular kernel to the corresponding peak theoretical throughput of the device indicates how much room for improvement there is for the kernel.

5.2. Maximize Utilization

To maximize utilization the application should be structured in a way that it exposes as much parallelism as possible and efficiently maps this parallelism to the various components of the system to keep them busy most of the time.

A few words on GPU programming

Taken from the CUDA C programming guide.

5.1. Overall Performance Optimization Strategies

Performance optimization revolves around three basic strategies:

- Maximize parallel execution to achieve maximum utilization;
- Optimize memory usage to achieve maximum memory throughput;
- Optimize instruction usage to achieve maximum instruction throughput.

Which strategies will yield the best performance gain for a particular portion of an application depends on the performance limiters for that portion; optimizing instruction usage of a kernel that is mostly limited by memory accesses will not yield any significant performance gain, for example. Optimization efforts should therefore be constantly directed by measuring and monitoring the performance limiters, for example using the CUDA profiler. Also, comparing the floating-point operation throughput or memory throughput - whichever makes more sense - of a particular kernel to the corresponding peak theoretical throughput of the device indicates how much room for improvement there is for the kernel.

5.2. Maximize Utilization

To maximize utilization the application should be structured in a way that it exposes as much parallelism as possible and efficiently maps this parallelism to the various components of the system to keep them busy most of the time.

An obsolete/cheap gaming GPU like the RTX 2070 (an expensive A100 datacentre GPU) can run ~1k threads (~30k threads) in parallel. In addition, they have powerful mechanisms to switch from one thread to another to avoid waiting for things (e.g. memory access).
⇒ Need to break down our algorithms into tens to hundreds of thousands of simple sub-tasks that can be executed using the SIMD paradigm. VERY HARD in experimental particle physics.

A few words on GPU programming

Taken from the CUDA C programming guide.

5.1. Overall Performance Optimization Strategies

Performance optimization revolves around three basic strategies:

- Maximize parallel execution to achieve maximum utilization;
- Optimize memory usage to achieve maximum memory throughput;
- Optimize instruction usage to achieve maximum instruction throughput.

Which strategies will yield the best performance gain for a particular portion of an application depends on the performance limiters for that portion; optimizing instruction usage of a kernel that is mostly limited by memory accesses will not yield any significant performance gain, for example. Optimization efforts should therefore be constantly directed by measuring and monitoring the performance limiters, for example using the CUDA profiler. Also, comparing the floating-point operation throughput or memory throughput - whichever makes more sense - of a particular kernel to the corresponding peak theoretical throughput of the device indicates how much room for improvement there is for the kernel.

5.2. Maximize Utilization

To maximize utilization the application should be structured in a way that it exposes as much parallelism as possible and efficiently maps this parallelism to the various components of the system to keep them busy most of the time.

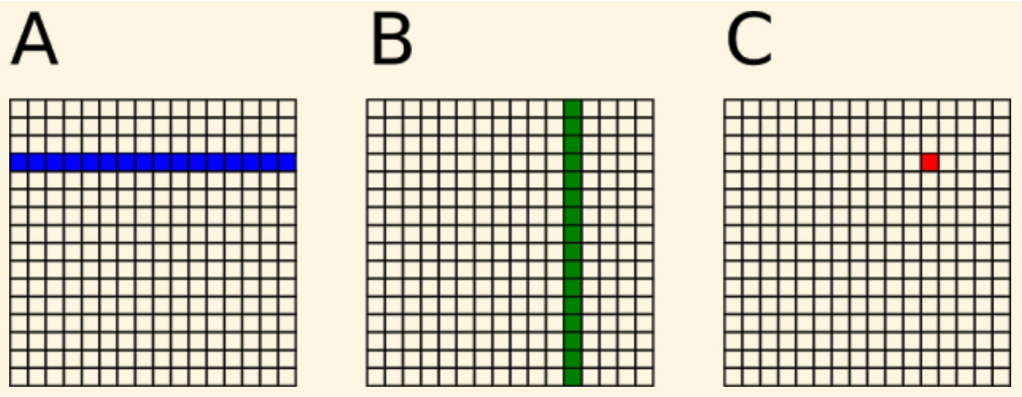
Whenever we talk about GPU usage, we should quote numbers for GPU utilisation, memory bus efficiency, etc. This is the *nerf de la guerre* in GPU computing.

If people (like the traccc demonstrator) don't give the numbers, then ask for them.

An obsolete/cheap gaming GPU like the RTX 2070 (an expensive A100 datacentre GPU) can run ~1k threads (~30k threads) in parallel. In addition, they have powerful mechanisms to switch from one thread to another to avoid waiting for things (e.g. memory access).
⇒ Need to break down our algorithms into tens to hundreds of thousands of simple sub-tasks that can be executed using the SIMD paradigm. VERY HARD in experimental particle physics.

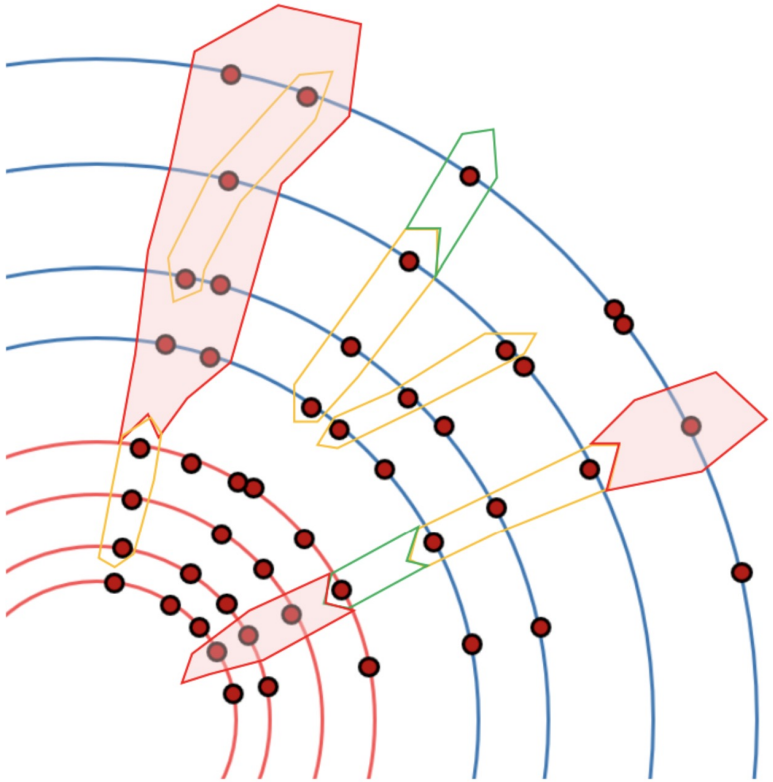
A few words on GPU programming

Example of an algorithm that runs efficiently on GPUs:
matrix operations on large matrices (e.g. A*B = C)



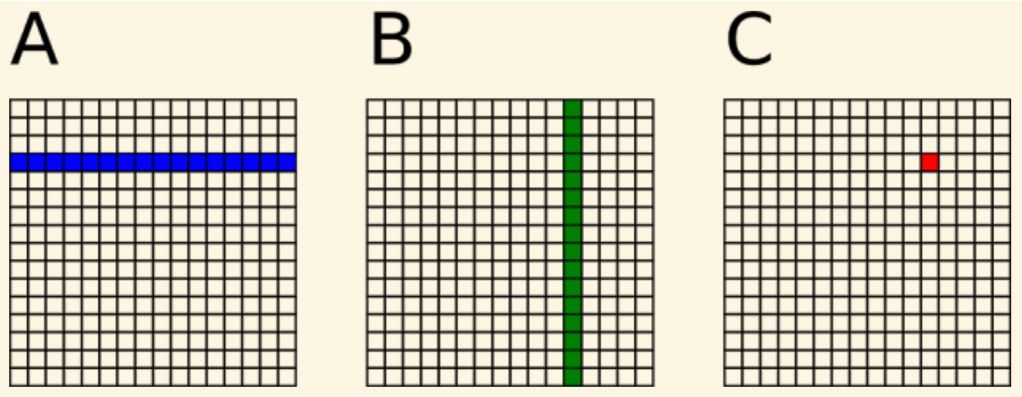
Neural networks can be expressed in terms of matrix operations.

Example of an algorithm that does not run efficiently on GPUs:
Combinatorial Kalman filtering in a detector with a complex geometry and a non-zero magnetic field.



