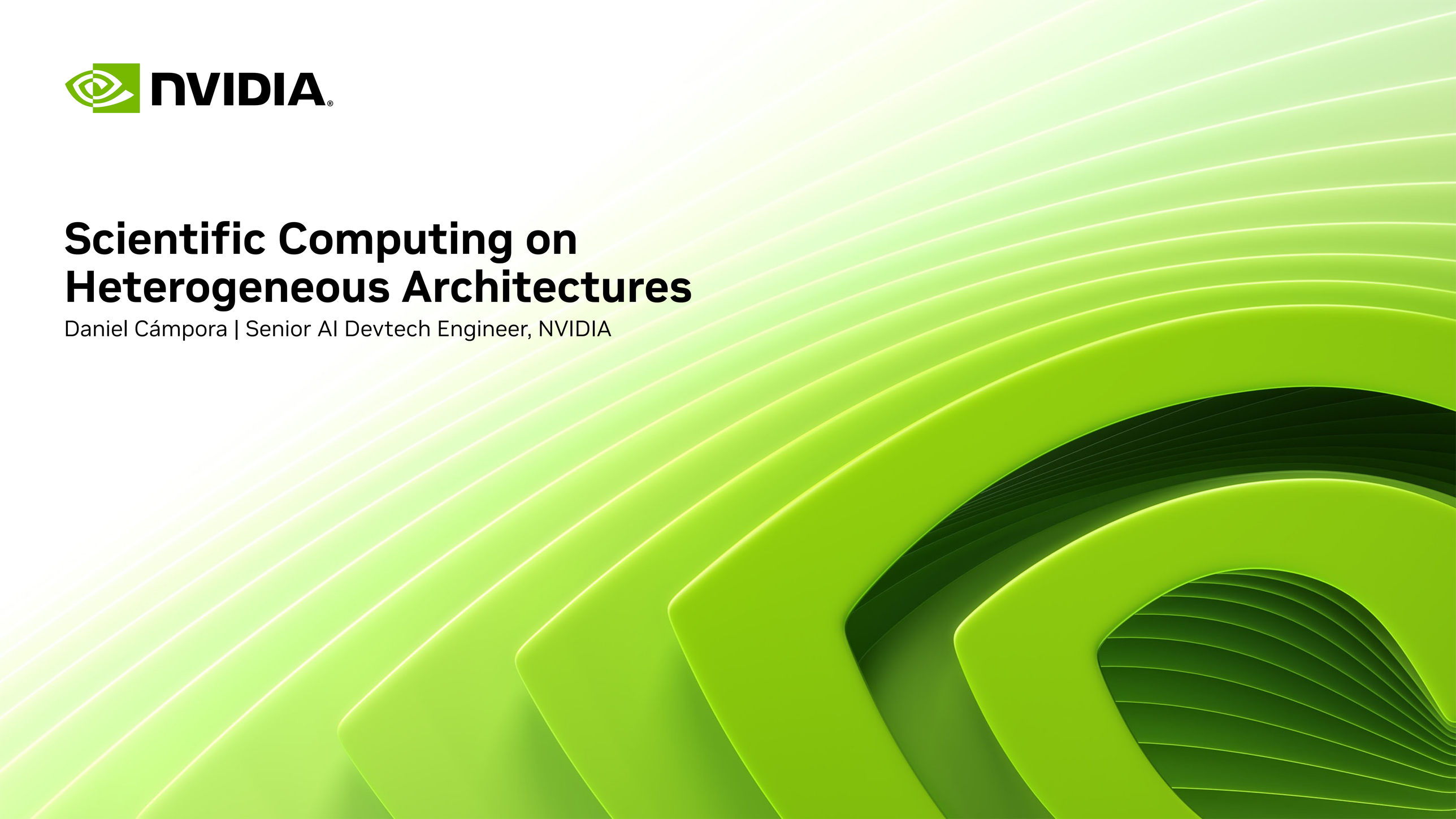


Scientific Computing on Heterogeneous Architectures

Daniel Cámpora | Senior AI Devtech Engineer, NVIDIA

$ whoami

- **CERN** (2010 – 2019)
  - Summer Student, Indico
  - Technical Student, ATLAS
  - Fellow, LHCb
  - Doctoral Student, LHCb
- **NIKHEF** (2020)
  - Postdoc, LHCb
- **Maastricht University** (2021 – 2023)
  - Assistant Professor (LHCb)
- **NVIDIA** (2023+)
  - Senior Devtech Engineer

- Particle physics reconstruction
  - Online
  - Reconstruction (decoding, tracking, etc.)
  - Framework
- AI inference
  - TensorRT-LLM

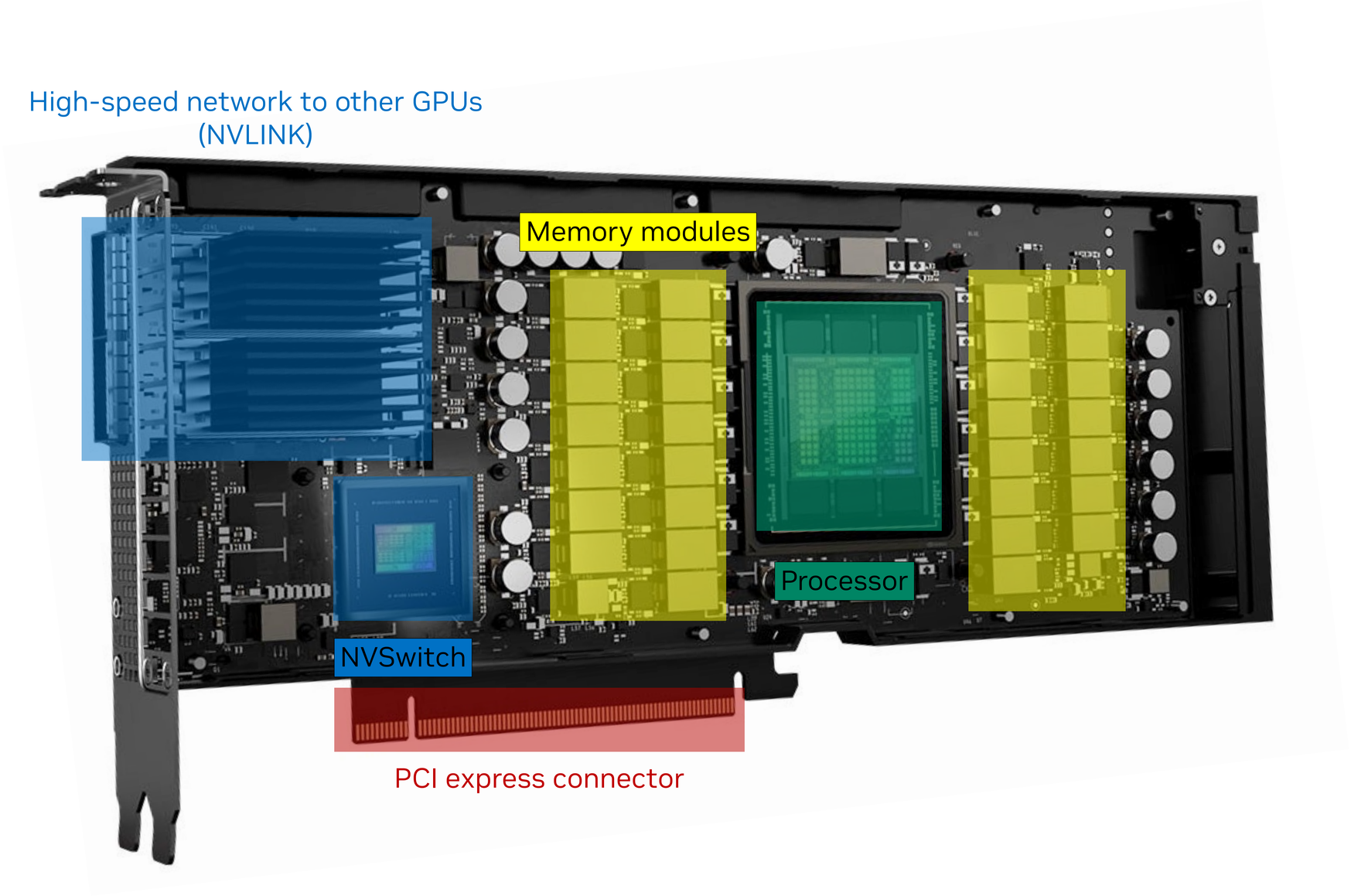
Table of Contents

GPUs: Parallel Processors

- GPUs are historically processors specialized to perform graphic-oriented workloads.

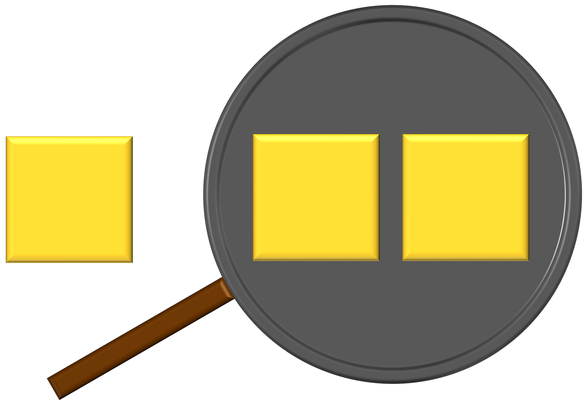
Dedicated GPU Card

Dedicated GPU Card – Detail

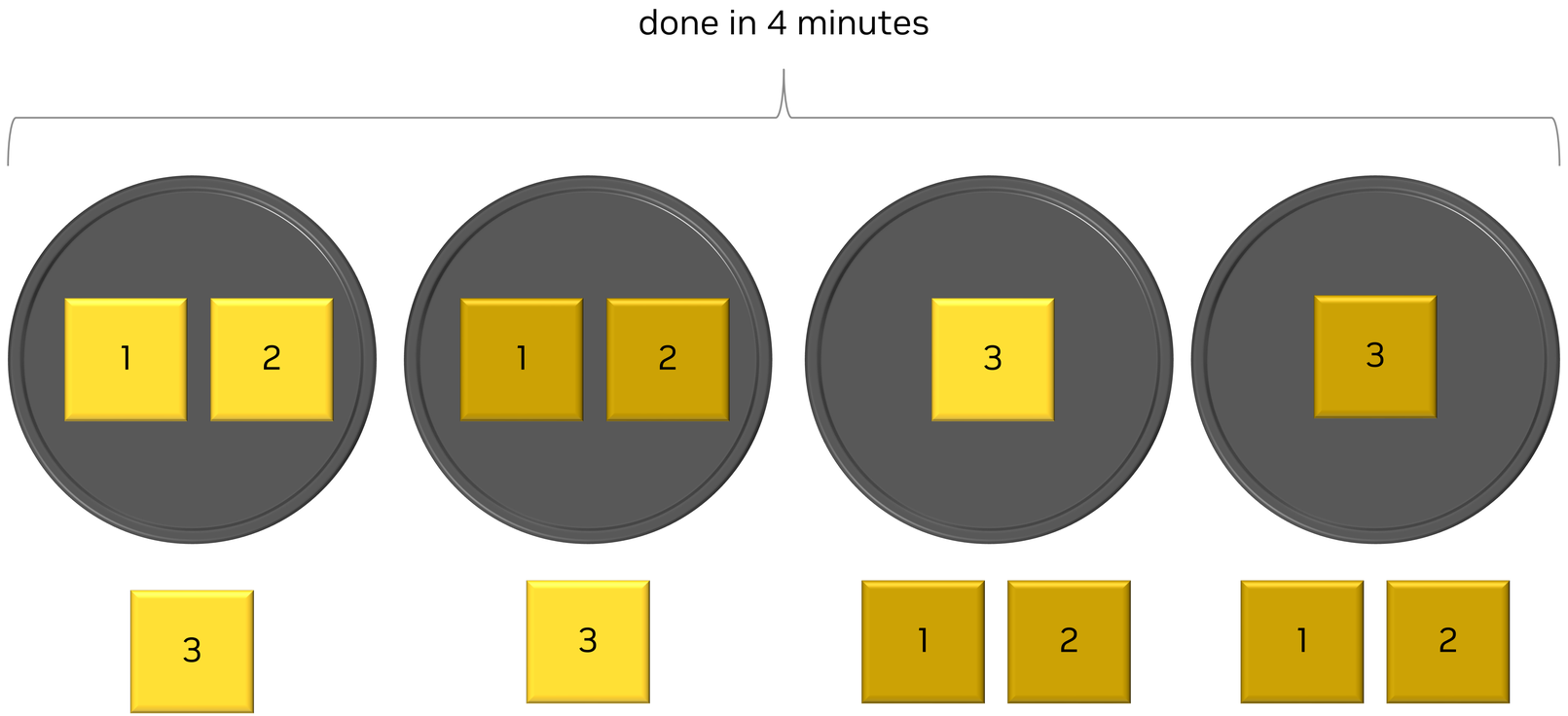


Making Toast Fast

https://www.youtube.com/watch?v=gVPK81rI390

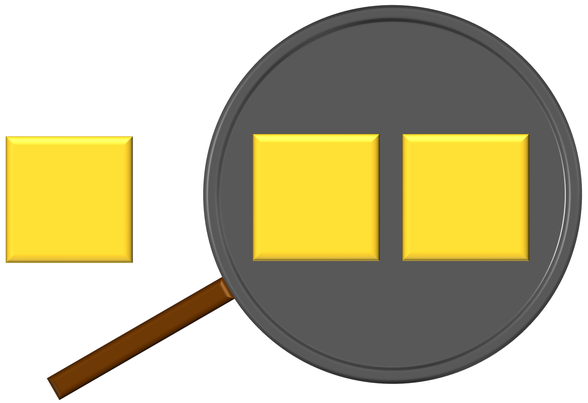


How long does it take to
fry 3 toast with a pan
if frying one side of
a toast takes 1 min?

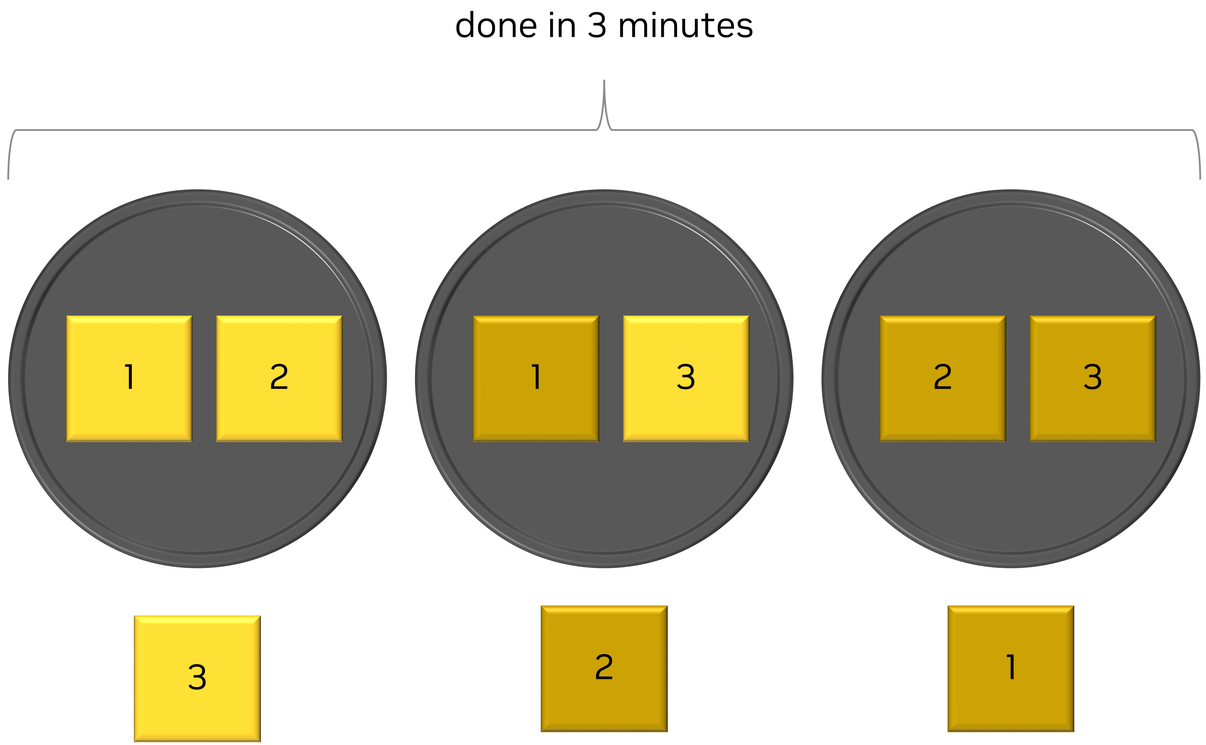


Making Toast Fast

https://www.youtube.com/watch?v=gVPK81rI390



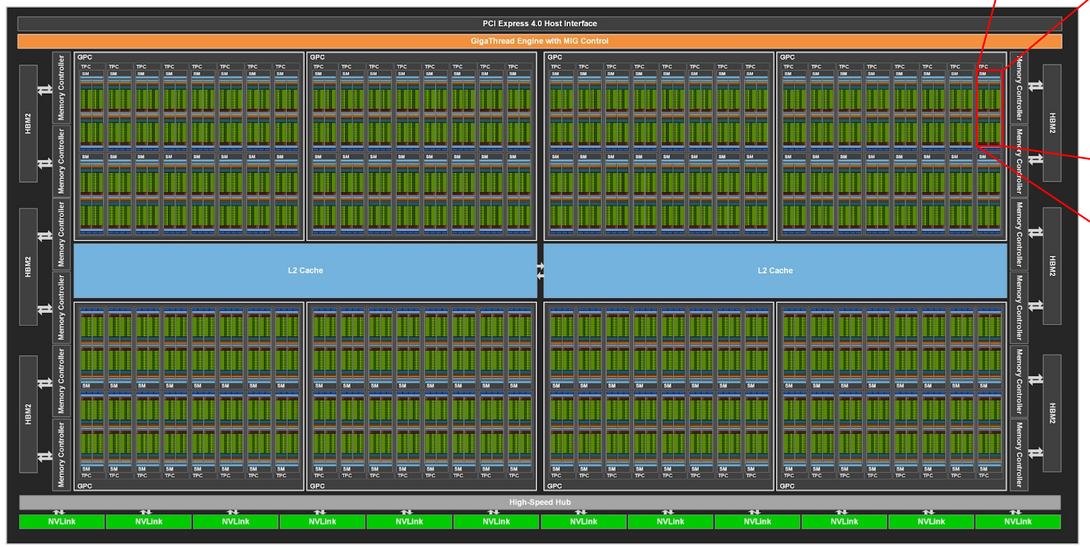
How long does it take to
fry 3 toast with a pan
if frying one side of
a toast takes 1 min?



