Status and plans for IPUs

Marco Adinolfi

University of Bristol

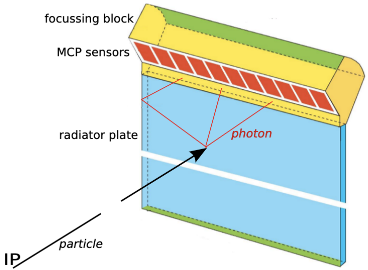
10 July 2024

IPU reminder

- The IPU itself is made up of many independent processing units, called tiles.
- Each tile consists of a single multi-threaded (workers) processor and its local memory.
- We are currently using a GC200 machine with 4 IPU, each consisting of 1472 tiles with 6 workers, for a total of 8832 workers per IPU.
- For the GC200 model each tile has 624 kB SRAM $\longrightarrow$ an IPU with 1472 tiles has $\sim$ 900 MB in processor memory.

- I/O data variables (tensors) are transferred to/from IPU and each element is placed on (mapped) to a specific tile.
- Tensors mapping is defined at compilation and cannot be changed during execution.
- Tiles can exchange data pretty fast. The compiler introduces ad-hoc code to execute the data transfer.
- If, during execution, a worker writes into a tensor element, workers on other tiles cannot access that element, even in read mode.
- Input data memory locations can be overwritten after the first time they are read.
- All data exchange across tiles is therefore fixed at compilation time and cannot be changed during the course of the execution.
- The 624 kB have to suffice for all I/O data, internal variables, code footprint, exchange-data code used by the tile.

TORCH reminder



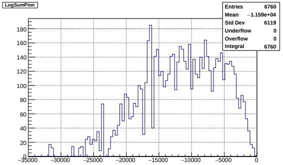
- A 3D (x, y, t) image of a TORCH module event has over 25M bins.
- The (x, y) image of a single particle is not a circle hence we use PID algorithm inspired by the RICH PID.
- There are 18 such modules and each can be treated as an entirely independent entity.
- The game is for each module:
  1) to estimate the fraction of hits expected for each particle/mass hypothesis combination;
  2) to estimate the probability each hit was produced by each combination;
  3) use this information to find the minimum $\mathcal{L}$.

$$\log \mathcal{L} = \sum_{\gamma} \log \left[\underbrace{\sum_{track\ j, j \neq t} \frac{N_j}{N_{tot}} P_j(\vec{x_\gamma} | h_j^{\text{best}})}_{\text{PDF for the best hypothesis assignment to all other tracks in the event}} + \underbrace{\frac{N_t}{N_{tot}} P_t(\vec{x_\gamma} | h_t^{\text{best}})}_{\text{PDF for the considered track}} + \underbrace{\frac{N_{bkg}}{N_{tot}} P_{bkg}(\vec{x_\gamma})}_{\text{PDF for the background assumed flat}} \right]$$

Where were we?

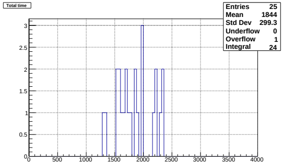
Forward reconstruction

- Generate the photon energy PDF.
- Trace $O(1M)$ photons through the module $\Longrightarrow$ list of pixel hits (x, y, t).
- Obtain a 3D image of the particle on TORCH.
- For each hit measured in TORCH define its probability of being produced by the combination as the fraction of photons with an hit in the corresponding pixel.
- Use these probabilities to estimate the $\mathcal{L}$ of the PID assignment.
- Use $\mathcal{L}$ to obtain DLLs.

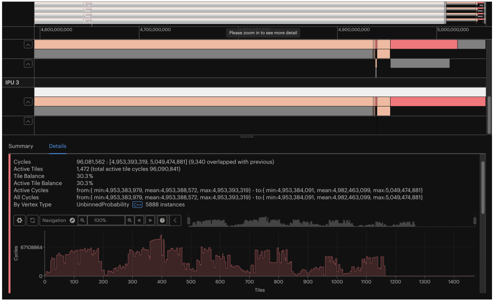


Backward reconstruction

- We calculate the probability a photon is emitted with the exact energy and azimuthal angle to hit the pixel.
- Account for reflections on the module sides, maximum 6 in X (i.e. 13) and 1 in Y.
- Work in $(E_\gamma, \phi_\gamma, t)$ instead of the (x, y, t) $\Rightarrow$ Find the Jacobian of the variables transformation
- Simulate 4 photons with $E \pm dE$ and $\phi \pm d\phi$ and trace them from the center of the track to the detector.
- For each combination trace $13 \times 2 \times 4 = 104$ simulated photons.



Exploitation of the IPU - I



Since then we have vastly increased our understanding of IPUs.