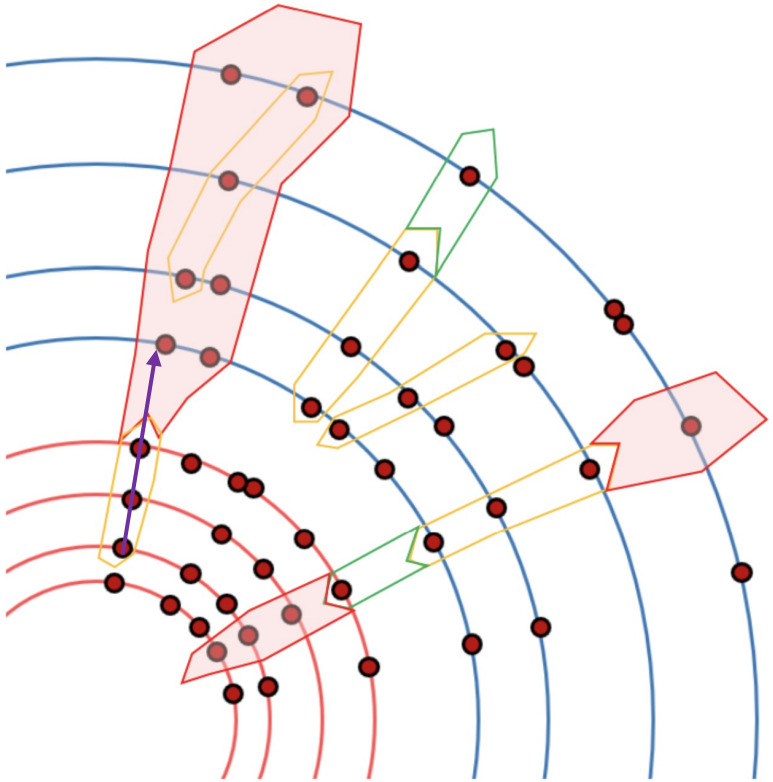
A few words on GPU programming

Example of an algorithm that runs efficiently on GPUs:
matrix operations on large matrices (e.g. A*B = C)



Neural networks can be expressed in terms of matrix operations.

Example of an algorithm that does not run efficiently on GPUs:
Combinatorial Kalman filtering in a detector with a complex geometry and a non-zero magnetic field.



Simply knowing on which detector module to look for the next hit Is a quagmire of “if” statements.
Then do this for the next ~15 layers in ITk ….

Simple detectors, simple algorithms

Computing and Software for Big Science (2020) 4:7
https://doi.org/10.1007/s41781-020-00039-7

ORIGINAL ARTICLE

Allen: A High-Level Trigger on GPUs for LHCb

R. Aaij[1] · J. Albrecht[2] · M. Belous[3,4] · P. Billoir[5] · T. Boettcher[6] · A. Brea Rodríguez[7] · D. vom Bruch[5] · D. H. Cámpora Pérez[1,8] · A. Casais Vidal[7] · D. C. Craik[6] · P. Fernandez Declara[9,10] · L. Funke[2] · V. V. Gligorov[5] · B. Jashal[11] · N. Kazeev[3,4] · D. Martínez Santos[7] · F. Pisani[9,12,13] · D. Pliushchenko[4,14] · S. Popov[3,4,15] · R. Quagliani[5] · M. Rangel[16] · F. Reiss[5] · C. Sánchez Mayordomo[11] · R. Schwemmer[9] · M. Sokoloff[17] · H. Stevens[2] · A. Ustyuzhanin[3,4,15] · X. Vilasís Cardona[18] · M. Williams[6]

Received: 18 December 2019 / Accepted: 3 April 2020 / Published online: 30 April 2020

Abstract
We describe a fully GPU-based implementation of the first level trigger for the upgrade of the LHCb detector, due to start data taking in 2021. We demonstrate that our implementation, named Allen, can process the 40 Tbit/s data rate of the upgraded LHCb detector and perform a wide variety of pattern recognition tasks. These include finding the trajectories of charged particles, finding proton–proton collision points, identifying particles as hadrons or muons, and finding the displaced decay vertices of long-lived particles. We further demonstrate that Allen can be implemented in around 500 scientific or consumer GPU cards, that it is not I/O bound, and can be operated at the full LHC collision rate of 30 MHz. Allen is the first complete high-throughput GPU trigger proposed for a HEP experiment.

Keywords GPU · Real-time data selection · Trigger · LHCb

✉ R. Aaij
raaij@cern.ch

✉ D. vom Bruch
dovombru@cern.ch

✉ D. H. Cámpora Pérez
dcampora@cern.ch

[1] Nikhef National Institute for Subatomic Physics, Amsterdam, The Netherlands
[2] Fakultät Physik, Technische Universität Dortmund, Dortmund, Germany
[3] National Research University Higher School of Economics, Moscow, Russia
[4] Yandex School of Data Analysis, Moscow, Russia
[5] LPNHE, Sorbonne Université, Paris Diderot Sorbonne Paris Cité, CNRS/IN2P3, Paris, France
[6] Massachusetts Institute of Technology, Cambridge, USA
[7] Instituto Galego de Física de Altas Enerxías (IGFAE), Universidade de Santiago de Compostela, Santiago de Compostela, Spain
[8] Faculty of Science and Engineering, Maastricht University, Maastricht, The Netherlands
[9] European Organization for Nuclear Research (CERN), Geneva, Switzerland
[10] Department of Computer Science and Engineering, University Carlos III of Madrid, Madrid, Spain
[11] Instituto de Física Corpuscular, Centro Mixto Universidad de Valencia, CSIC, Valencia, Spain
[12] INFN Sezione di Bologna, Bologna, Italy
[13] Università di Bologna, Bologna, Italy
[14] National Research University Higher School of Economics, Saint Petersburg, Russia
[15] National University of Science and Technology MISIS, Moscow, Russia
[16] Instituto de Física, Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil
[17] University of Cincinnati, Cincinnati, OH, USA
[18] DS4DS, la Salle, Universitat Ramon Llull, Barcelona, Spain

Springer

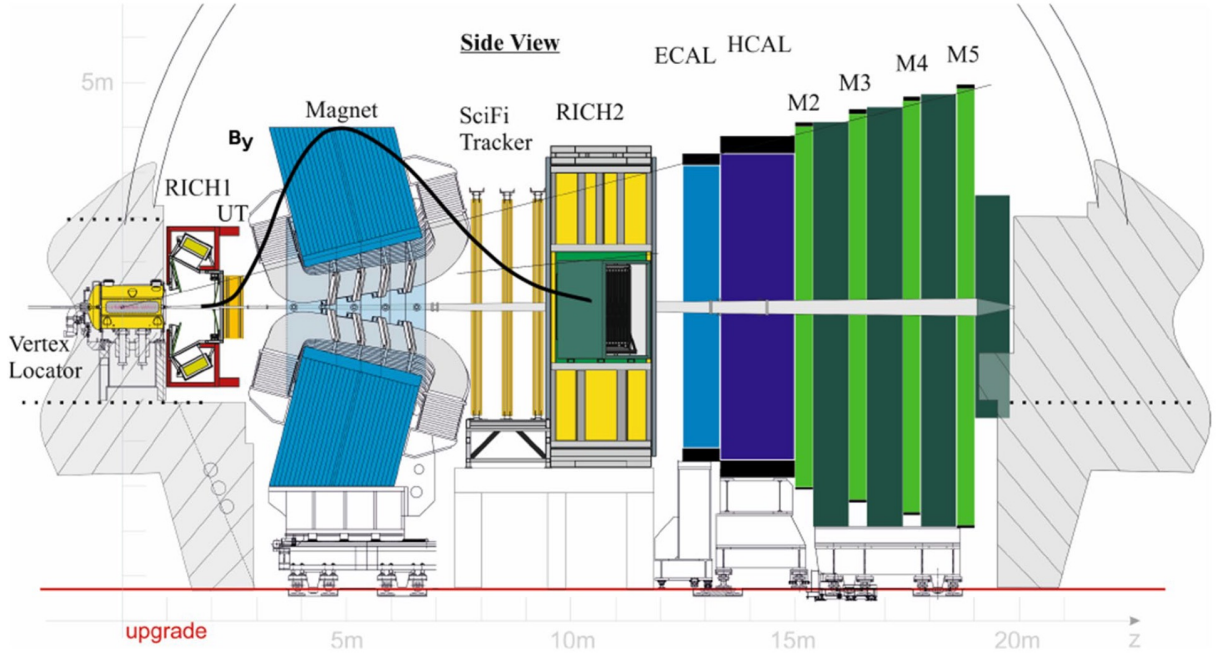
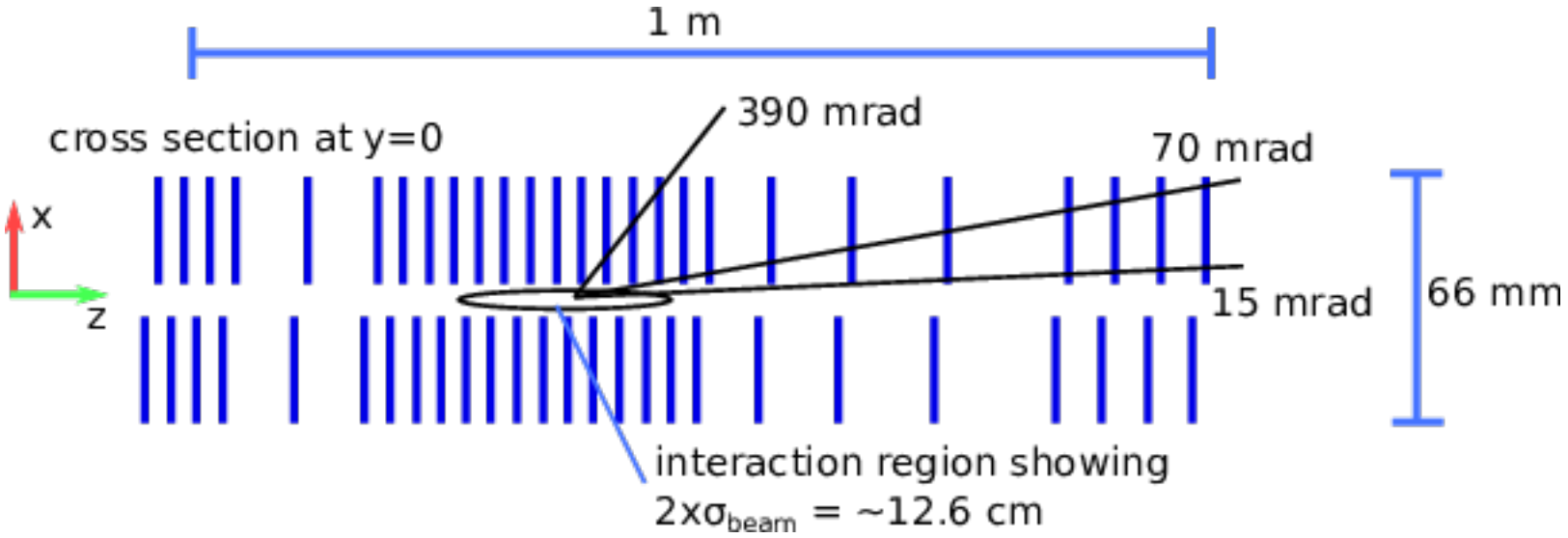