CUDA Cores

How many pans does it have?

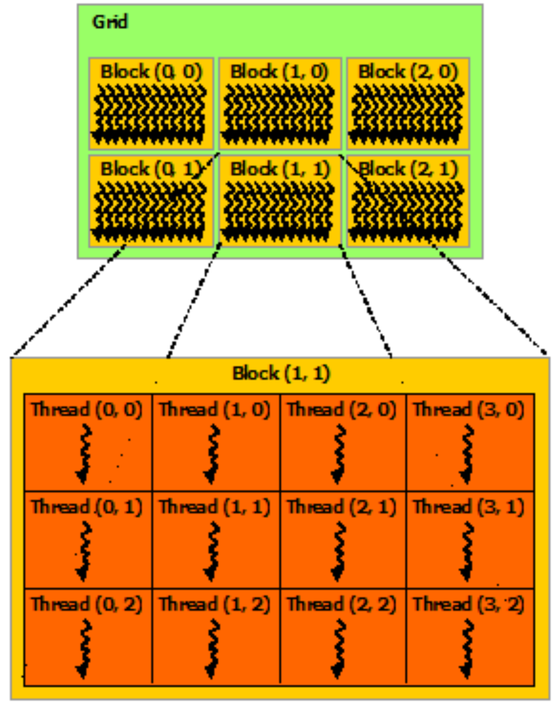
- A GPU is made up of Streaming Multiprocessors, 100s of them.
- Arithmetic takes up most of the processor space.
- Applications may use one or more SMs simultaneously.
- GPU applications have thousands of threads in flight.



Simplified Execution Model

How to operate the pans?

- N *Kernels* are executed in parallel by N different *CUDA threads.*
- Threads are arranged a one-dimensional, two-dimensional, or three-dimensional block of threads, called a *thread block.* A set of thread blocks are launched to execute a function.
- It is usually better the overcommit w.r.t. the number of threads to facilitate instruction latency (“prepare toast while other getting fried”).
- When a multiprocessor is given one or more thread blocks to execute, it partitions them into warps and each warp gets scheduled by a warp scheduler for execution.
- It is important to avoid *warp divergence* (“frying toasts with different cooking times in the same pan”) whenever possible!
- A set of thread blocks running concurrently is called a *wave*. The more waves, the better to minimize tail effects.



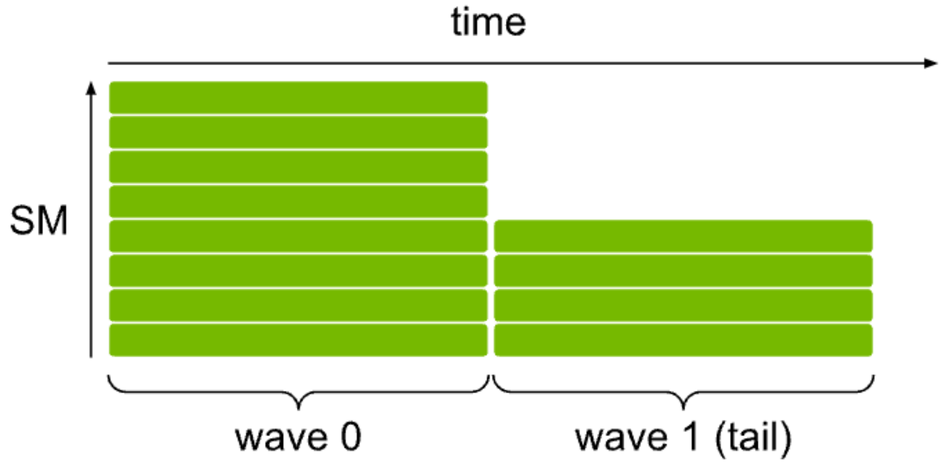


Table of Contents

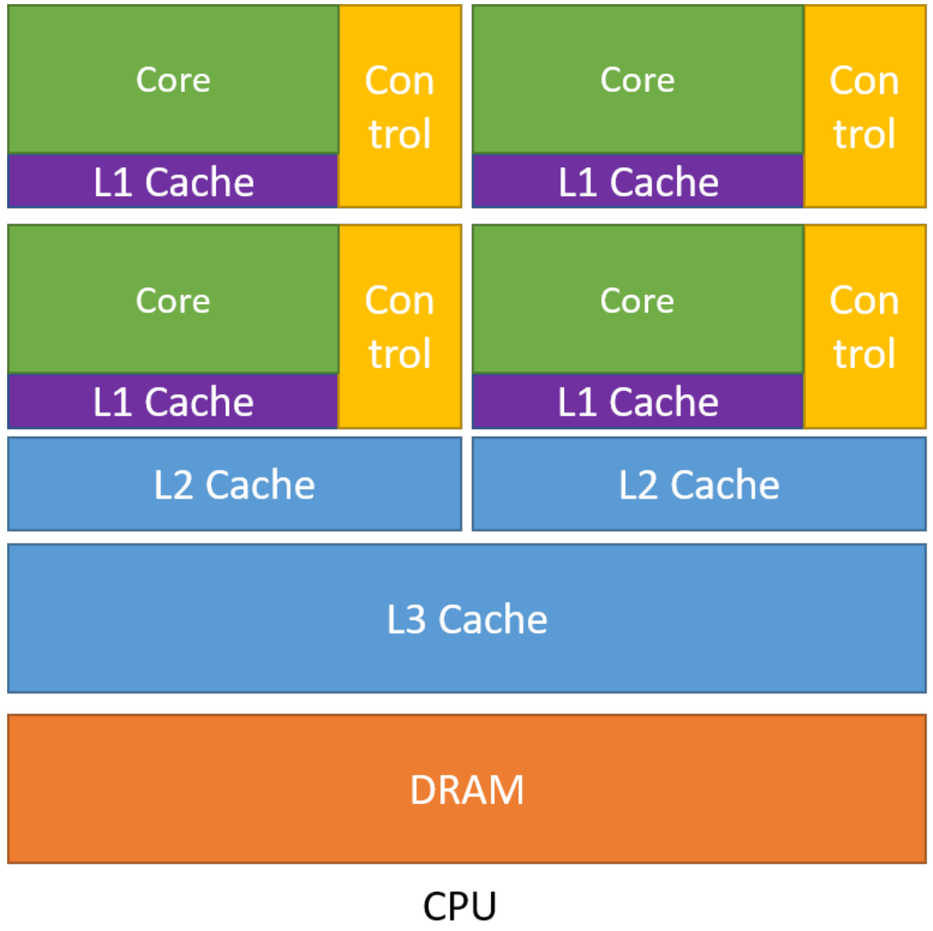
CPUs and GPUs are Designed Very Differently

- CPU – **Latency** oriented cores.
  - Finish quickly a series of instructions.

- GPU – **Throughput** oriented cores.
  - Perform as many instructions per second as possible.

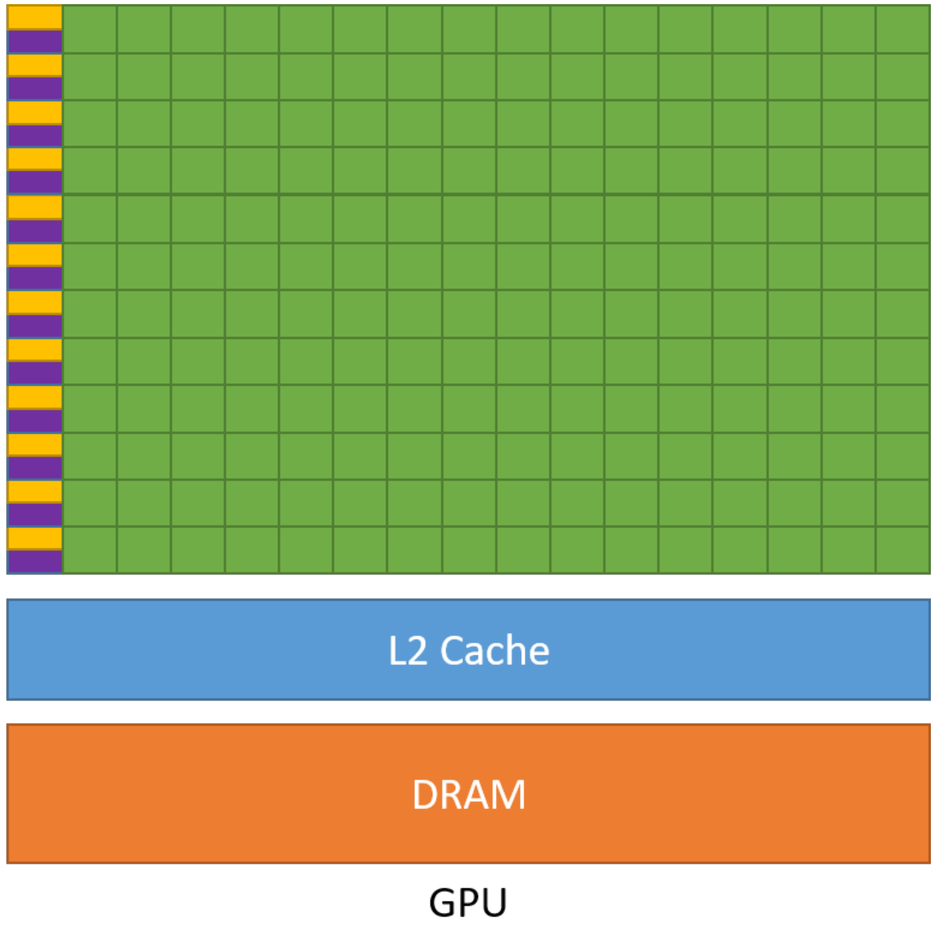
CPU Highlights

- Powerful ALU
  - Reduced operation latency
- Large caches
  - Convert long latency memory accesses to short latency cache accesses
- Sophisticated control
  - Branch prediction for reduced branch latency
  - Data forwarding for reduced data latency

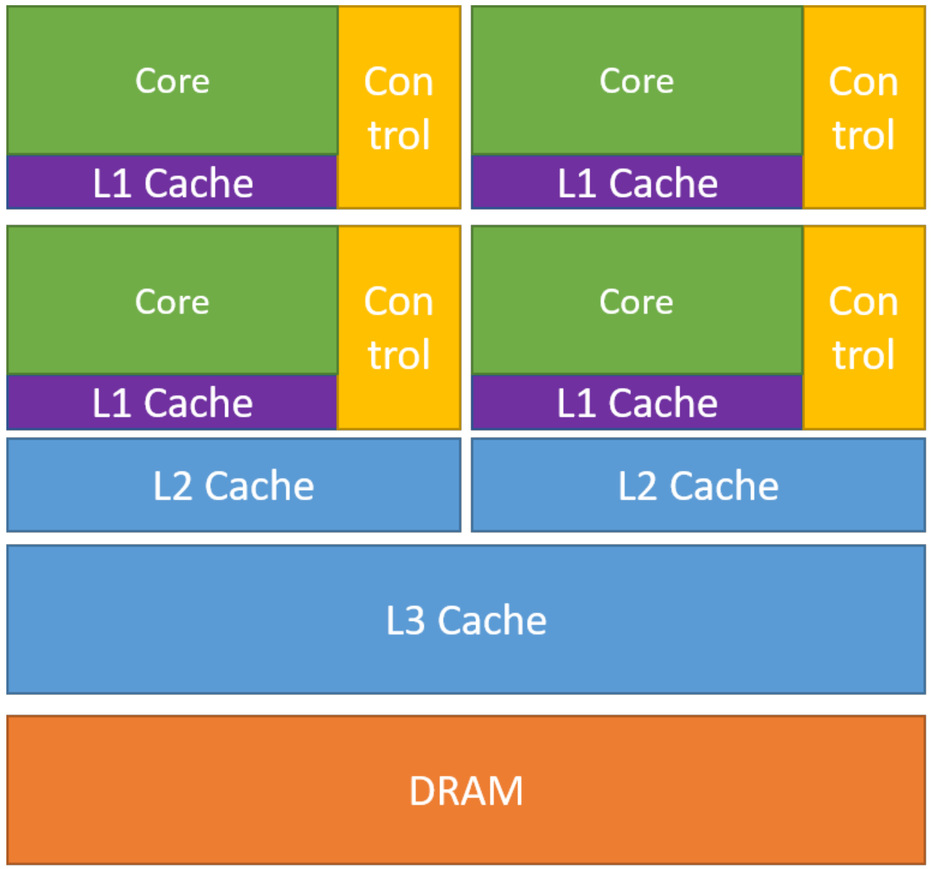


GPU Highlights

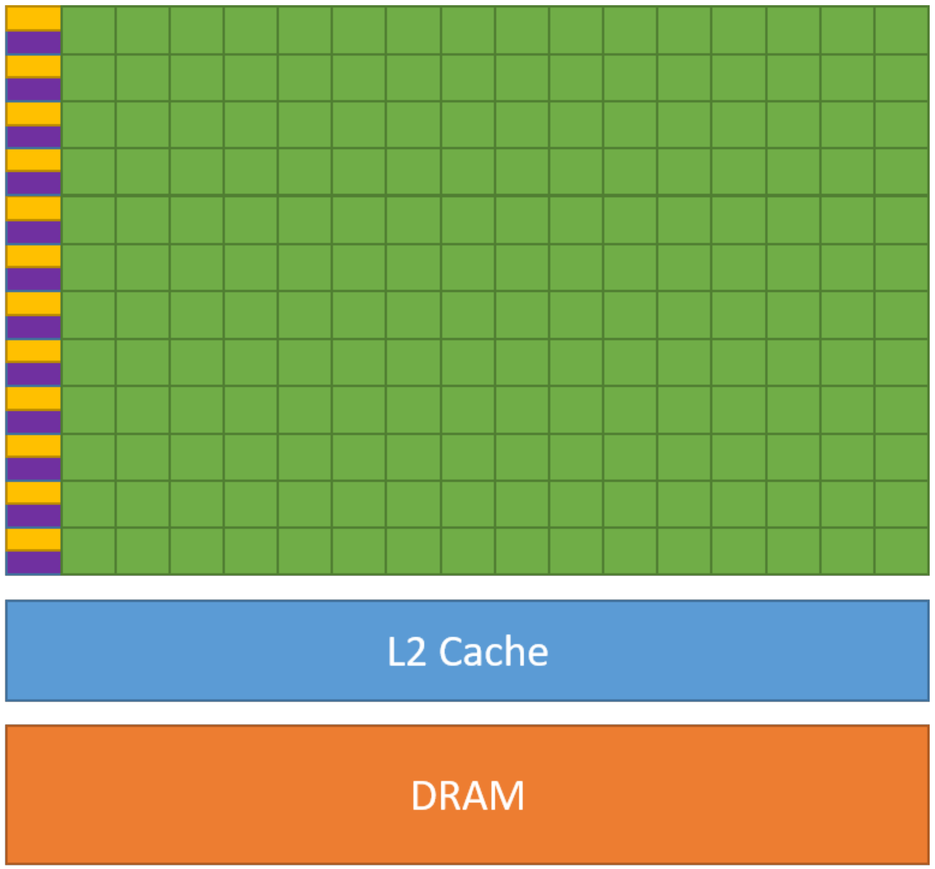
- Small caches
  - To boost memory throughput
- Simple control
  - Simplistic branch prediction
  - No data forwarding
- Energy efficient ALUs
  - Many
  - Long latency but heavily pipelined for high throughput
- Require massive number of threads to tolerate latencies
  - Threading logic
  - Thread state



CPU vs GPU



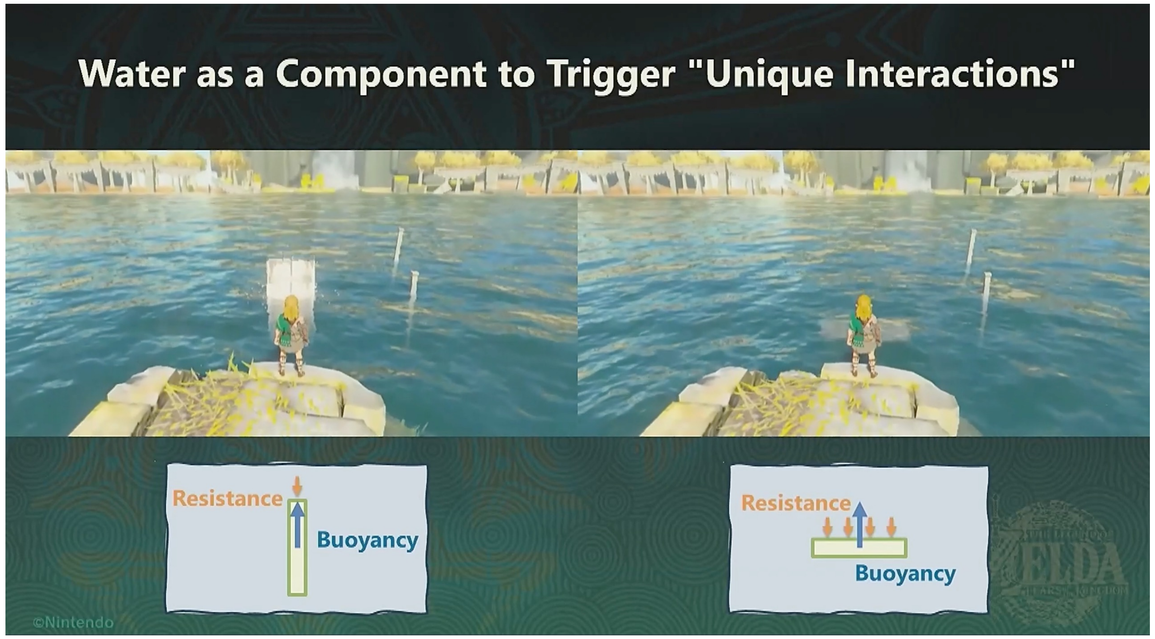
CPU



GPU

Demanding Applications Use Both in Tandem

- NPC interactions
- *Physics simulation*
- Character animations
- Path finding
- Game logic
- Texture prefetching



From presentation Tunes of the Kingdom: Evolving Physics and Sounds for The Legend of Zelda: Tears of the Kingdom

- Rendering
- Shaders
- Lighting, reflections
- Normal mapping
- Level of detail
- Filtering

CPU ***GPU***

In Scientific Computing This Is Also the Case

- CPUs excel at **sequential problems** where **latency matters**.
- GPUs excel at **parallel problems** where **throughput matters**.