Fig. 4 Upgraded LHCb detector. The y-component of the magnetic field B_y is overlaid to visualize in which parts of the detector trajectories are bent. The maximum B_y value is 1.05 T

– LHCb raw events have an average size of 100 kB. When copying raw data to the GPU, the PCIe connection between the CPU and the GPU poses no limitation to the system, even when several thousand events are processed in parallel.

Simple detectors, simple algorithms



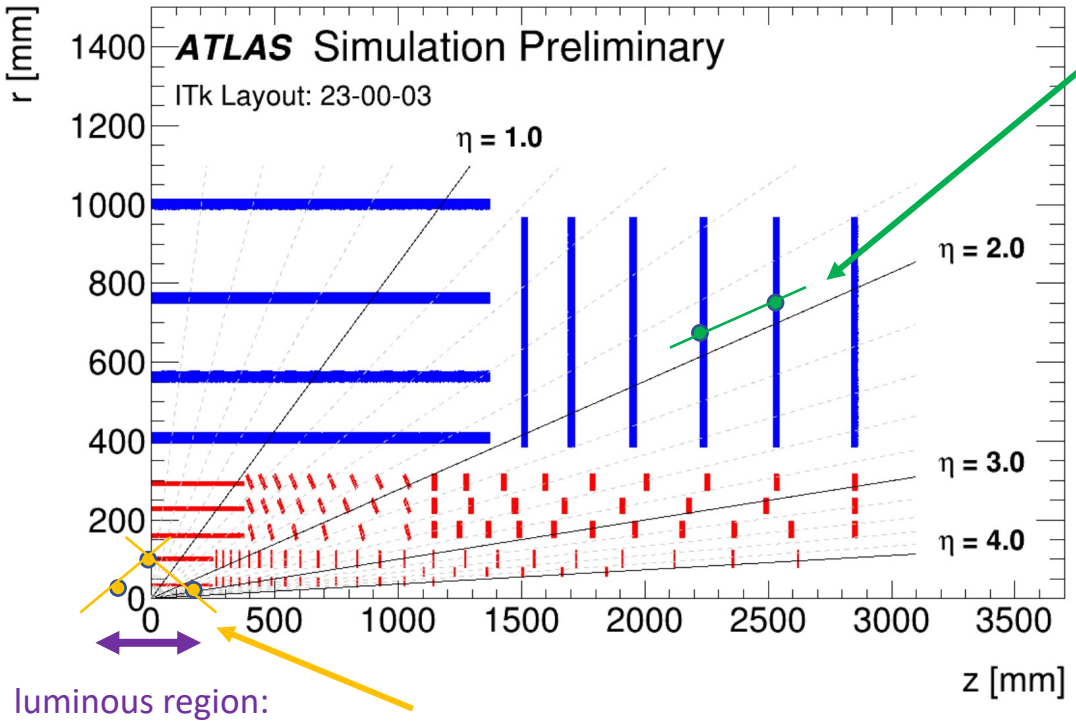
Velo Detector

The Velo detector consists of 26 planes of silicon pixel sensors placed around the interaction region. Its main purpose lies in reconstructing the *pp* collisions (primary vertices or PVs) and in creating seed tracks to be further propagated through the other LHCb detectors. The Velo track reconstruction is fully described in an earlier publication [17] and is recapped here for convenience.

The reconstruction begins by grouping measurements caused by the passage of a particle within each silicon plane into clusters, an example of a more general process known as connected component labeling. Allen uses a clustering algorithm employing bit masks, which searches for clusters locally in small regions. Every region can be treated independently, allowing for parallel processing.

Straight-line tracks are reconstructed by first forming seeds of three hits from consecutive layers ("triplets"), and then extending these to the other layers in parallel. We exploit the fact that prompt particles produced in *pp* collisions traverse the detector in lines of constant ϕ angle (within a cylindrical coordinate system where the cylinder axis coincides with the LHC beamline) and sort hits on every layer by ϕ for fast look-up when combining hits to tracks.

On fancy detectors



In this region, it is relatively clear in which direction to look for the next hit.

luminous region:
-200 < z < 200
at r = 0

In this region, the direction is less clear.
In addition, this is where the density of hits is largest.

Graph Neural Networks (GNN)

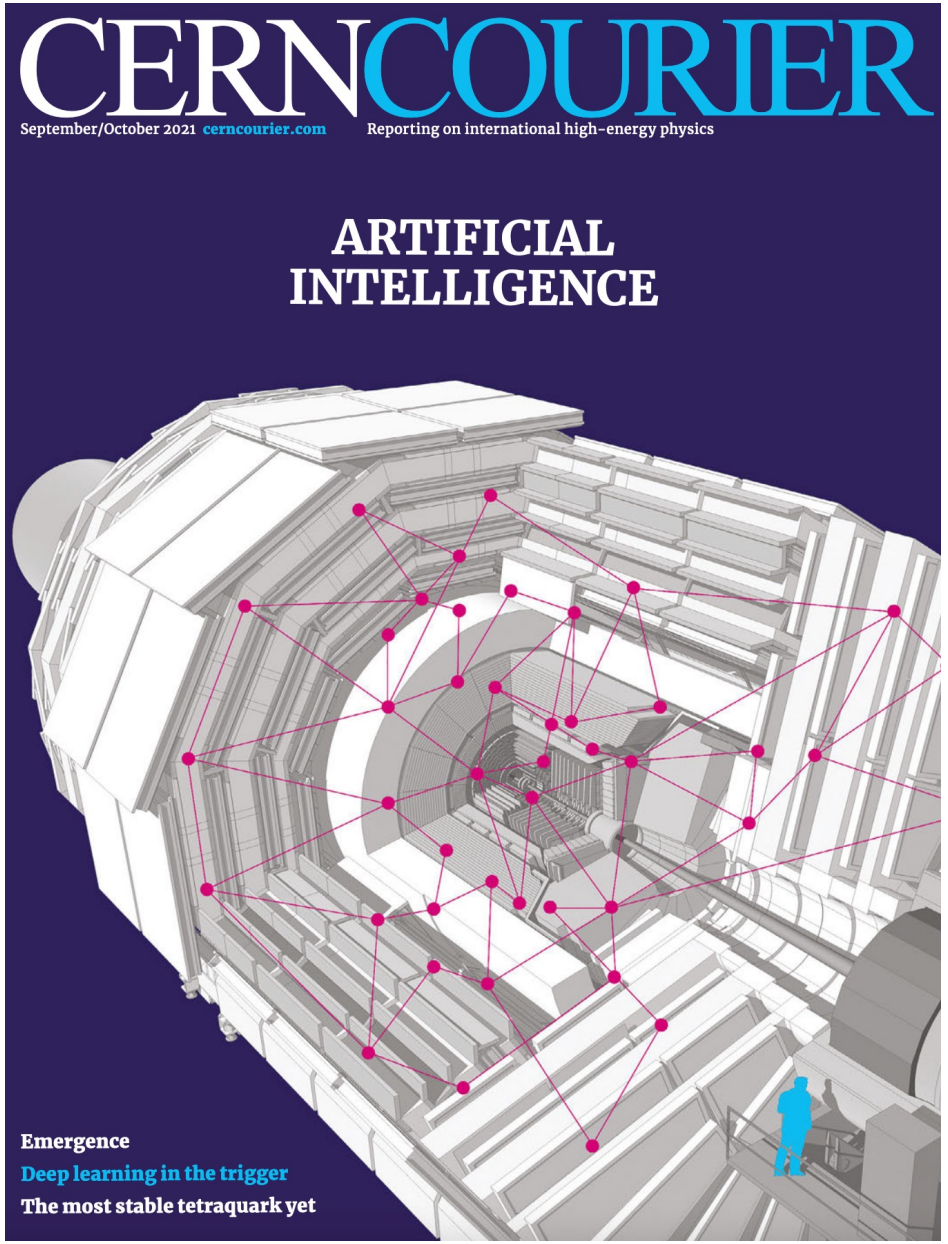
Data from HEP detectors in general,
and tracking detectors in particular, are sparse.