Constructing a tower is faster on a sequential processor:
each level requires the previous one!

Image by freepik

Calculating an image filter is faster on a parallel processor:
each pixel only requires the neighbouring ones!

Table of Contents

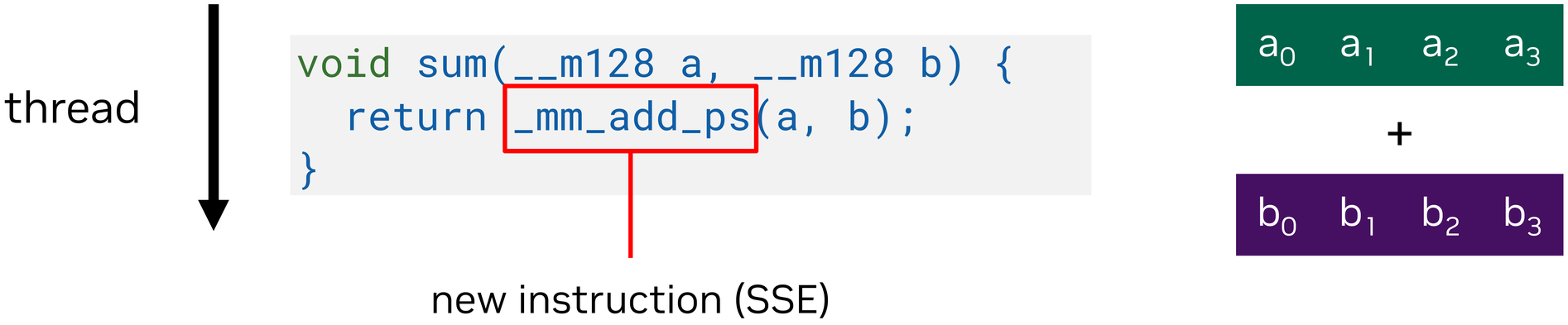
SIMD and SIMT

- Modern CPUs are **SIMD** (*single instruction multiple data*)
  - A thread can mutate several pieces of data owned by the same thread.

- GPUs are **SIMT** (*single instruction multiple thread*)
  - Multiple threads are executing, each with its own data.

SIMD Example

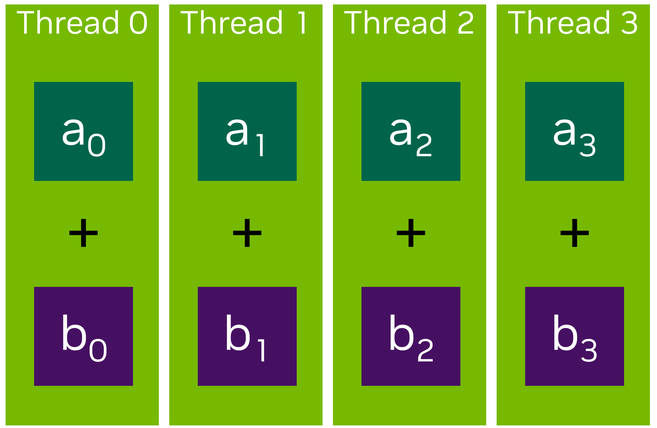
- SIMD expands the set of instructions (ISA) with extensions:
  - SSE, SSE2, SSSE3, SSE4.1, SSE4.2, POPCNT, AVX, AVX2, AVX512-F, …
- These extensions provide instructions that can run over several data simultaneously.



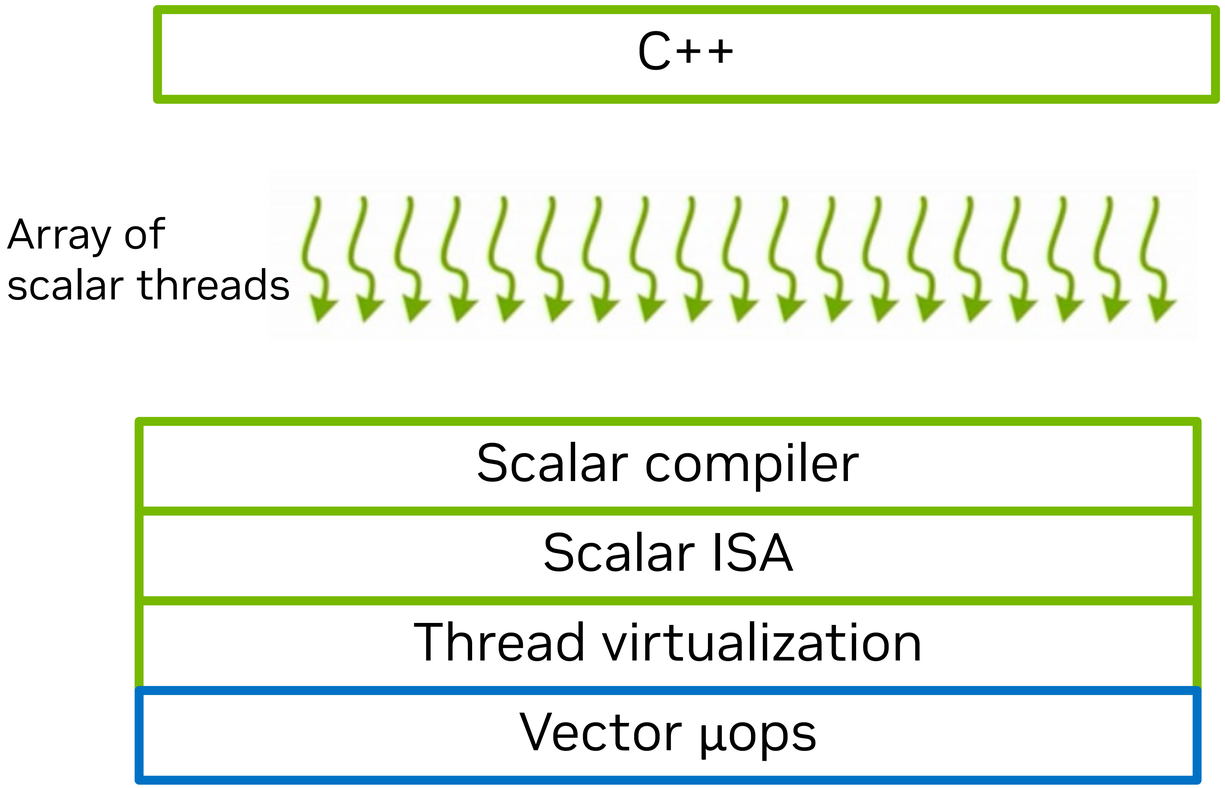
SIMT Example

- In SIMT, one uses the same set of instructions as for single threads.
- They are executed by a *gang of threads* (in CUDA terminology a warp, more details in *Performant programming for GPUs*).
- Each thread has its own:
  - Program Counter – *Pointer* indicating executing instruction.
  - Registers
- The ISA is extended with special instructions like barriers, atomics, etc.
  - Most code remains familiar.

```
void sum(float* a, float* b, int i) {
  return a[i] + b[i];
}
```



Many Scalar C++ Threads



- CUDA exposes a scalar set of instructions (ISA). Each thread will run the same code.
- This choice eases syntax for the user.
- The hardware manages the complexity of many alive threads (100k+ threads possible).
- The performance hit of managing this complexity is negligible.

Single Instruction Multiple Thread

- SIMT stems from *Render Man*, language developed by Pixar in the 80's.
  - C-like language where one would write what each color channel should do.
  - Program is run by a *gang of 4 threads*.
- *Programmable pixel shaders* became supported with GeForce 3 (2001).
  - *Pixel shaders* calculate effects on a per-pixel basis: pixels are rendered, lit, shaded and colored.
  - Shortly after, *geometry shaders* were introduced, which allow transformations such as *tessellations*.

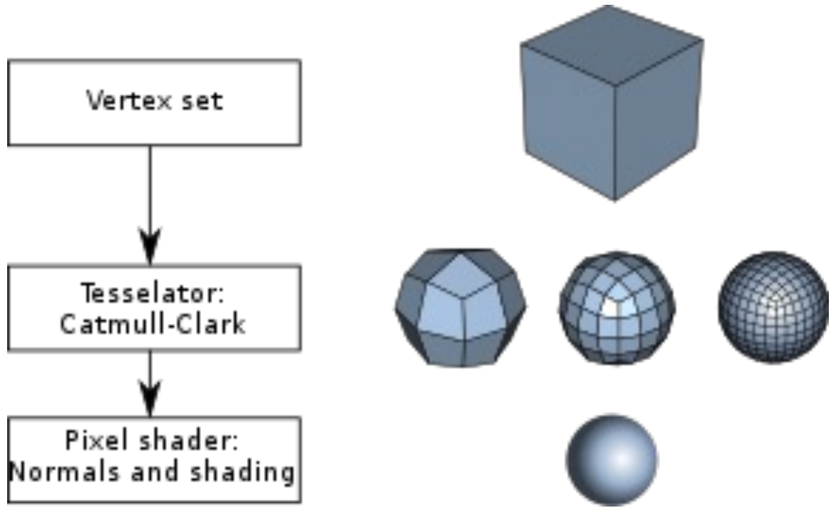
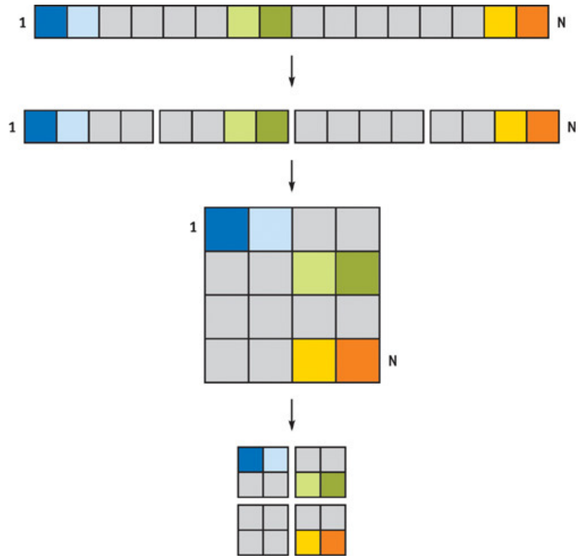


Image from wikipedia

General Purpose GPU

- Shaders were programmed at a low-level.
  - A portable, higher level language for programming the GPU was created: **Cg** (C for graphics; aka HLSL).
  - Cg outputs DirectX or OpenGL, standard APIs for graphics.
  - Other similar languages exist, such as GLSL.

A screenshot from Far Cry, programmed in Cg.

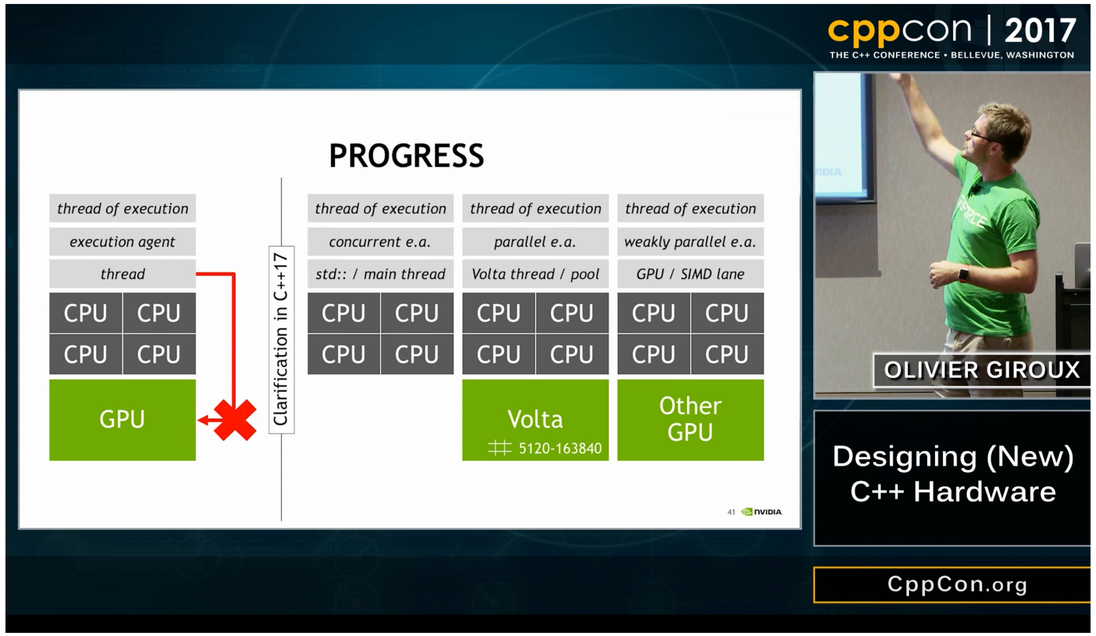


Vector representation in shaders (GPU Gems 2)

- Scientists started using GPUs through shader languages.
  - Adapting the use-cases to the available 4-way vector APIs.
  - GPU Gems 2 has some fun examples.
  - Significant speedups were observed for a few applications.

CUDA

- GPU architecture evolved to be more homogeneous.
- *Compute Unified Device Architecture* (CUDA) was created in 2006.
  - It hid the complexity of shaders behind a SIMT language extension to C.
  - Shortly after, many papers were published reporting speedups in many fields.
- Successive iterations of GPUs are now created as hardware optimized for C++.
  - Impacting future versions of the C++ standard as well.
  - CUDA keeps up with the latest C++ standard support (C++20 at the moment).



From talk [Designing (New) C++ Hardware]

Artificial Intelligence

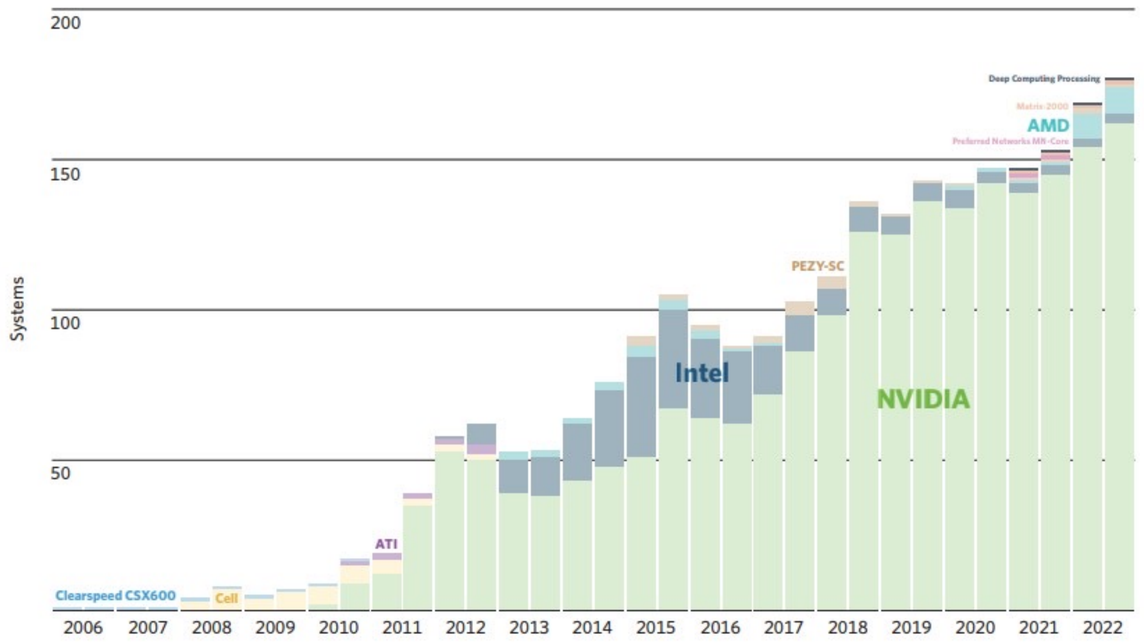
- Today, GPUs are widely used as processors for AI.
  - Both learning and inference.
  - All major AI-oriented libraries support GPUs.
- GPUs are the preferred platform to run AI:
  - Hardware support specific to speedup matrix-matrix multiplication and addition.
  - Software support.
- CUDA provides access to these low-level optimizations.
- This trend is likely to continue and drive hardware design in the future.