ATLAS ITk tracker at HL-LHC:
- 9 billion channels
- “only” 300k hits in one given event

The detectors are inhomogeneous
(combine different technologies)
and have complex geometry.

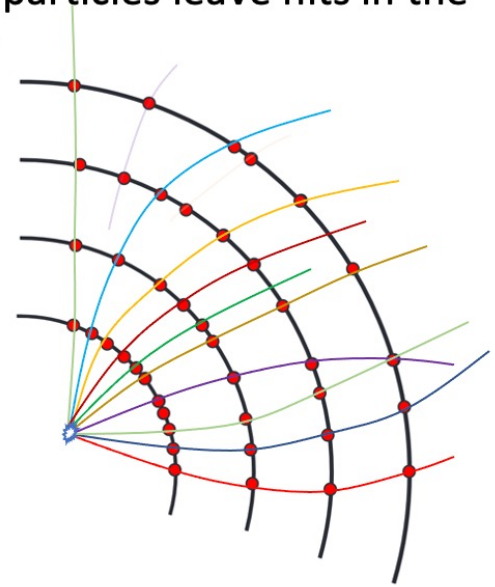
Such data are hard to represent as images.

Graphs are a natural tools to represent such data.
GNNs are neural networks that operate on
graphs of any topology and complexity.

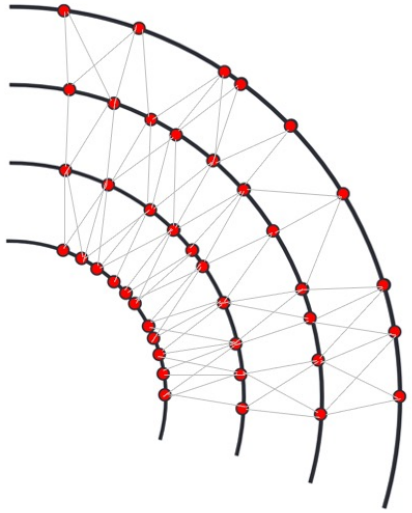


Tracking based on GNNs

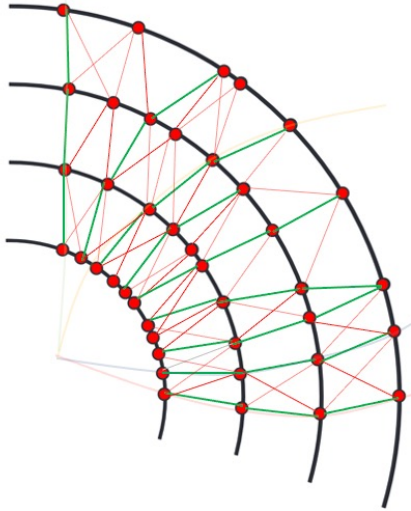
Charged particles leave hits in the detector



Represent the data using a graph



Goal:
classify the edges of the graph



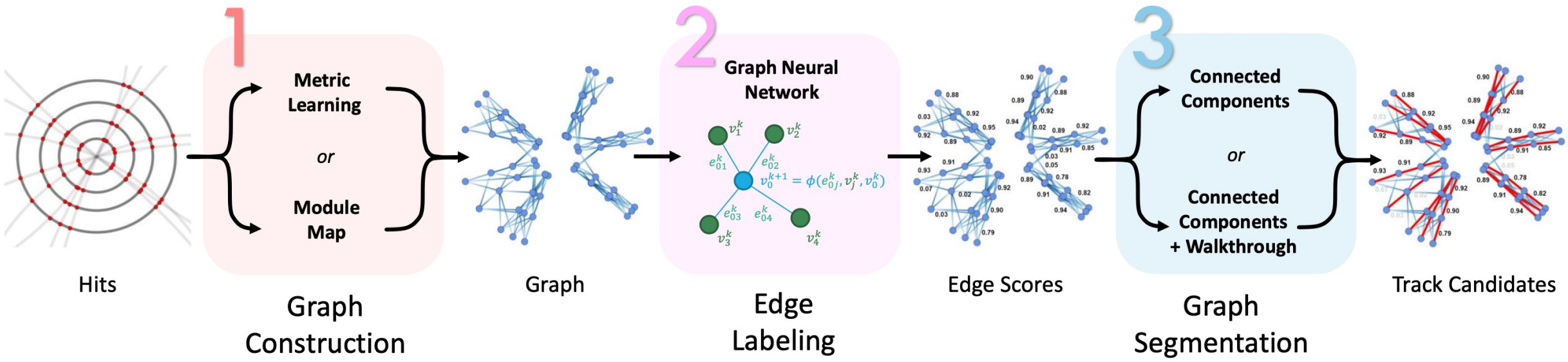
High classification score
=> **high probability** that the edge is part of a track

Low classification score
=> **low probability** that the edge is part of a track

One node of the graph = one hit in the detector

Connect two nodes using an edge
if “it seems possible” that the two hits
are two (consecutive) hits on a track

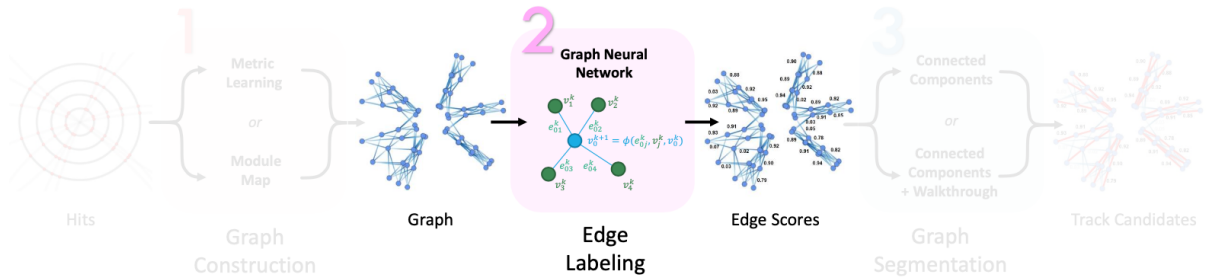
Tracking based on GNNs



Slide from Heberth at CTD2023

GNN4ITk

Edge labeling with GNN

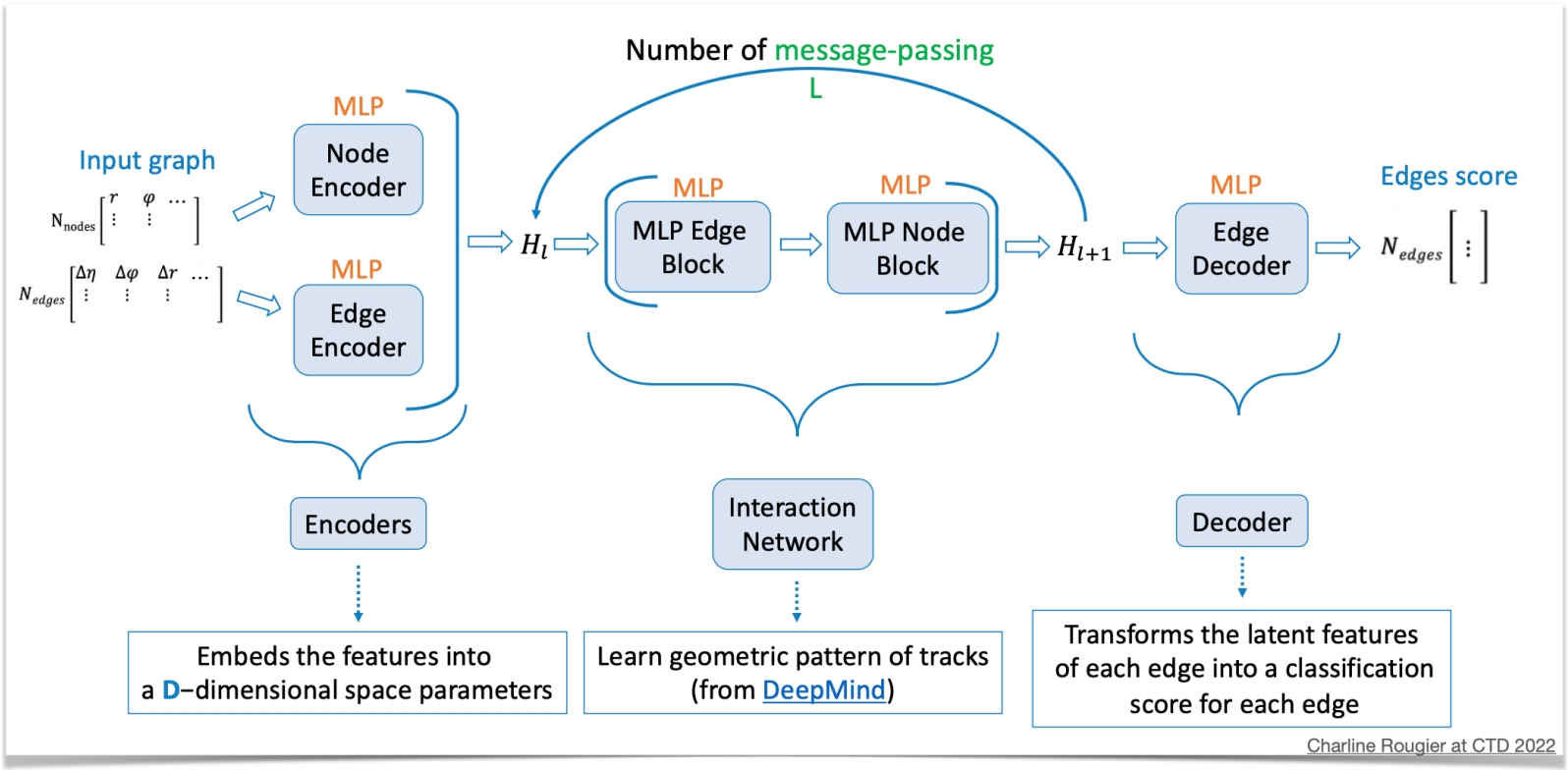


GNN config:

- 2 layers per MLP
- 128D latent space
- 8 message-passing

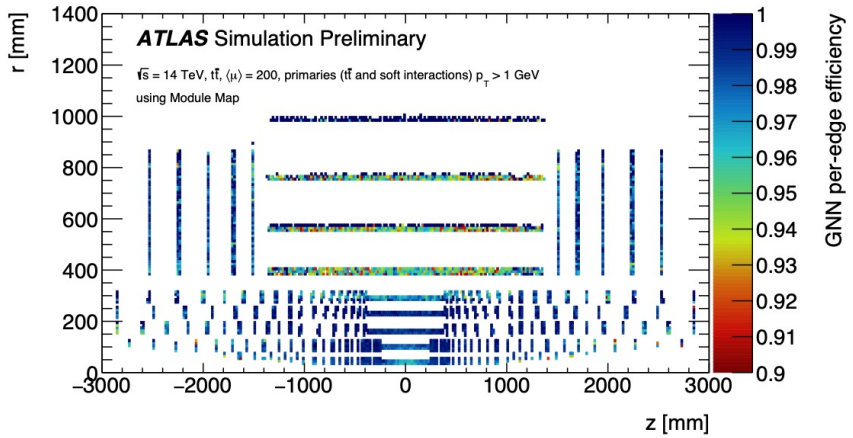
New w.r.t. CTD 2022:

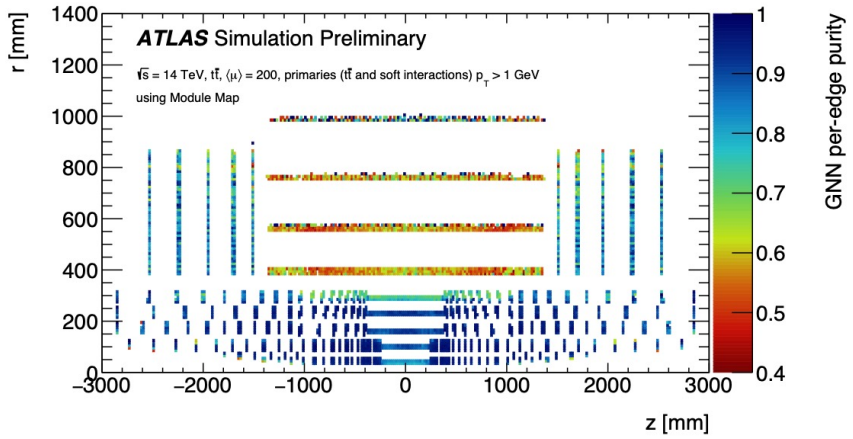
- Non-recurrent interaction network
- Doing batch norm
- Heterogeneous data



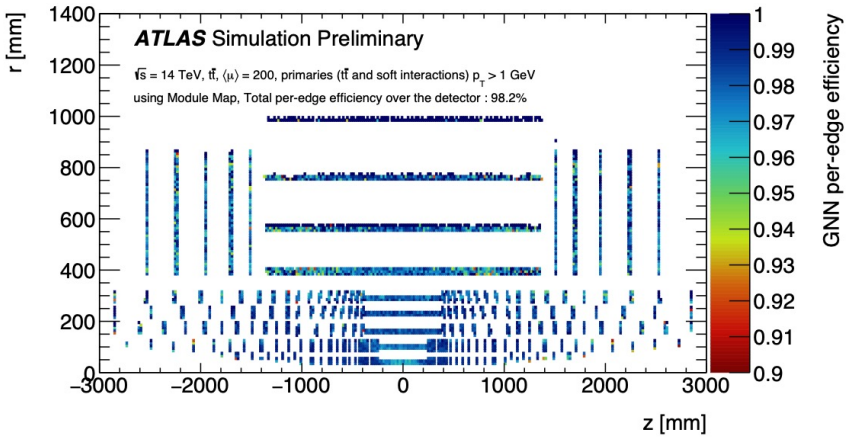
Efficiency and purity vs (r, z)

CTD2022



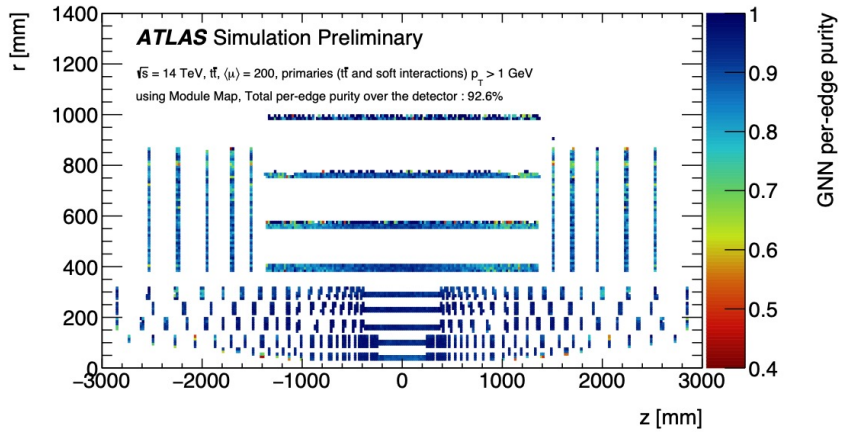


Heterogeneous Data + Extended GNN



$$efficiency = \frac{n_{true>treshold}}{n_{true}}$$

- Global efficiency of ~98%
- Efficiency more uniform in BARREL region



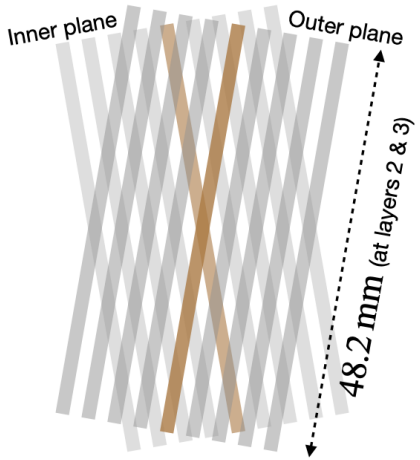
$$purity = \frac{n_{true>treshold}}{n_{true>treshold} + n_{fake>treshold}}$$

- **Significant improvement of purity in the STRIP BARREL region from ~40% to ~80%**
- Global purity of ~95%

Slide from Heberth at CTD2023

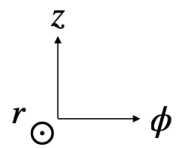
GNN4ITk

Edge labeling with GNN



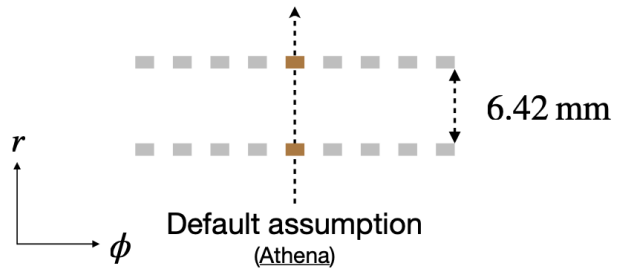
Double strip sensor planes in barrel module
Two strips fired by a particle in brown
Where did the particle hit the inner plane?