Tensor Cores take up about a third of arithmetic space on the die.

So, What Are GPUs Useful For?

- **Parallelizable problems**.

Data Intensive Analytics	Medical Imaging	Financial Analysis	Scientific Simulation
Biomedical Informatics	Electronic Design Automation	Digital Audio Processing	Digital Video Processing
Engineering Simulation	Statistical Modeling	Numerical Methods	
Computer Vision	Ray Tracing Rendering	Interactive Physics	

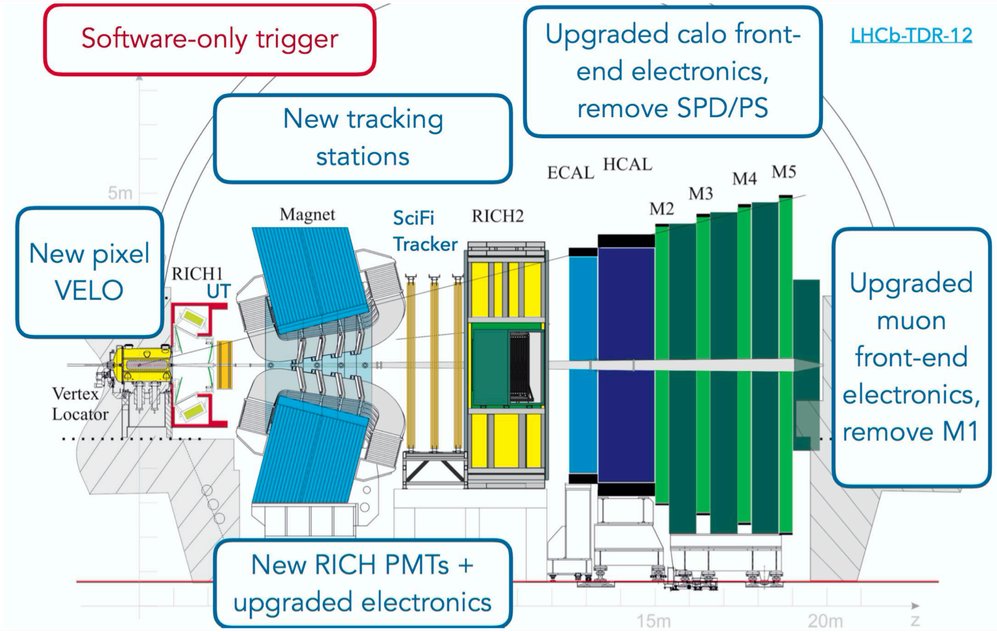


Accelerators in the top500

- Most data centers have GPUs today.
- Making a successful GPU application requires a good understanding of the **hardware**, **software**, and **a lot of hard work**!

Table of Contents

The LHCb U1 Upgrade



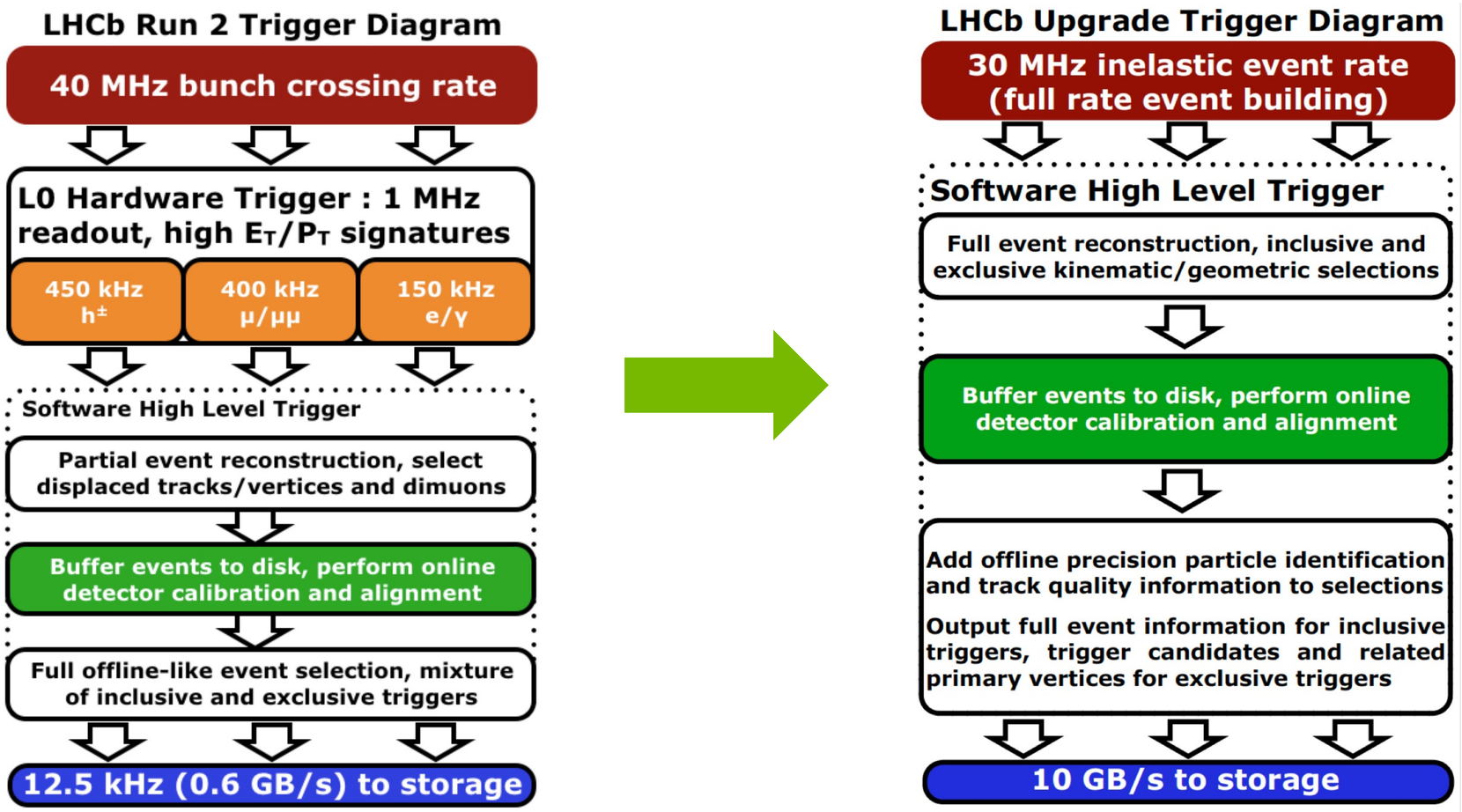
The LHCb detector at CERN:

- Single-arm forward spectrometer for high-precision flavour physics
- High precision tracking and vertexing.
- Complemented with excellent Particle Identification.

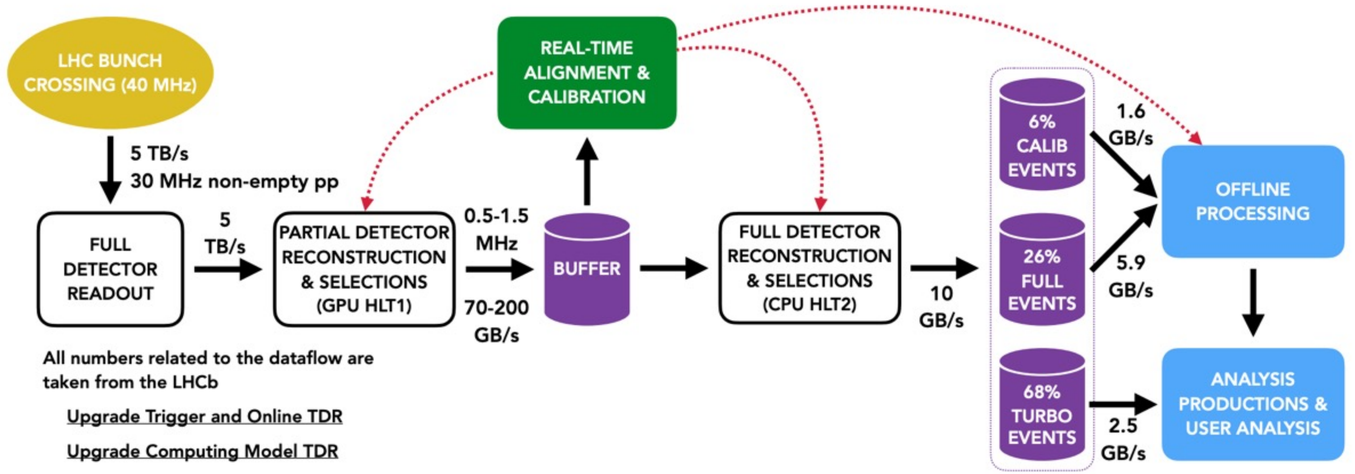
The U1 upgrade:

- Instantaneous luminosity increased by 5x.
- Major upgrade in all sub-detectors to handle increased rates.
- Software-only trigger!

Someone Had to Pull the (Hardware) Trigger

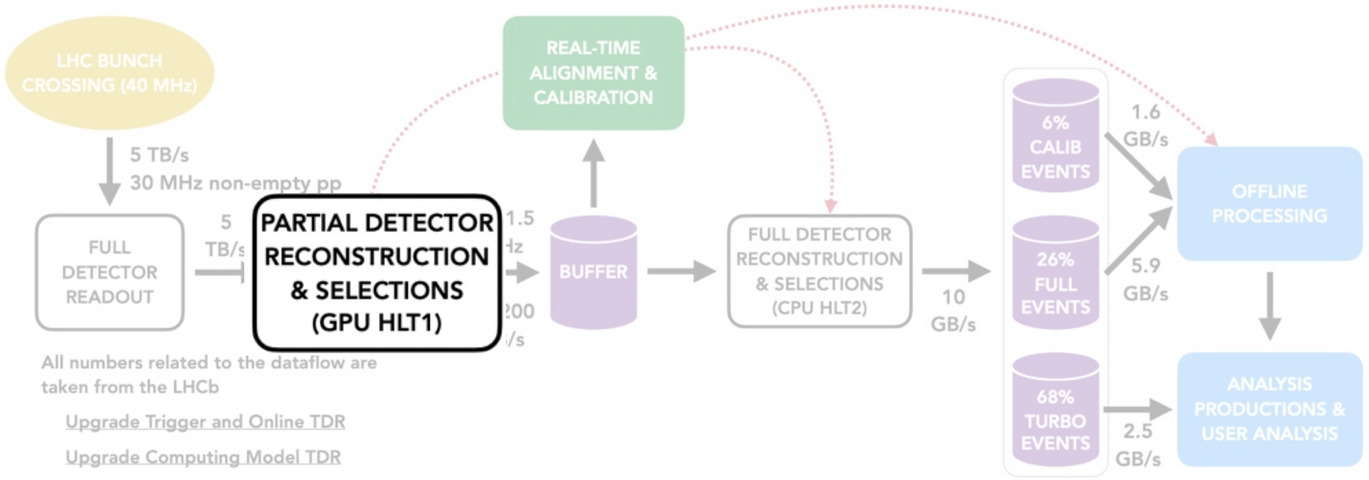


The LHCb Data-flow



- Detector data recieved by ~500 FPGAs and built into events in the event building (EB) farm servers.
- 2-stage software trigger: HLT1 and HLT2.
- Real-time alignment and calibration.
- After HLT2: 10 GB/s of data for offline processing.

The LHCb First Level Trigger



The goal of HLT1:

- Be able to intake the entirety of the LHCb raw data (5 TB/s) at 30 MHz.
- Perform partial event reconstruction and coarse selection of broad LHCb physics cases.
- Reduce the input rate by a factor of 30 (~1 MHz).
- Store selected events in intermediate buffer for real-time alignment and calibration.

[Massive] Parallelism in HLT1

In LHCb alone, 32 Tbits per second are being processed in Run 3.

These data are structured as follows:

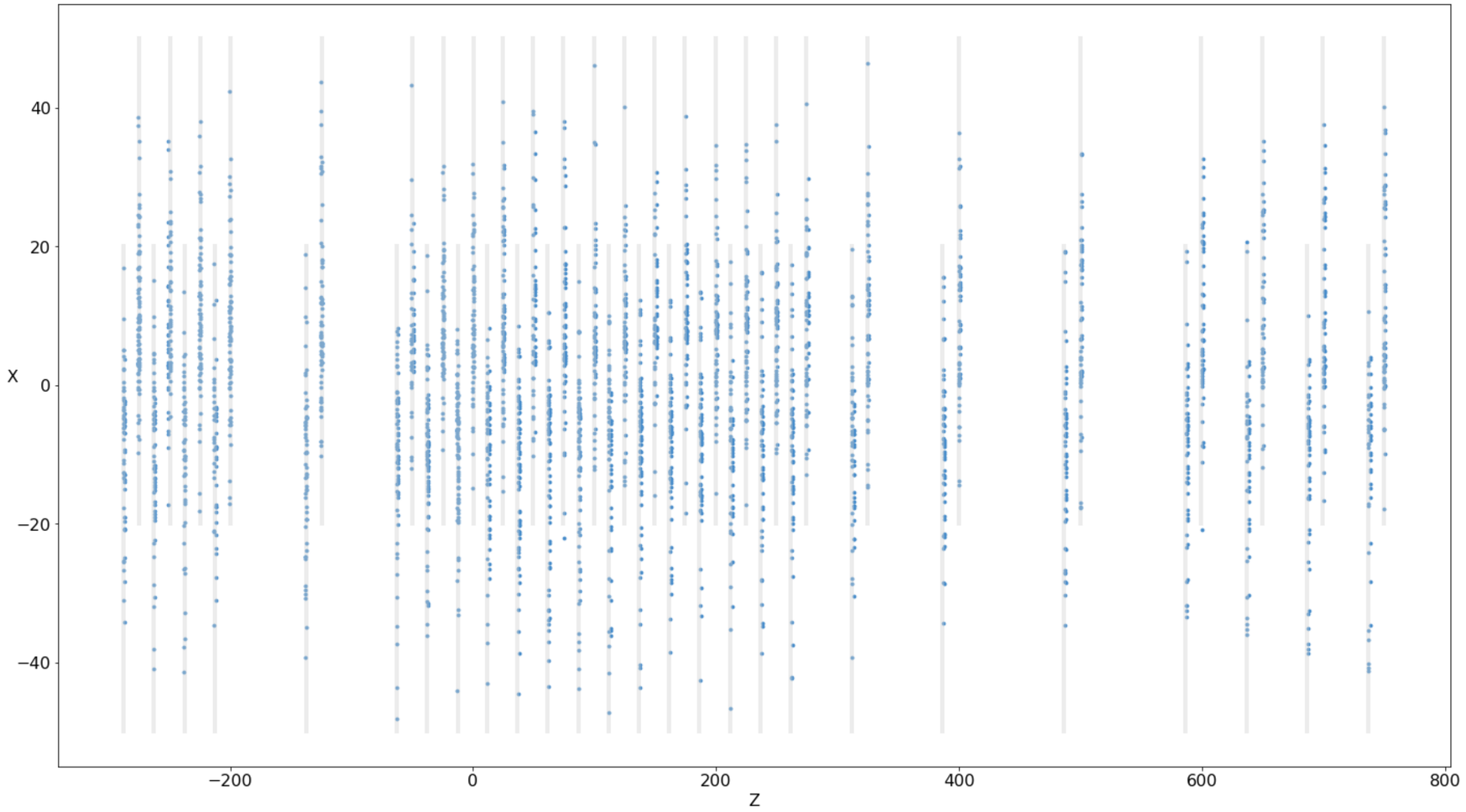
- **30 million independent collision events** per second, 100 kB each.

For each event:

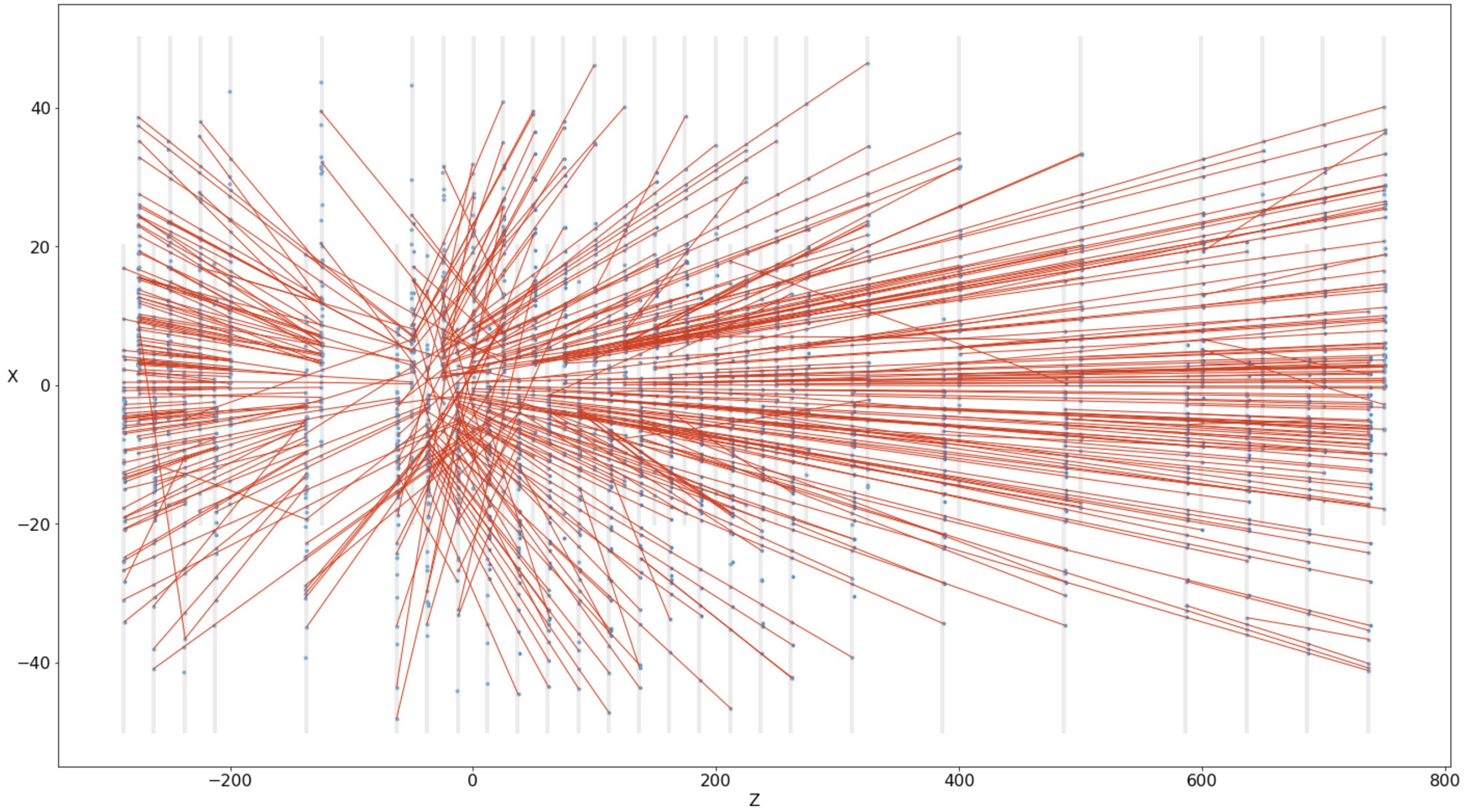
- Four tracking problems – **100s of tracks** per event each.
- Vertex finders – **10s of vertices** per event.
- Kalman filter – **100s of instances** per event.