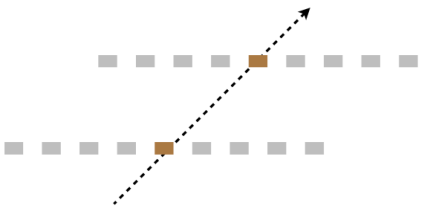
- Strip hits

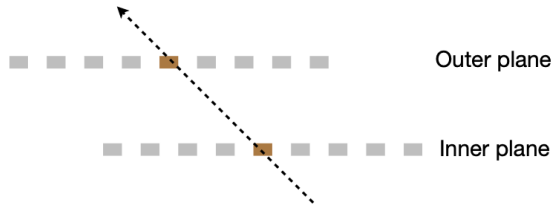


Default hits:
Poor σ_z ~ 1-3 cm
(was limiting GNN performance)

Some possible options:







- Problem addressed by passing info of the two individual strip clusters to the GNN; node features:
  - Strip barrel: $r_{\text{hit}}, \ldots, r_{\text{cl1}}, \ldots, r_{\text{cl2}}, \ldots$
  - Pixel: $r_{\text{hit}}, \ldots, r_{\text{hit}}, \ldots, r_{\text{hit}}, \ldots$

  (**Heterogeneous data** format)

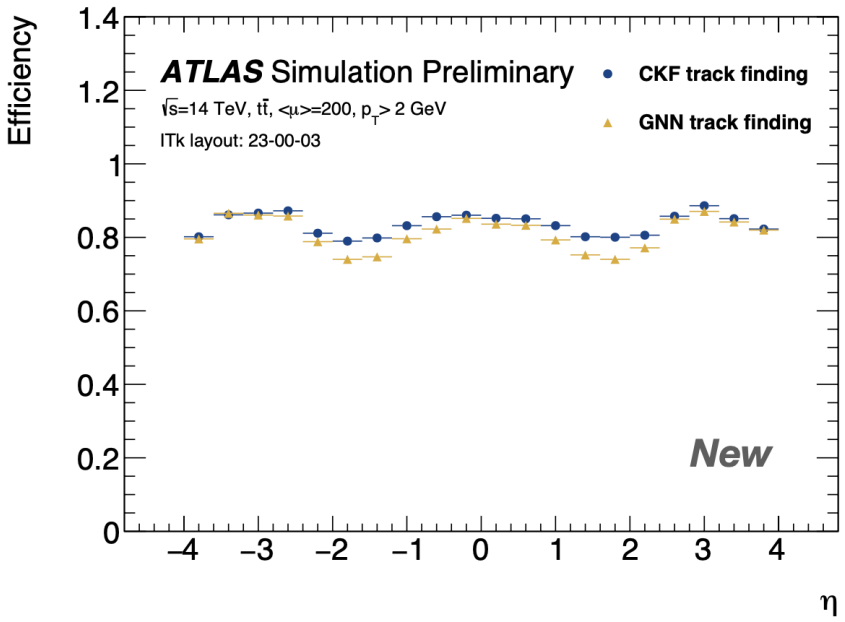
- Other alternatives under study: hand-engineered edge features based on hit pair info, & heterogeneous GNN model

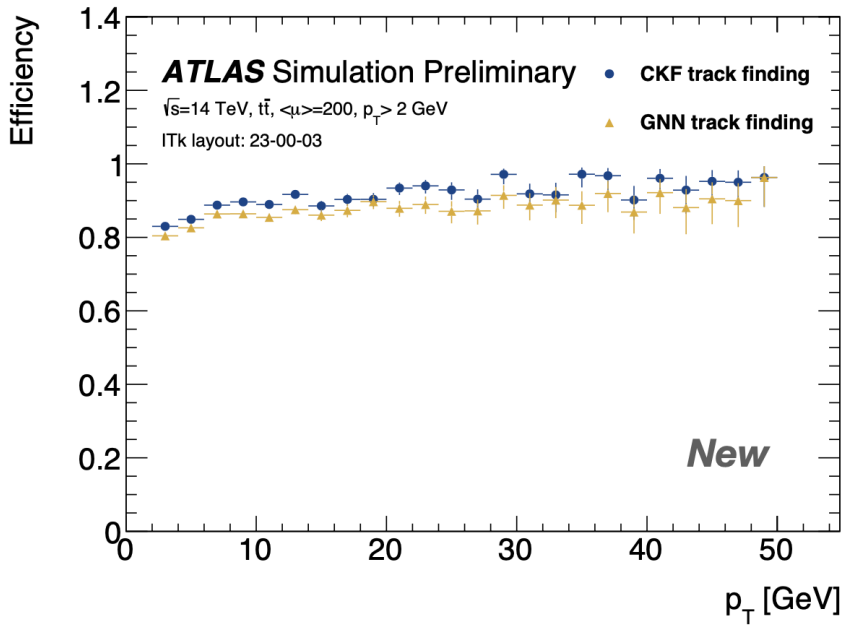
Slide from
Heberth
at CTD2023

Tracking efficiency

- Competitive “physics” efficiency
  (excluding electrons)

[O(10^{-3}) fake tracks:
Track candidates not matched to any particle]

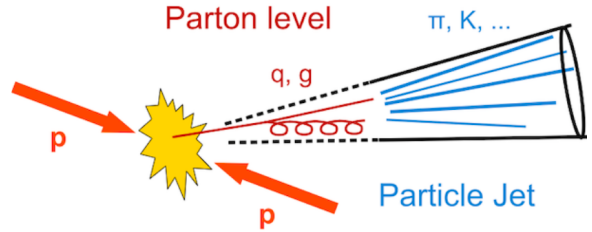




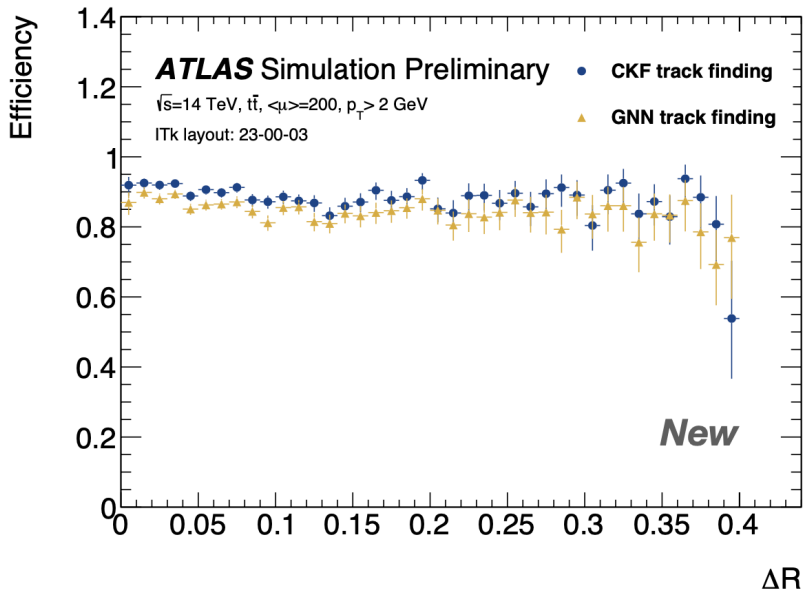
Slide from
Heberth
at CTD2023

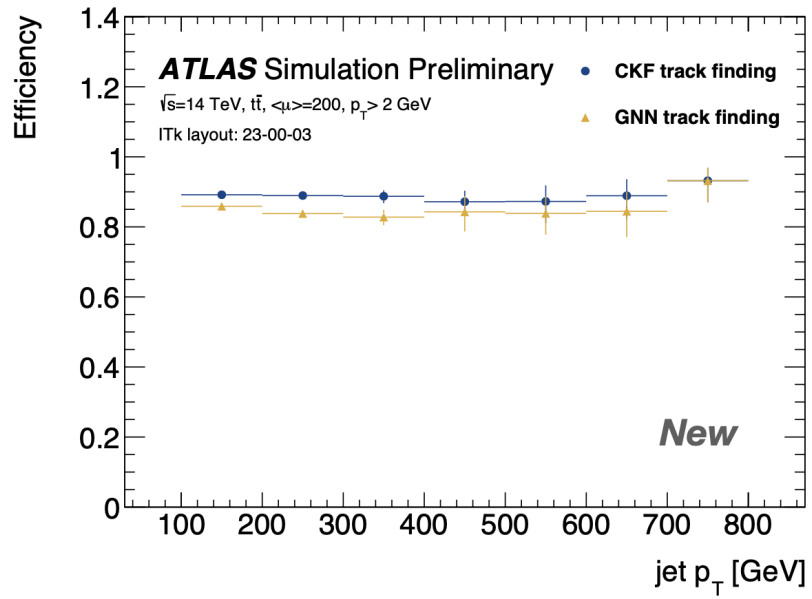
Tracking efficiency

Tracking inside jets



- Competitive “physics” efficiency even in dense environment
  (excluding electrons)