Looks like a problem where massively parallel architectures can make a difference!

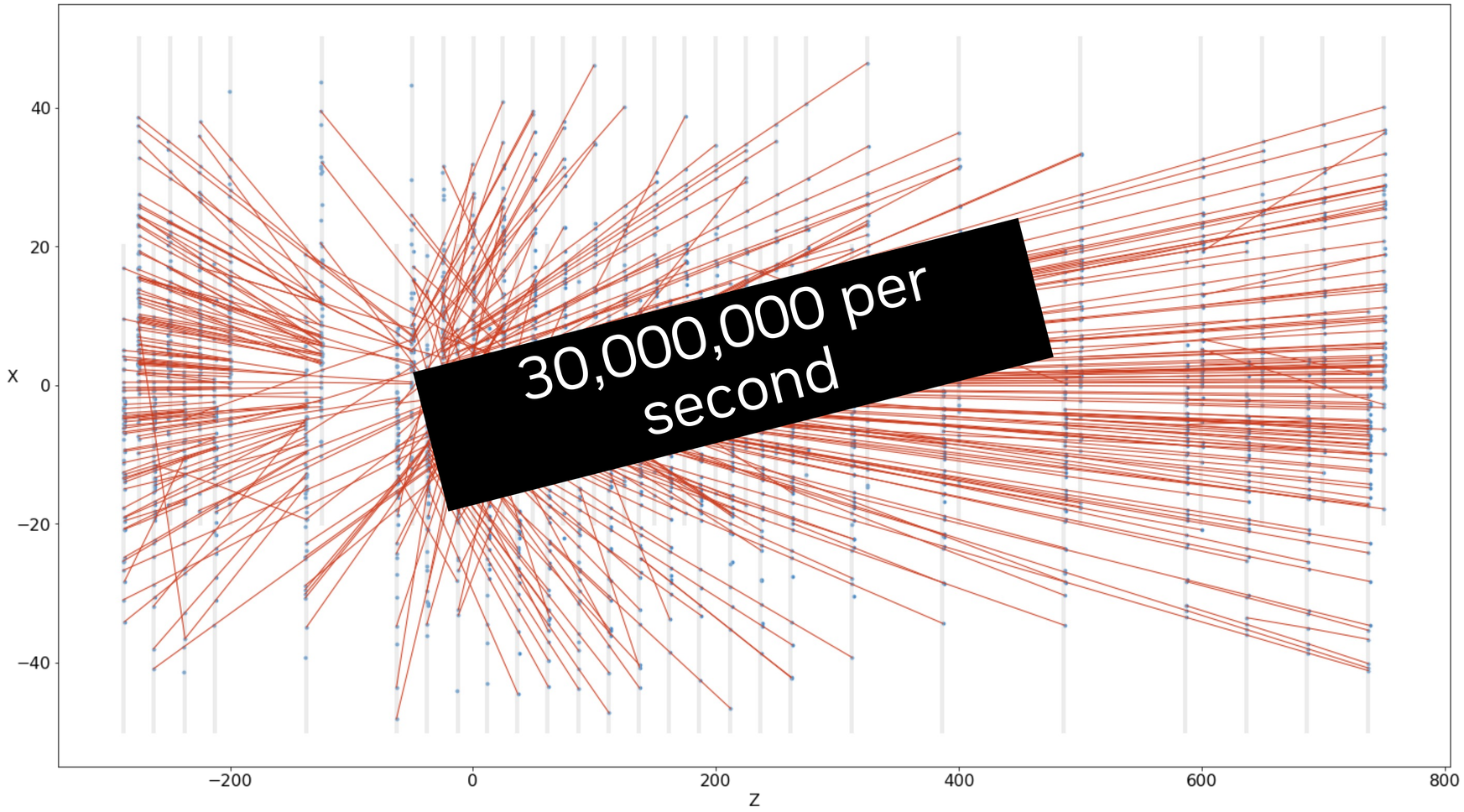
Going From This...



Into This...



At a High Rate!

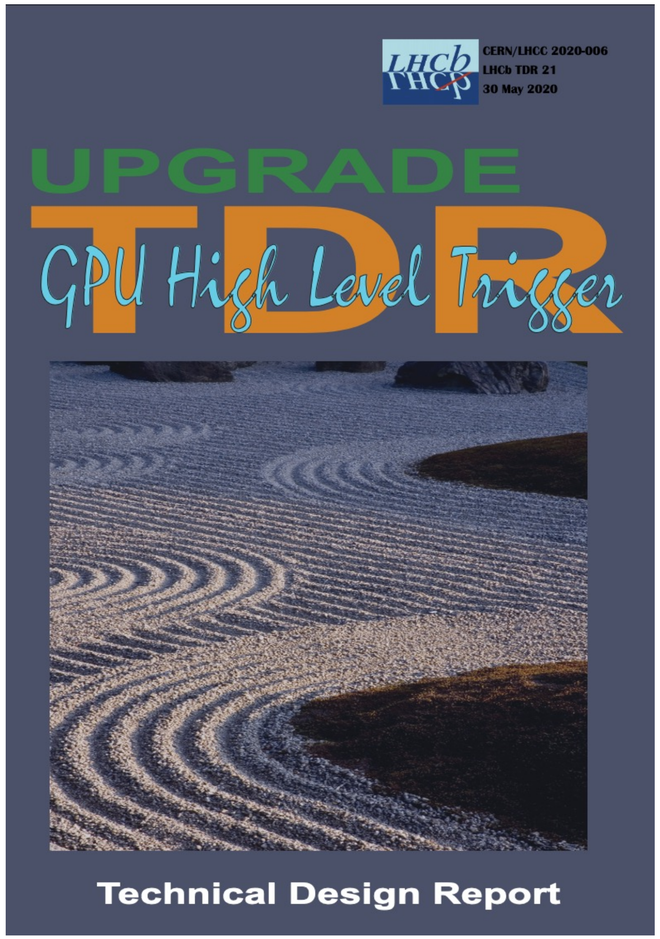


Allen

The Allen framework is a modular, scalable and flexible framework for physics reconstruction on accelerators.

Features:

- Supports CPU, CUDA.
- Uses C++17.
- Multi-threaded, pipelined, configurable framework.
- Multi-event scheduler, event batches support.
- Custom memory manager, no dynamic allocations, flexible datatypes.
- Built-in validation. Generation of graphs with ROOT.

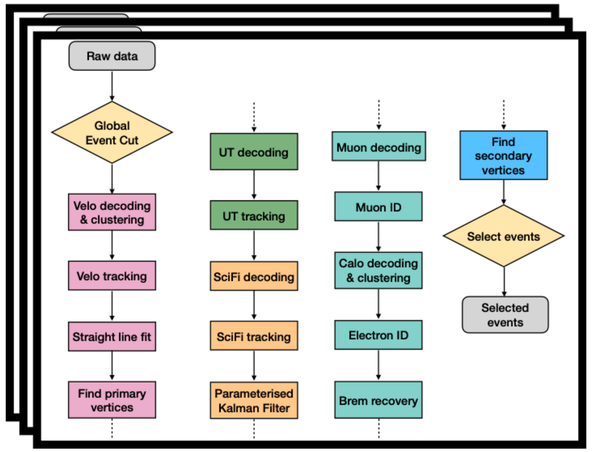


http://cds.cern.ch/record/2717938/files/LHCB-TDR-021.pdf

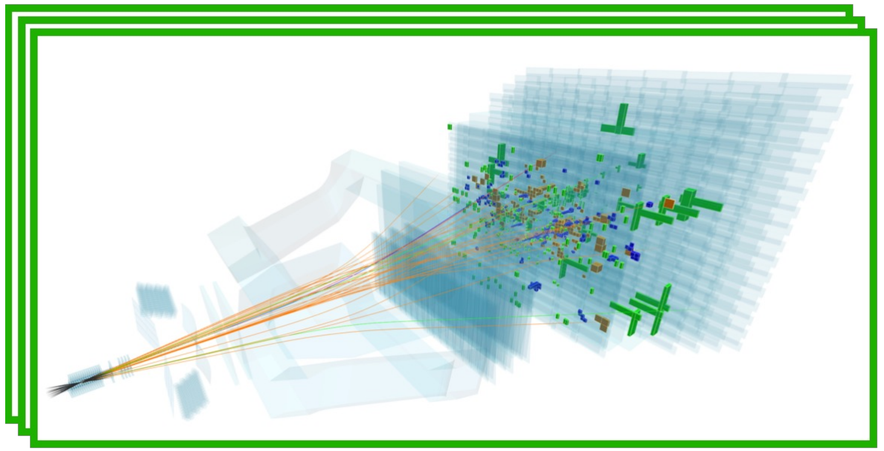
Parallelization

We parallelize at three levels:

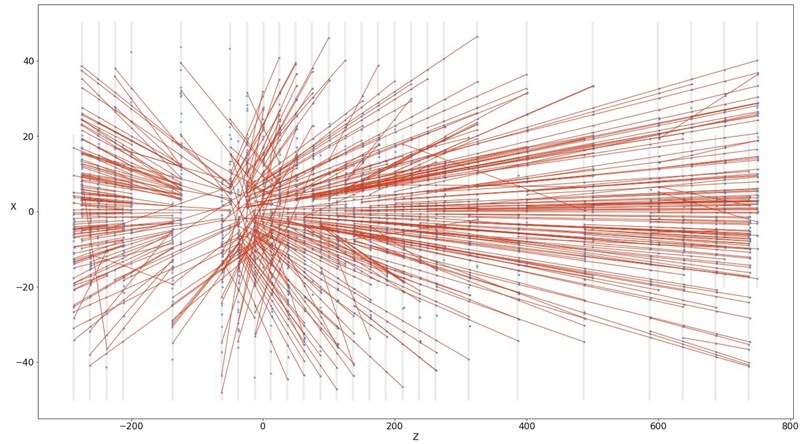
Sequences



Events



Intra-algorithms



	Sequences	Events	Intra-Algorithms
CPU		Threads	Vectorisation
GPU	Streams	Blocks	Threads

NVIDIA

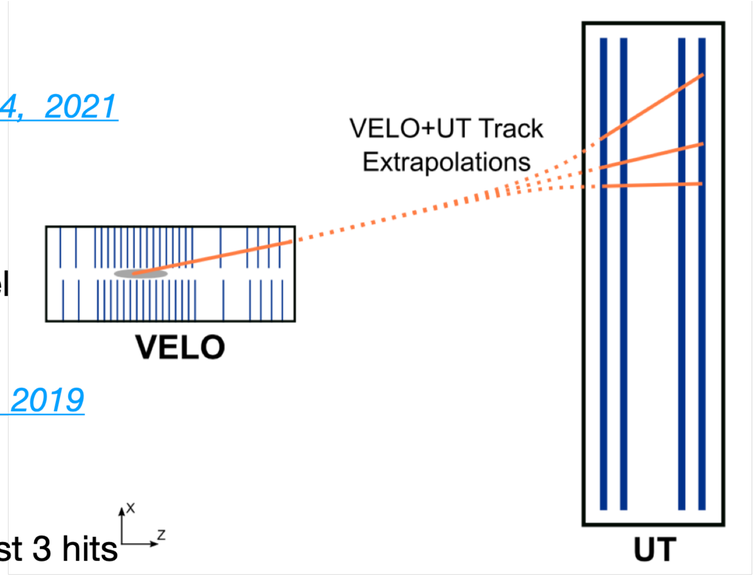
Track Reconstruction at LHCb HLT1

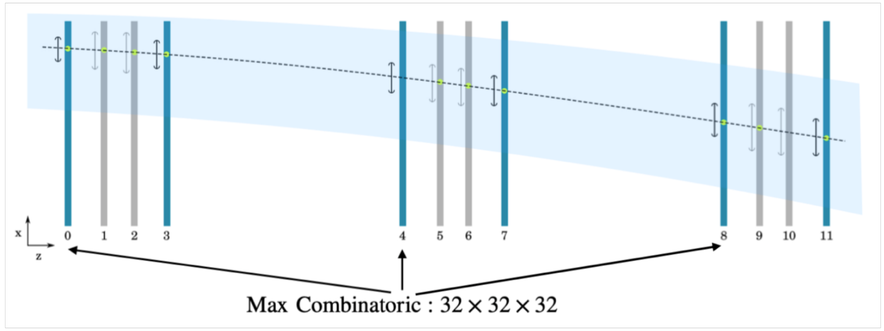
Velo tracking: *Journal of Computational Science, vol. 54, 2021*

- 52 silicon pixel modules with $\sigma_{x,y} \sim 5\ \mu m$
- Parallel local tracking algorithm: *Search by Triplet*
- Tracks fitted with simple Kalman filter assuming straight line model

Velo-UT tracking: *IEEE Access, vol. 7, pp. 91612-91626, 2019*

- 4 layers of silicon strips
- Velo tracks extrapolated to UT taking into account fringe B-field
- Parallelized tracklet finding inside search windows requiring at least 3 hits



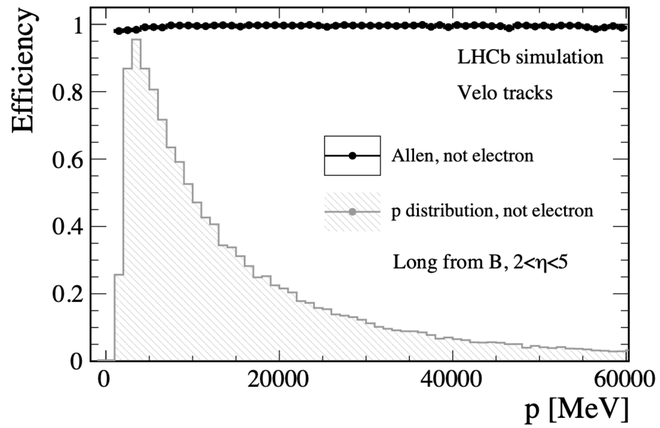


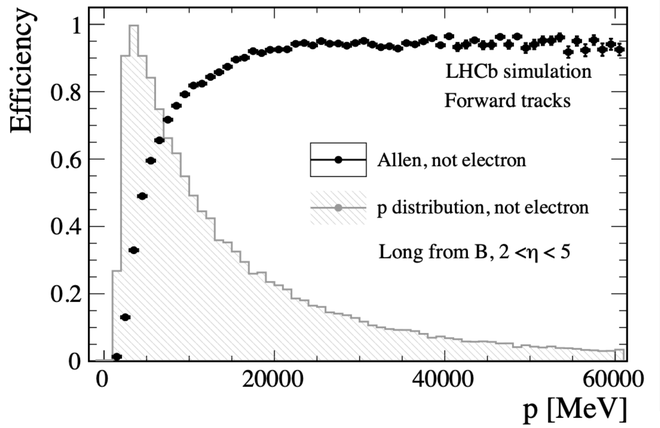
Forward tracking: *Comput Softw Big Sci 4, 7 (2020)*

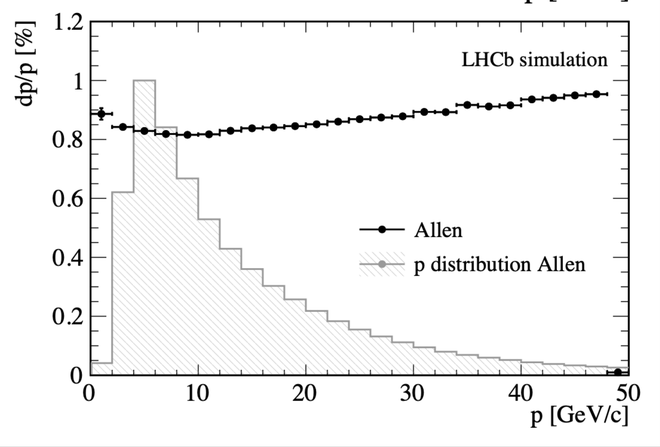
- 3 stations with 4 layers of Scintillating Fibres
- Velo-UT tracks extrapolated using parametrization
- Parallelized *Forward algorithm* to reconstruct **long tracks:**
  - Search windows from on Velo-UT momentum estimate
  - Form triplets and extend to remaining layers

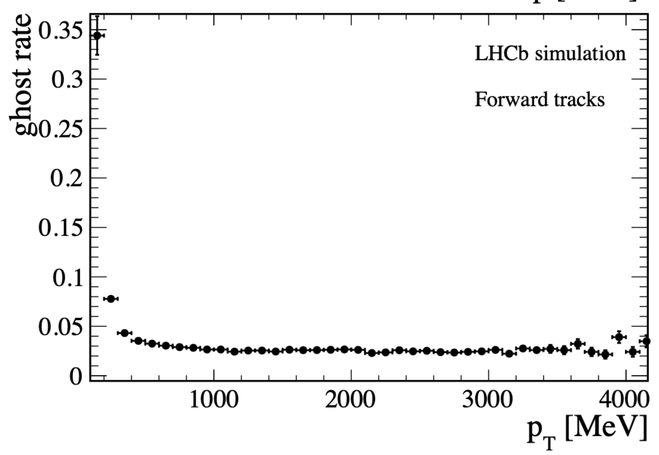
With State-of-the-Art Efficiencies

- **Run 2 efficiency maintained at x5 instantaneous luminosity**
- Excellent track reconstruction efficiency (> 99% for VELO, 95% for high-p forward tracks)
- Good momentum resolution and fake rejection







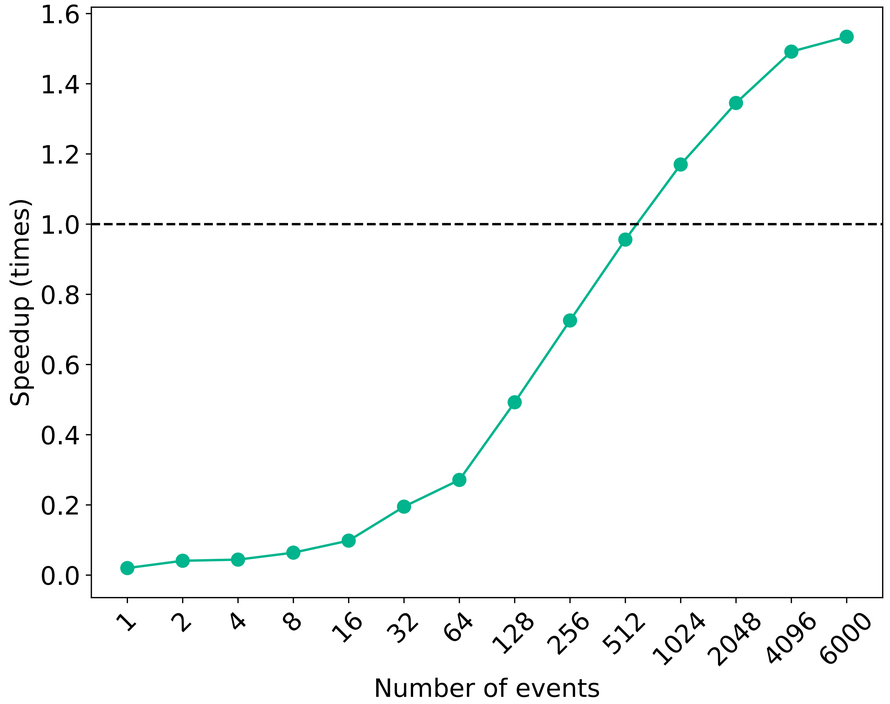


LHCb-FIGURE-2020-014

Not Always a Path of Roses

Are GPUs a match?

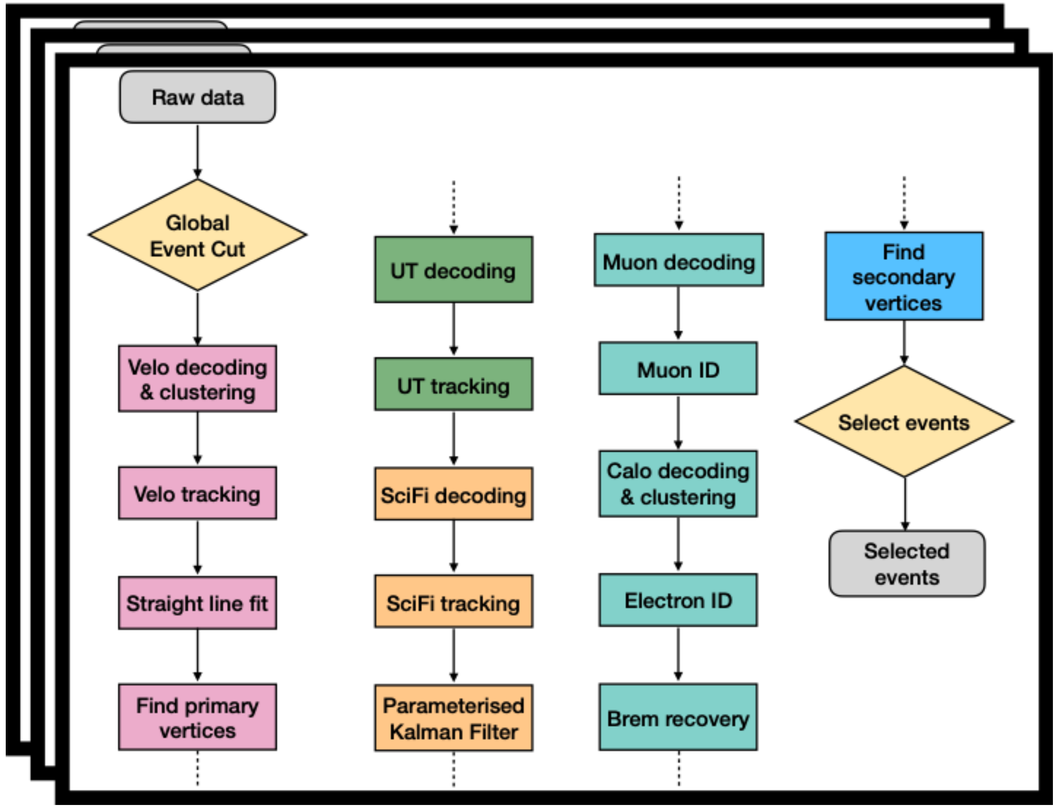
- Not everything was a match from the beginning, we worked through issues.
- Eg. event size in LHCb is too small – only 100 kB. Processing one event at a time is very inefficient! GPUs need more work to be efficient.
  - Pipelining event processing wasn't good enough: we would need to produce events at a pace that keeps up with the GPU.
- ***Demonstrators are fundamental!***
  - To convince your colleagues.
  - To convince yourselves.
  - To prove your hypotheses.
- Allen was designed to process **tens of thousands of events in flight**.



Not Always a Path of Roses (2)

Are GPUs a match?

- Events don't always follow the same execution path. Processing **tens of thousands of events** surely leads to a lot of **branching**!
  - "Select events" runs O(100) algorithms, the outcome of each is not known beforehand.
  - Can one afford to cover all cases for all events sequentially? **No.**

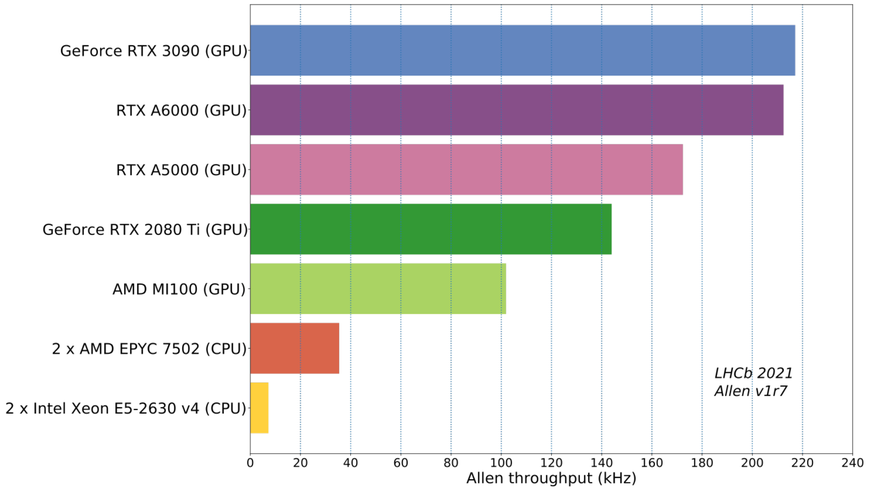


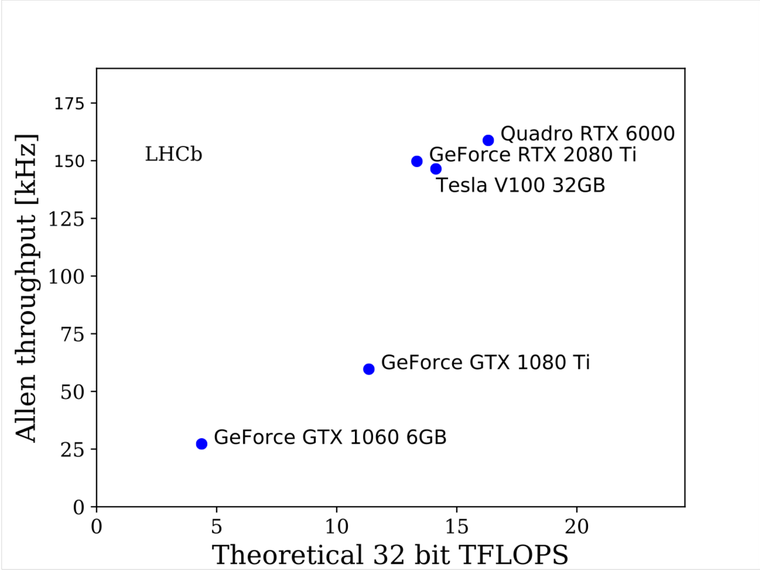
- ***Knowing your hardware is paramount.***
  - To design efficient and scalable solutions.

- Allen produces **a single algorithm execution list** and runs it with a **mask mechanism of active events** (aka *Multi-event scheduling*).

Performance

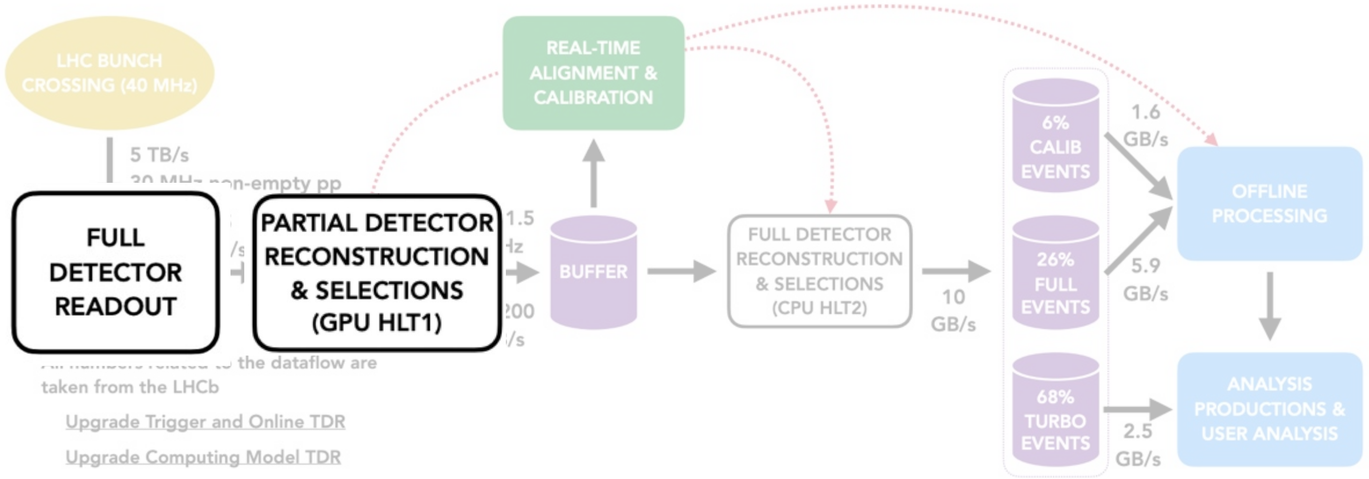
- Making it faster has a big impact: **cost efficient**, **energy efficient**, **wider physics programme**.
  - 30 MHz goal can be achieved with O(200) GPUs (maximum the Event Builder server can host is 500)
  - Throughput scales well with theoretical TFLOPS of GPU card
  - Additional functionalities are being explored





[LHCb-FIGURE-2020-014]

The LHCb Data Acquisition System (DAQ)



- The success of the GPU HLT1 in LHCb is partially due to making a **converged architecture**.
- Combine **detector readout**, **event building** and **event filtering** under the same server farm.

Event Building

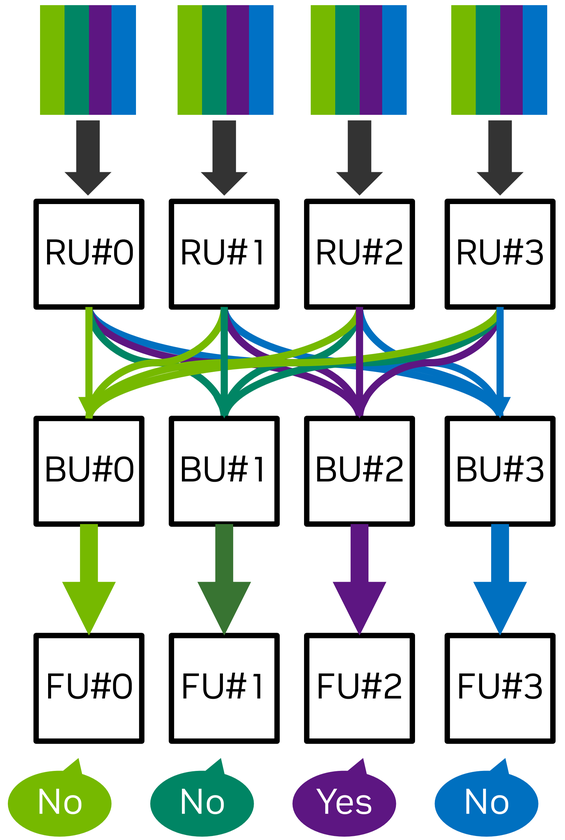
- Each collision of particle bunches (*bunch-crossing*) produces an event.
- Events:

- but events are tiny, sending them one by one wouldn't be efficient for the network.
- So they are bundled:

- But the subdetectors are far apart, so each event is divided in pieces.
- The bundling looks more like:

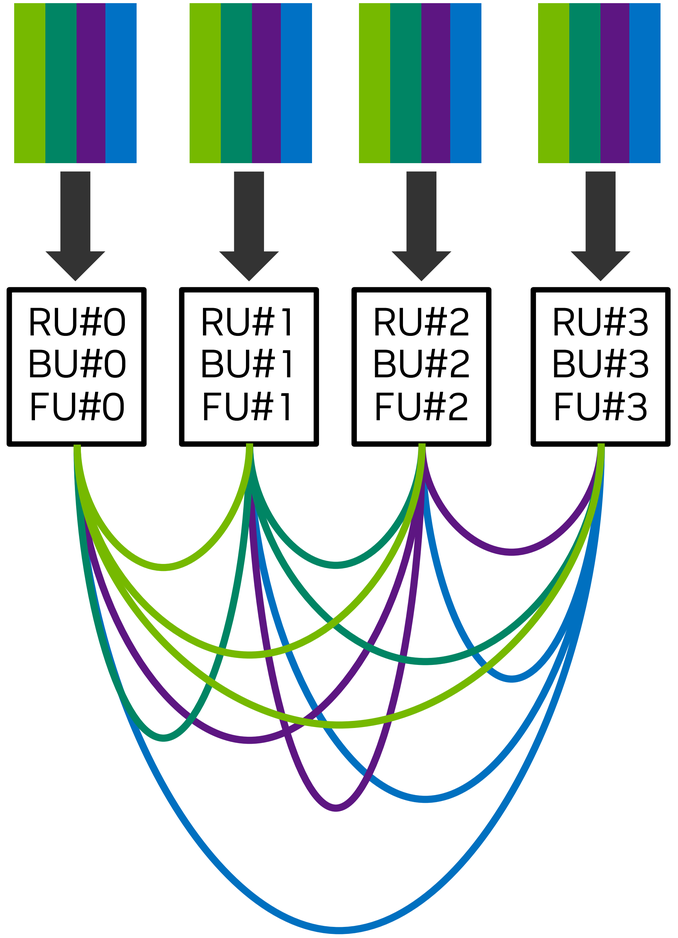
Data Acquisition

- *Readout units* perform the ***readout*** of the detector by looking at these fragments.
- *Builder units* **put together** the fragments of each event.
- *Filter units* **reconstruct** the events and **select** the most interesting ones, discarding the others.
- This is typically done in separate networks and separate server farms!