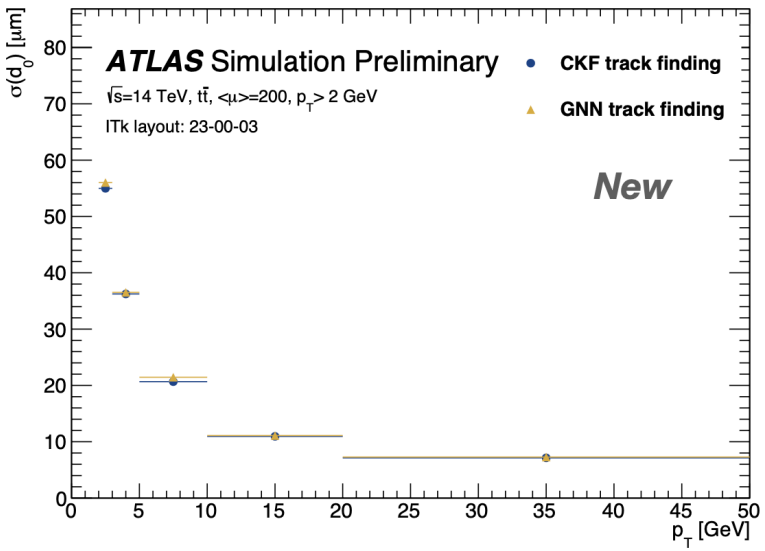


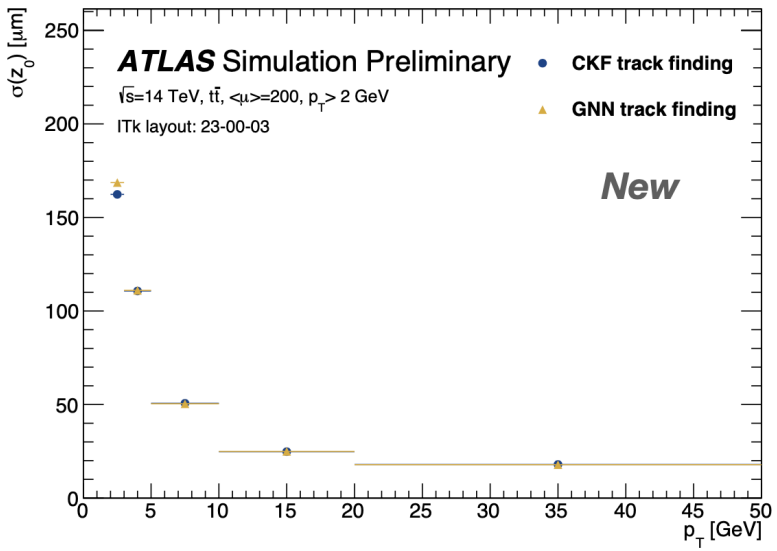


Slide from
Heberth
at CTD2023

Impact parameter resolution

- Given the good pixel hit content, good impact parameter resolution





Connecting The Dots 2023

10-13 October

8th International CTD workshop
Université Paul Sabatier, Toulouse, France

https://indico.cern.ch/e/CTD2023
ctd2023-loc@l2it.in2p3.fr

satellite event on Real time Tracking:
triggering events with tracks (October 13th)

Local Organizing Committee

Catherine Biscarat (L2IT)
Sylvain Caillou (L2IT)
Jocelyne Gauthier (L2IT)
Jan Stark (L2IT)
Jeanette Thibaut (L2IT)
Alexis Vallier (L2IT) - chair

International Advisory Committee

Alberto Annovi (INFN Pisa)
Paolo Calafiura (LBNL)
Giuseppe Cerati (FNAL)
Michel De Cian (EPFL)
Matthias Danninger (SFU)
Markus Elsing (CERN)
Frank Gaede (DESY)
Jose E. Garcia (IFIC Valencia)
Maurice Garcia-Sciveres (LBNL)
Vladimir Gligorov (LPNHE)
Heather Gray (UC Berkeley/LBNL)
Phil Harris (MIT)
David Lange (Princeton)
Salvador Marti (IFIC Valencia)
Fabrizio Palla (INFN Pisa)
David Rousseau (IJCLab)
Andi Salzburger (CERN)
Louise Skinnari (Northeastern U.)

cnrs Institut national de physique nucléaire et de physique des particules
Université de Toulouse
UNIVERSITÉ TOULOUSE III PAUL SABATIER
L2IT Laboratoire des 2 Infinis Toulouse

Talk at CTD2023 from LHCb/LPNHE

GNN-based pipeline for track finding in the Velo at LHCb

Allen: current HLT1, classical algorithms on GPU

Ext4velo: ML algorithm (GNN-based) on GPU

3. Track-Finding Performance

39

Category	Metric	Allen	$s_{\text{triplet}} > 0.32$ Etx4velo $d^2_{\text{max}} = 0.010$	$s_{\text{triplet}} > 0.36$ Etx4velo $d^2_{\text{max}} = 0.020$
Long, no electrons ✓ In acceptance ✓ Reconstructible in the velo ✓ Reconstructible in the SciFi ✓ Not an electron	Efficiency	99.26%	99.28%	99.51%
	Clone rate	2.54%	0.96%	0.89%
	Hit efficiency	96.46%	98.73%	98.90%
	Hit Purity	99.78%	99.94%	99.94%
Long electrons ✓ In acceptance ✓ Reconstructible in the velo ✓ Reconstructible in the SciFi ✓ Electron	Efficiency	97.11%	98.80%	99.22%
	Clone rate	4,25%	7.42%	7.31%
	Hit efficiency	95.24%	96.54%	96.79%
	Hit purity	97.11%	98.46%	98.46%
Long, from strange ✓ In acceptance ✓ Reconstructible in the velo ✓ Decays from a strange *Good proxy for displaced tracks*	Efficiency	97.69%	97.50%	98.06%
	Clone rate	2.50%	0.92%	0.81%
	Hit efficiency	97.69%	98.22%	98.77%
	Hit purity	99.34%	99.68%	99.68%
X	Ghost rate	2.18%	0.76%	0.81%

- Evaluation with 5,000 events
- **Track matched to a particle** if at least 70% of its hits belong to this particle
- Allen algorithm described in arXiv:2207.03936v2
- 2 different GNN trainings for $d^2_{\text{max}} = 0.010$ and $d^2_{\text{max}} = 0.020$

Long categories

Worse — Better

(link to talk)

Tracking based on GNNs

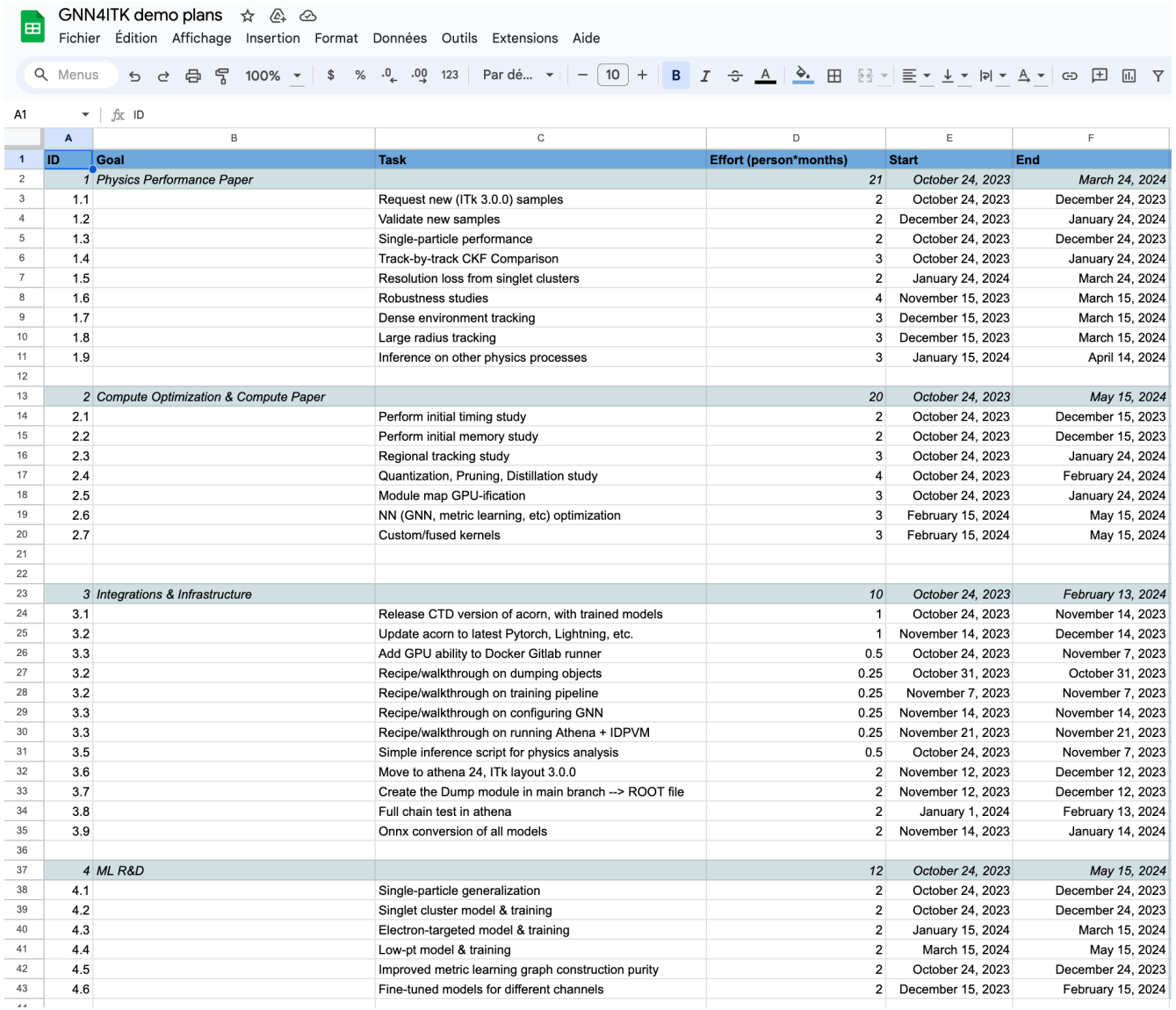
Recently detailed our schedule toward the TDR in Q3 of 2024.