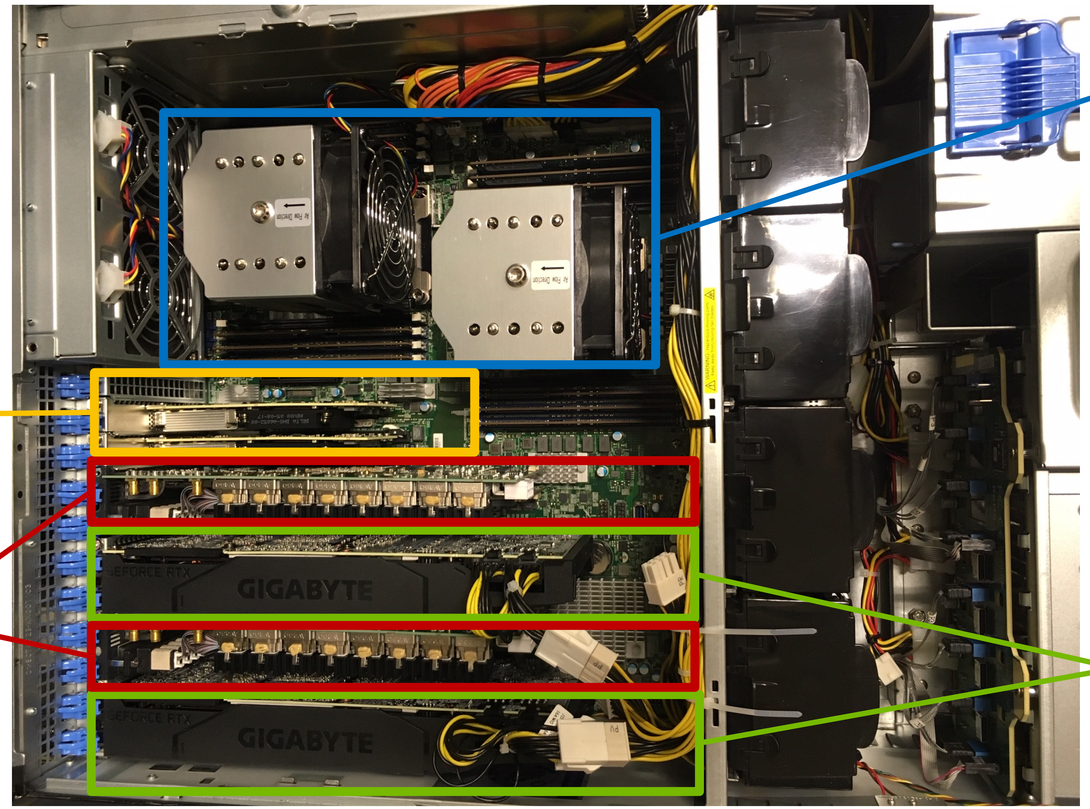
Converged Architecture

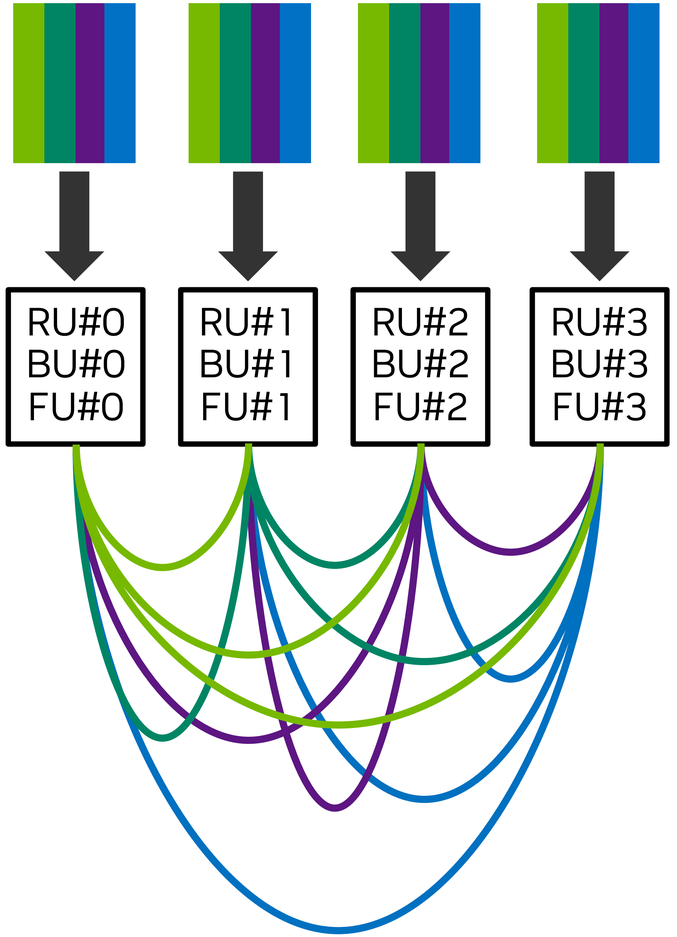
- In LHCb we decided to do it in **one** server farm (with HLT1)!



Converged Architecture (2)

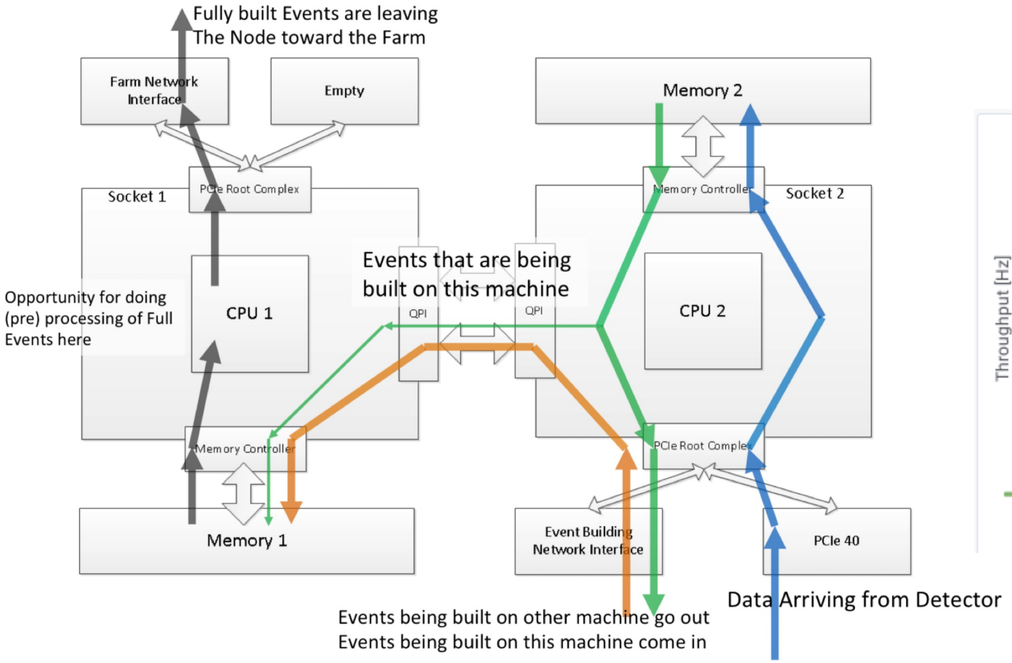
- In LHCb we decided to do it in **one** server farm (with HLT1)!
- It looks pretty awesome.

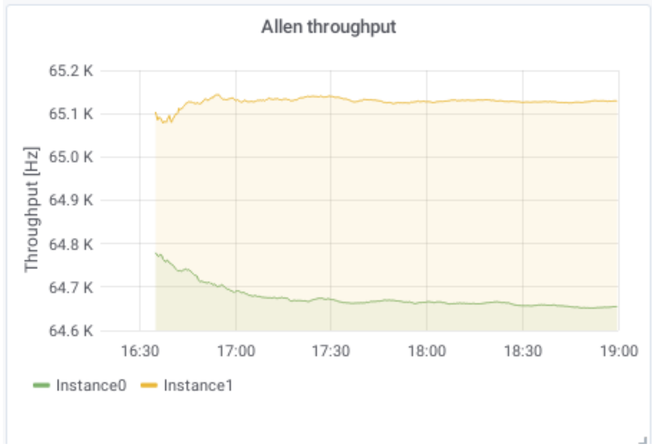


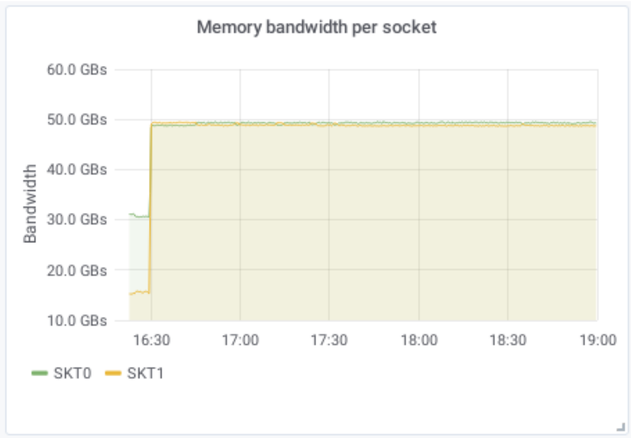


Converged Architecture (3)

- In LHCb we decided to do it in **one** server farm (with HLT1)!
- It looks pretty awesome.
- This put a lot of pressure on CPU memory throughput, CPU utilisation, network throughput. We tested it all…







Converged Architecture (4)

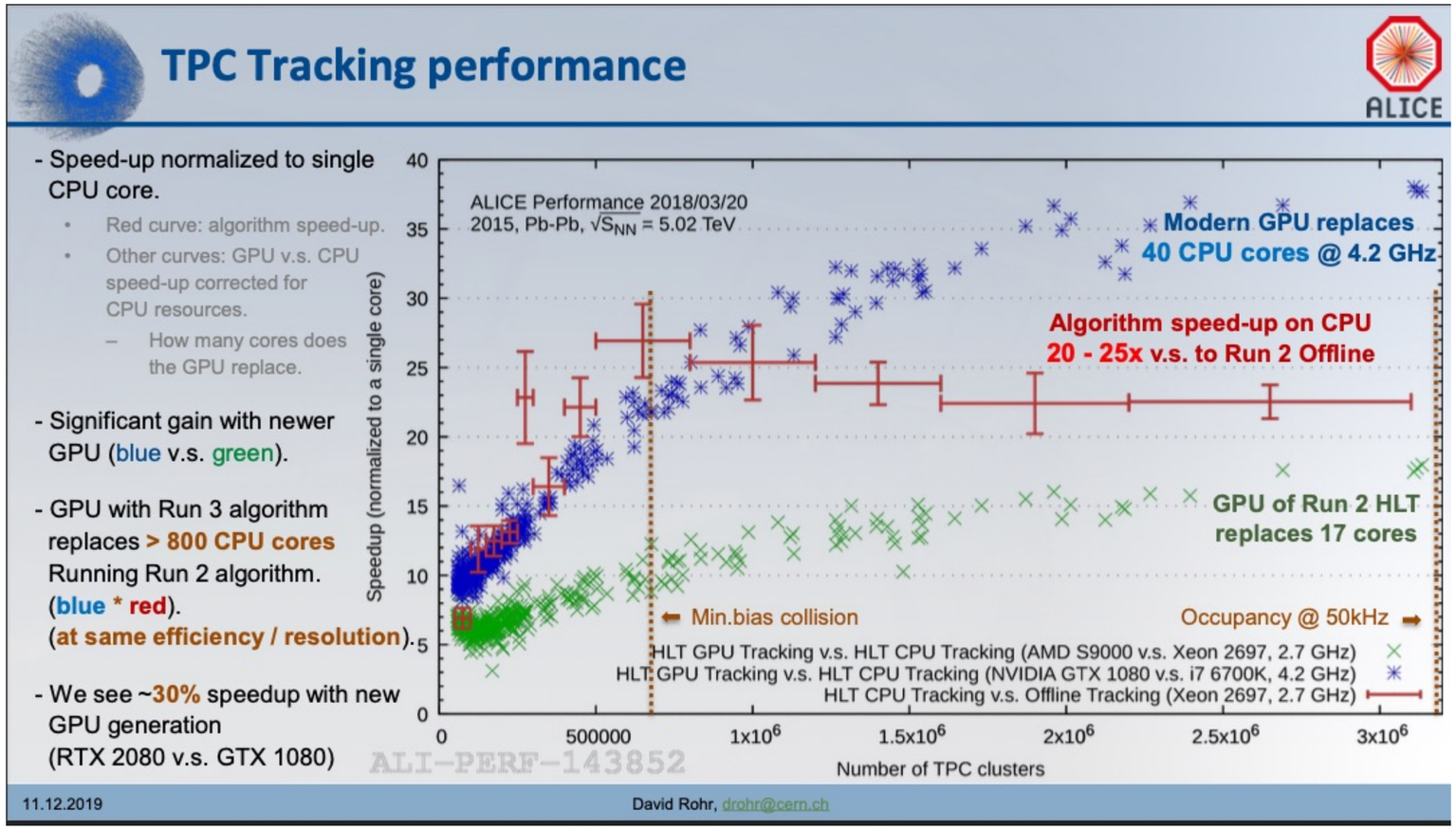
- In LHCb we decided to do it in **one** server farm (with HLT1)!
- It looks pretty awesome.
- This put a lot of pressure on CPU memory throughput, CPU utilisation, network throughput. We tested it all…
- and it led to **huge savings**!
  - Cost, but also easier to maintain (less parts)

*Consider **high-risk high-gain scenarios** in your application.*

Table of Contents

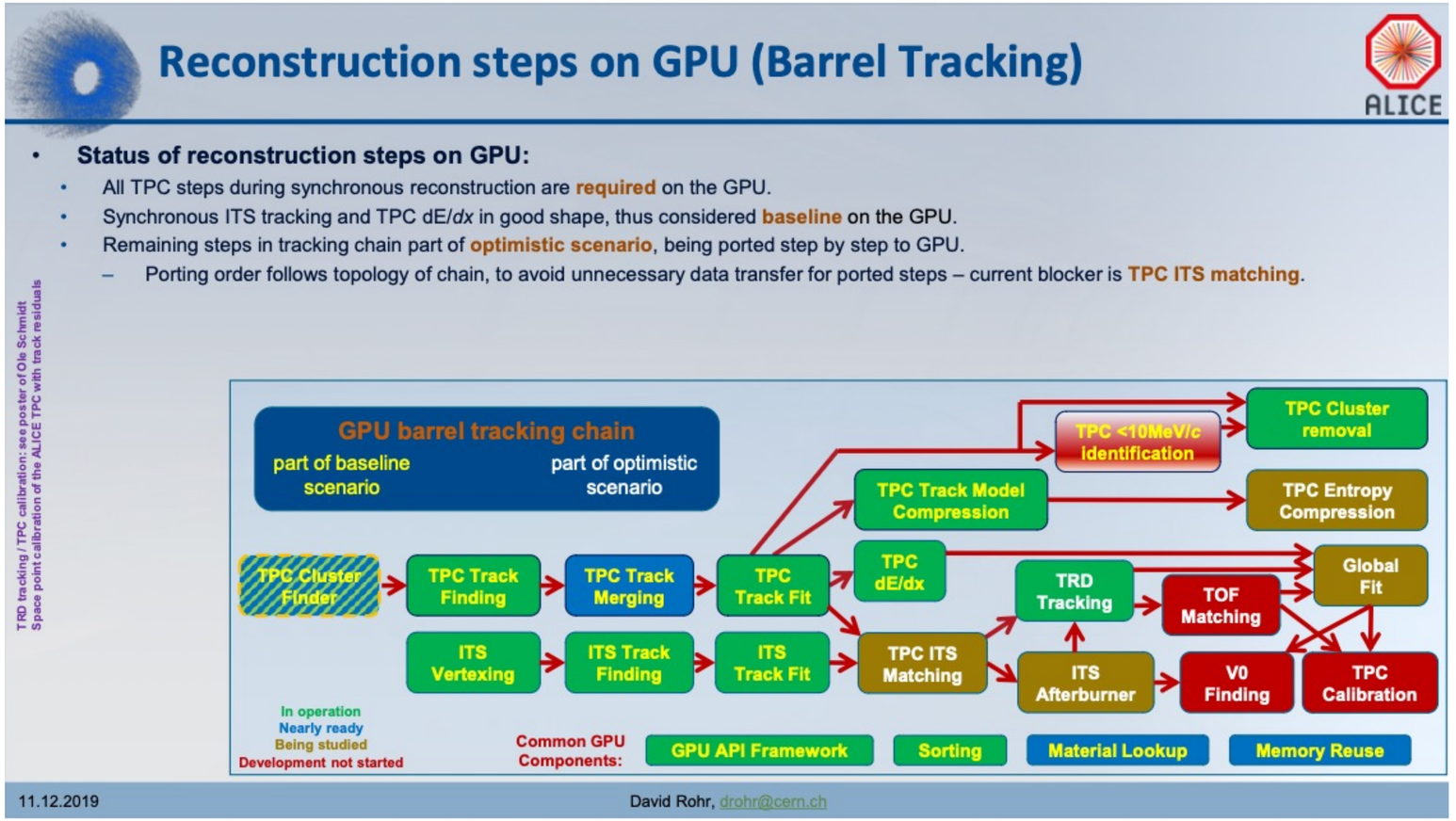
HEP Experiments: ALICE

- ALICE was the first LHC experiment to embrace GPUs.
- Events in ALICE were originally ~40 MBs, dominated by a single tracking problem (the TPC).



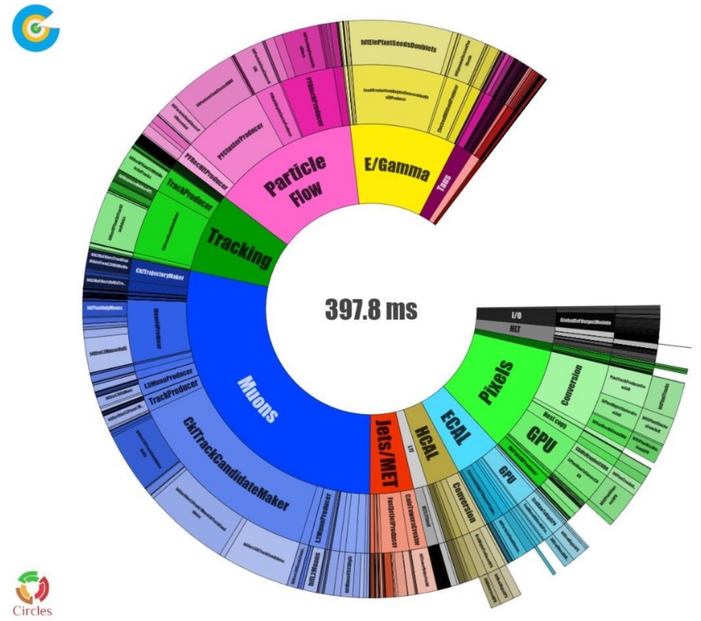
HEP Experiments: ALICE (2)

- Nowadays, most of the reconstruction sequence runs on GPU.



HEP Experiments: CMS

- CMS offloads part of computation to scientific low-profile cards on each node.
  - The HLT menu has total of ~**4400 modules**
  - GPU Offloaded parts
    - Pixel detector reconstruction: from RAW data unpacking up to tracks and vertices (11 modules)
    - ECAL local reconstruction (4 modules)
    - HCAL local reconstruction (3 modules)
  - 57 unique kernels, ranging from 2 μs to 7 ms in these events

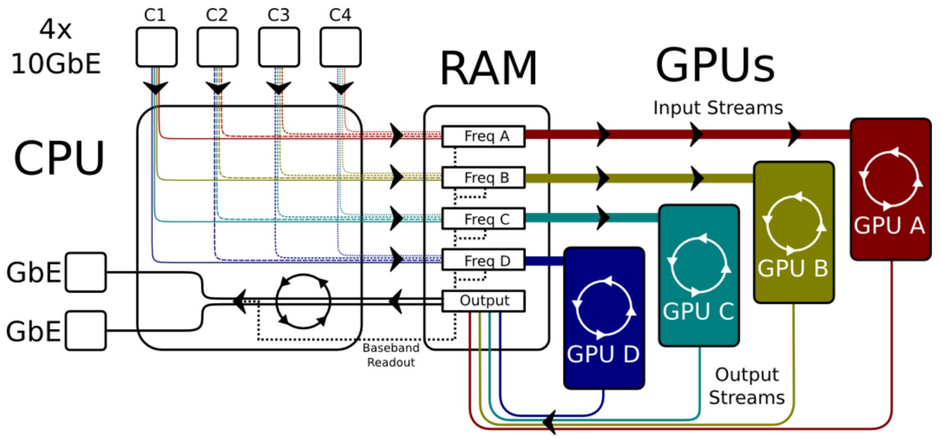


  - Memory pool to amortize cost of raw memory allocations and provide asynchronous allocation interface in CUDA stream order
  - All offloaded modules have CPU versions that are used for reference measurement

From talk CMS by A. Di Florio

Radio Astronomy

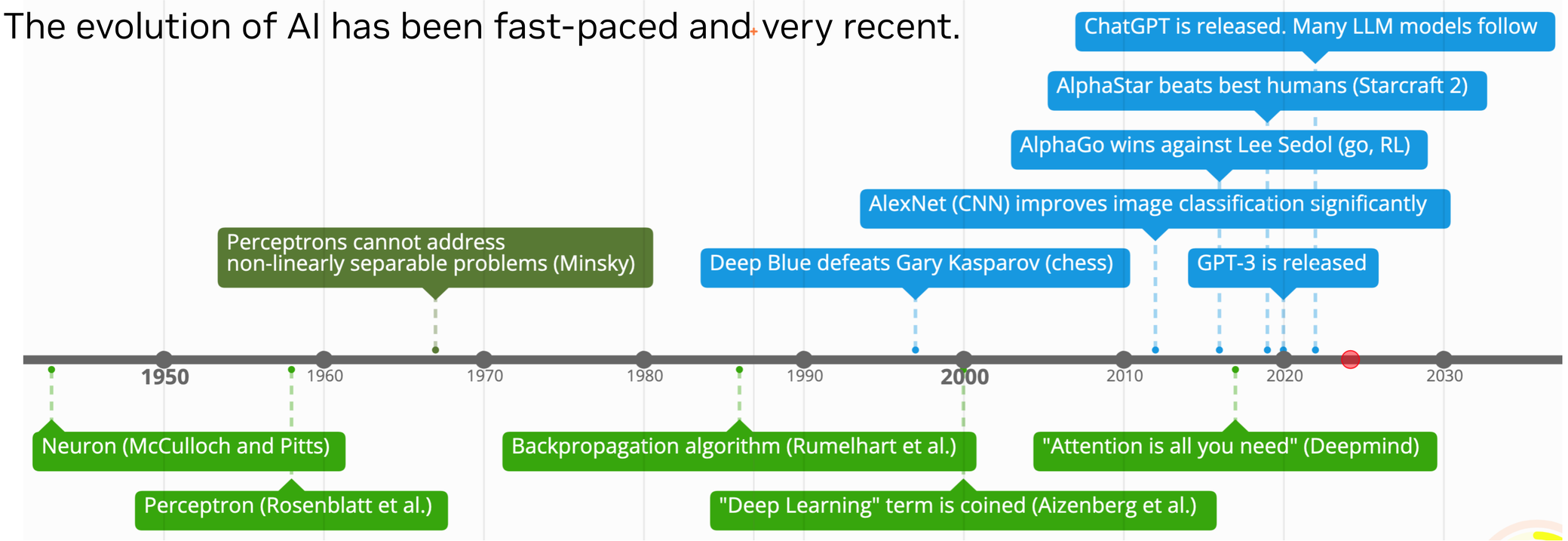
- Radio astronomy also has streaming problems that are solved at a high rate.
- *Correlation* – imaging modes that result in sky-images.
- *Beamforming* – Time series analysis searching for patterns indicative of astronomical events (eg. pulsars, bursts).
- “Event” boundaries are not known in advance.
- “Event” rates are also unknown!

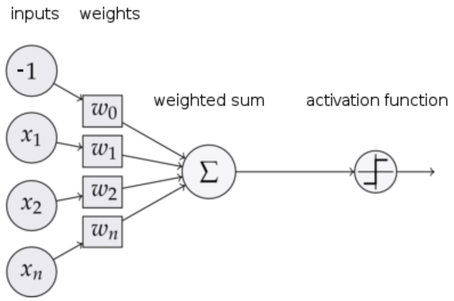


From paper [A GPU Spatial Processing for CHIME]

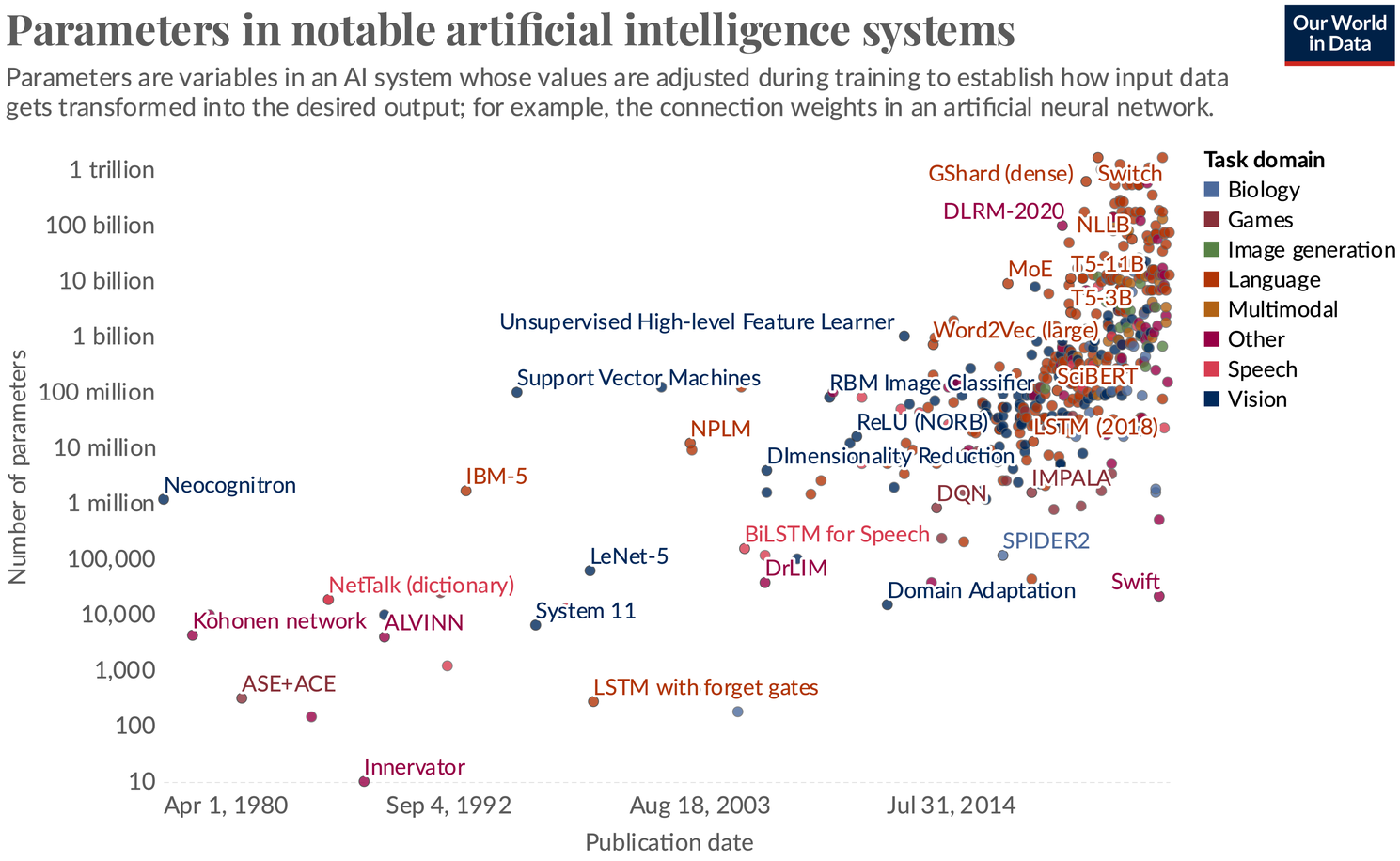
AI

- The evolution of AI has been fast-paced and very recent.





AI Model Sizes

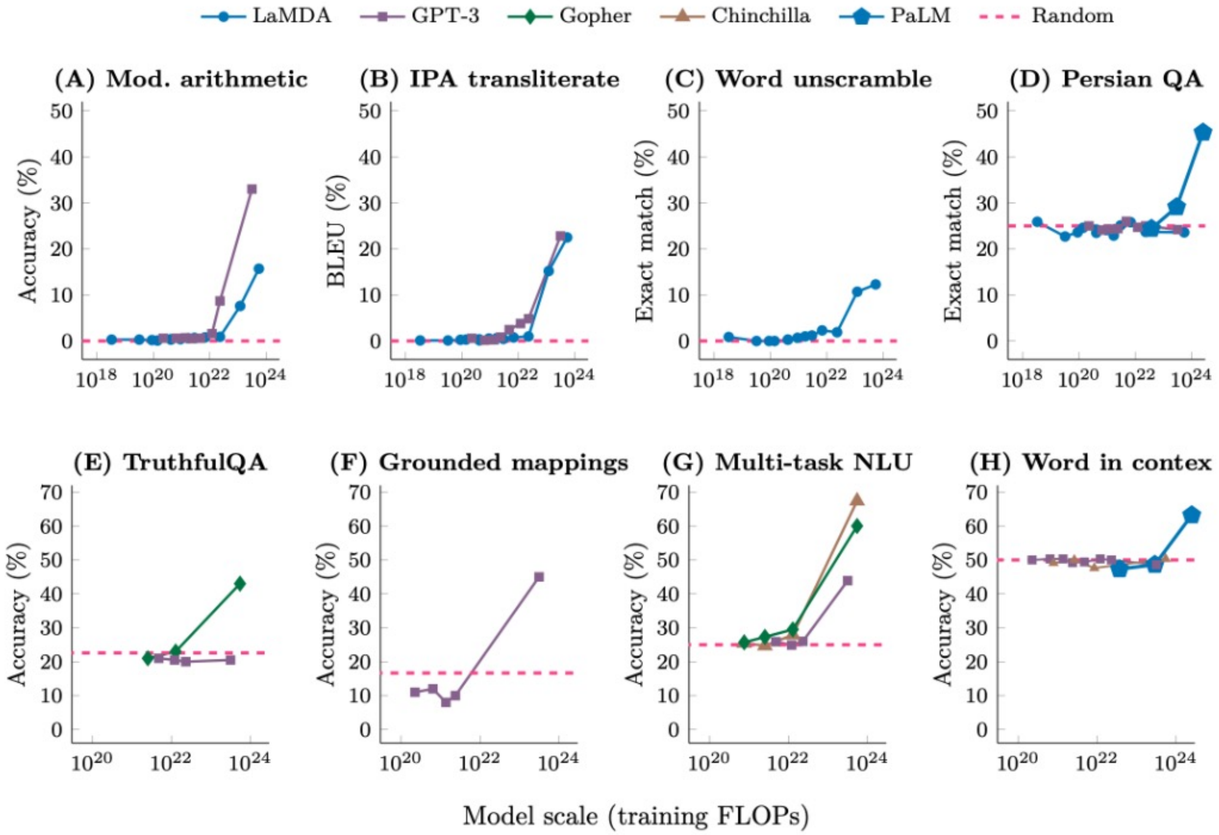


Data source: Epoch (2024) OurWorldInData.org/artificial-intelligence | CC BY

Note: Parameters are estimated based on published results in the AI literature and come with some uncertainty. The authors expect the estimates to be correct within a factor of 10.

Scaling Up Unlocks *Emergent* Capabilities

- LLMs exhibit emergent capabilities that are only available with a large enough model size.



Emergent Abilities of Large Language Models (Wei et al., 2022)

(a) Modified arithmetic

```
In the following lines, the symbol ->
represents a simple mathematical
operation.
100 + 200 -> 301
1 + 1 -> 3
2 + 2 -> 5
```

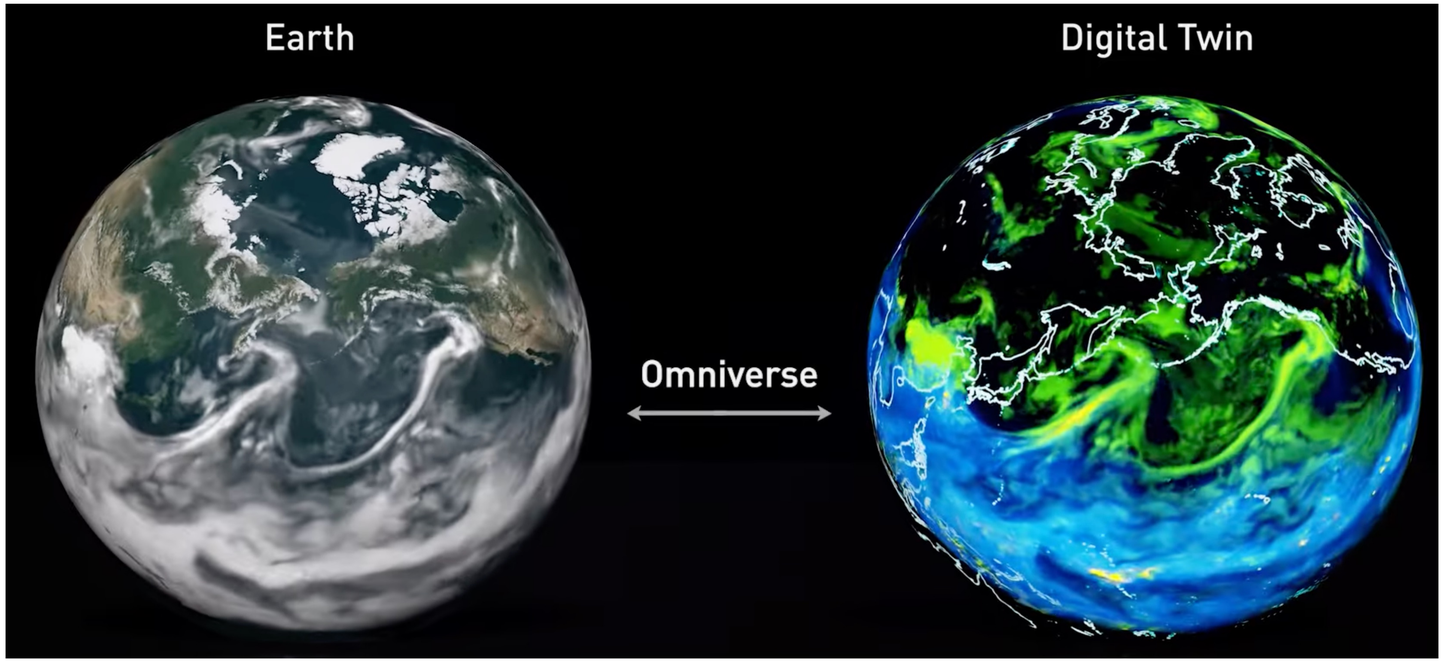
(c) Word unscrambling

```
Input: The word hte is a scrambled
version of the English word
Output: the

Input: The word sohpto is a scrambled
version of the English word
Output: photos
```

An example of AI Today: Weather Prediction

- “Digital Twins” replicate a system to make simulations.
- AI-based predictions are comparable to analytical simulations.
- Performance is **5 orders of magnitude faster** than traditional methods.



From Earth-2

Table of Contents

Summary

- Learn to use your processor efficiently.
- Use GPUs in jobs that are **parallelizable**.
- GPUs are Single Instruction Multiple Thread.
- You will learn a scalar-like language to program them.

- The LHCb HLT1 uses GPUs efficiently, getting more from the detector in near real-time.
- Boundary conditions are very relevant.

- GPUs are not limited to HEP, many HPC applications exist.
- AI is a change of paradigm that is impacting many fields.

Resources Used in the Talk

- Talk *Designing (New) C++ Hardware* by O. Giroux.
- Course *TinyML and Efficient Deep Learning Computing* by Song Han.