We have the physics performance that we need for a demonstrator.

Moving a lot of focus on implementation aspects.

- full chain (from clusters to fitted tracks) on GPU, without any intermediate transfers to/from the host
- use “GPU EDM” and track fit from ACTS/traccc

This domain (implementation) is new for many in our field (in ATLAS and elsewhere). It is crucial that we coordinate our activities with the LHCb colleagues at LPNHE, our ACTS/ATLAS colleagues at IJCLab and the Reprises project.


GNN4ITK demo plans

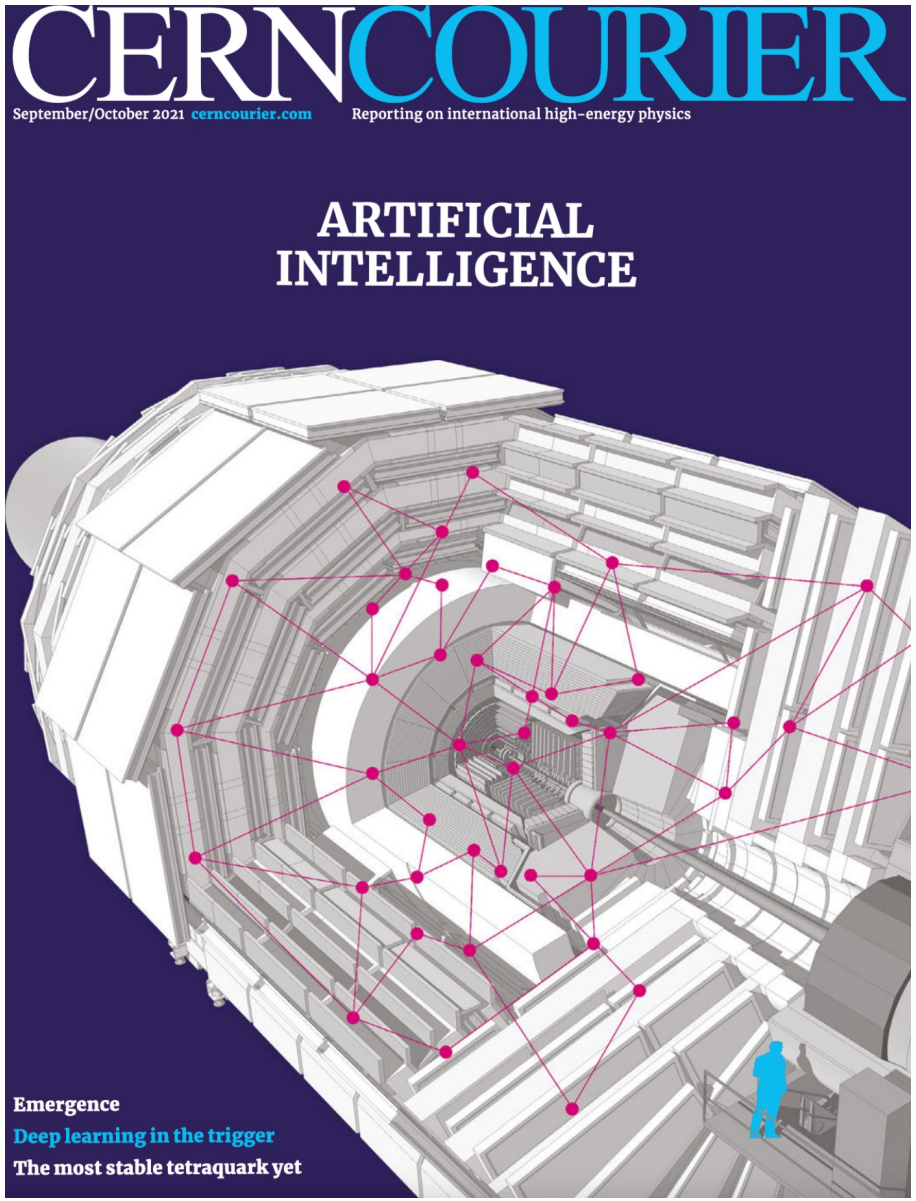
Fichier Édition Affichage Insertion Format Données Outils Extensions Aide

A1 | fx ID

ID	Goal	Task	Effort (person*months)	Start	End
1	*Physics Performance Paper*		*21*	*October 24, 2023*	*March 24, 2024*
1.1		Request new (ITk 3.0.0) samples	2	October 24, 2023	December 24, 2023
1.2		Validate new samples	2	December 24, 2023	January 24, 2024
1.3		Single-particle performance	2	October 24, 2023	December 24, 2023
1.4		Track-by-track CKF Comparison	3	October 24, 2023	January 24, 2024
1.5		Resolution loss from singlet clusters	2	January 24, 2024	March 24, 2024
1.6		Robustness studies	4	November 15, 2023	March 15, 2024
1.7		Dense environment tracking	3	December 15, 2023	March 15, 2024
1.8		Large radius tracking	3	December 15, 2023	March 15, 2024
1.9		Inference on other physics processes	3	January 15, 2024	April 14, 2024
2	*Compute Optimization & Compute Paper*		*20*	*October 24, 2023*	*May 15, 2024*
2.1		Perform initial timing study	2	October 24, 2023	December 15, 2023
2.2		Perform initial memory study	2	October 24, 2023	December 15, 2023
2.3		Regional tracking study	3	October 24, 2023	January 24, 2024
2.4		Quantization, Pruning, Distillation study	4	October 24, 2023	February 24, 2024
2.5		Module map GPU-ification	3	October 24, 2023	January 24, 2024
2.6		NN (GNN, metric learning, etc) optimization	3	February 15, 2024	May 15, 2024
2.7		Custom/fused kernels	3	February 15, 2024	May 15, 2024
3	*Integrations & Infrastructure*		*10*	*October 24, 2023*	*February 13, 2024*
3.1		Release CTD version of acorn, with trained models	1	October 24, 2023	November 14, 2023
3.2		Update acorn to latest Pytorch, Lightning, etc.	1	November 14, 2023	December 14, 2023
3.3		Add GPU ability to Docker Gitlab runner	0.5	October 24, 2023	November 7, 2023
3.2		Recipe/walkthrough on dumping objects	0.25	October 31, 2023	October 31, 2023
3.2		Recipe/walkthrough on training pipeline	0.25	November 7, 2023	November 7, 2023
3.3		Recipe/walkthrough on configuring GNN	0.25	November 14, 2023	November 14, 2023
3.3		Recipe/walkthrough on running Athena + IDPVM	0.25	November 21, 2023	November 21, 2023
3.5		Simple inference script for physics analysis	0.5	October 24, 2023	November 7, 2023
3.6		Move to athena 24, ITk layout 3.0.0	2	November 12, 2023	December 12, 2023
3.7		Create the Dump module in main branch --> ROOT file	2	November 12, 2023	December 12, 2023
3.8		Full chain test in athena	2	January 1, 2024	February 13, 2024
3.9		Onnx conversion of all models	2	November 14, 2023	January 14, 2024
4	*ML R&D*		*12*	*October 24, 2023*	*May 15, 2024*
4.1		Single-particle generalization	2	October 24, 2023	December 24, 2023
4.2		Singlet cluster model & training	2	October 24, 2023	December 24, 2023
4.3		Electron-targeted model & training	2	January 15, 2024	March 15, 2024
4.4		Low-pt model & training	2	March 15, 2024	May 15, 2024
4.5		Improved metric learning graph construction purity	2	October 24, 2023	December 24, 2023
4.6		Fine-tuned models for different channels	2	December 15, 2023	February 15, 2024

Additional material

AISSAI conference on heterogeneous data in Toulouse – spring 2024



All nodes are pink, regardless of their position in the detector (tracker, calorimeter, muon detectors) ! Not only in this illustration, this reflects what is typically done in practice.

Ideally, nodes in different subdetectors would represent different types of measurements in different subdetectors (3D hits in the tracking pixel detectors, 2D measurements in the tracking strip detectors, energy deposits in the calorimeters, ...) and they would be shown in different colours in illustrative figures as the one above.

Developments of models and techniques, initially driven by applications in particle physics, could accelerate developments in this domain.

Subtopics:

- Heterogeneous GNN architectures.
- The next big thing in geometric deep learning ? Modelling complex systems requires going beyond graphs.
- Green AI is an integral part of this. Heterogeneous architectures will likely be run on low-level reconstruction tasks like particle flow and track reconstruction, i.e. run on essentially every event. This is where the big potential gains in energy savings are.

The definition of the contours of the conference is curently being finalised

- Spring 2024
- ~80 participants
- At the same beautiful venue right in the city centre as CTD 2023
  (Le Village by CA and Flashback café, this has a start-up flair to it).
- With contributions from ANITI chairs
- Fix date before end of Nov. 2023