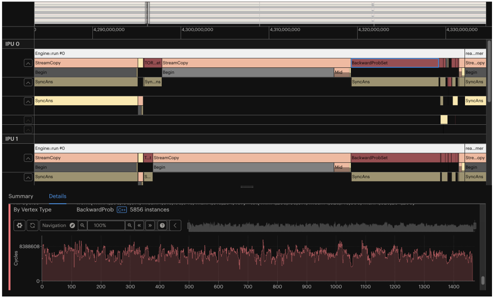
- The first attempts were extremely inefficient.

1. Many tiles are not used: events were badly distributed across tiles.
2. The number of cycles required by the tiles varies by orders of magnitude: the workload is not split correctly across the tiles.
3. The IPUs also require very different number of cycles.

- The tile memory constraints impose fixing limits on the sizes of the tensors used by the reconstruction.

- These in turn impose limit on the number of particles per IPU and the number of hits per tile, hence on the number of probability calculations that can be perfomed.
- To fully exploit the IPUs, probability calculations have to be distributed across the tiles in a non trivial way and simple implementations are not very efficient.

Exploitation of the IPU - II

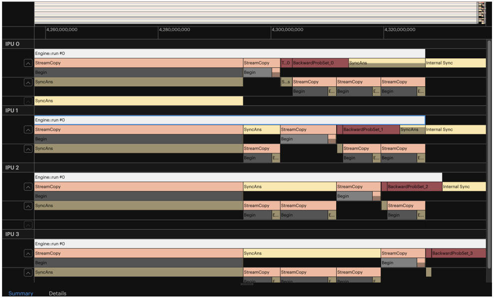


- The current implementation of the algorithm vastly improves the exploitation of the IPU
- Each IPU requires roughly the same time to process its input.
- The workload distribution in much more even across tiles.
- A few tiles are reserved for the $\log \mathcal{L}$ calculation
- Most of the processing time is spent in the calculation of the hit probabilities.

Note:

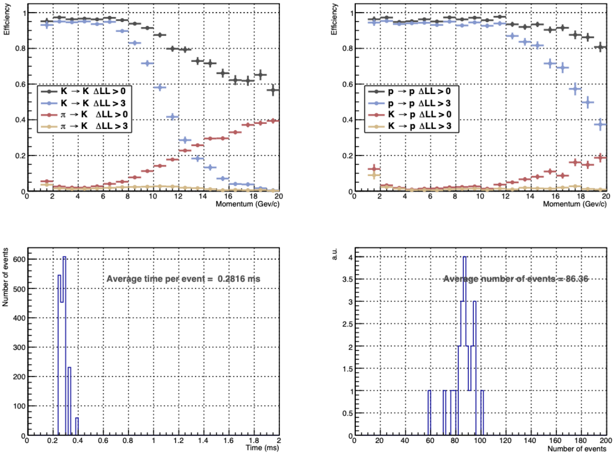
- The calculation of the $\log \mathcal{L}$ is affected by the tiles memory limits and it cannot be performed directly and needs to be split in several steps.
- Nevertheless the iteration to run the $\log \mathcal{L}$ sums requires $< 5\%$ of the total exeecution time.
- The transfer of the hits information to the IPU requires $\sim 50\%$ of the total execution time.

Staggering the input



- Currently input data is very efficiently packed before being transferred to the IPU.
- Other than applying some form of compression, which obviously will cost execution time, the amount of data to be transferred cannot be reduced.
- A more efficent way to stagger the data transfer so that eecution can start on an IPU while data is transferring on the other is currently being investigated

Current status



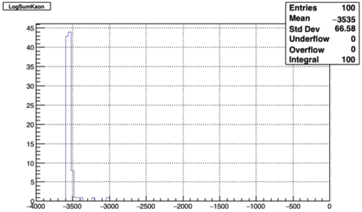
- With the current implementation the PID reconstruction in TORCH requires $\sim 282\mu s$.
- Further improvements in terms of code optimization and algorithm modifications are currently ongoing.

Extending IPU usage to RICH - 1

- Having successfully used IPUs to achieve a very fast TORCH PID reconstruction it is only natural to ask if they can be exploited elsewere.
- The most obvious candidate is the RICH PID reconstruction
- The expression $\log \mathcal{L}$ that needs to be minimized is the same for TORCH and RICH
- In principle the only difference in the algorithm is the different geometries of the 2 detectors.
- $\Longrightarrow$ Ideally the same framework can be used for TORCH and RICH.

Extending IPU usage to RICH - 2

- In order to achieve a proof of principle that the RICH PID reconstruction can be efficiently performed on IPUs we:
- Generated single track events in the RICH;
- Assumed no background whatsoever;
- Developed the **Forward reconstruction** algorithm for the RICH.
- Correctly defined $\log \mathcal{L}$ are achieved.



→ Have a working RICH PID reconstruction algorithm on IPU using toy tracks

- Planning to apply the IPU RICH **Forward reconstruction** to LHCb data.
- The **Backward reconstruction** for RICH is also being developed

Summary and conclusions

- TORCH PID reconstruction is fully available on IPU and it is already very fast: 280μs per event.
- Improvements to further speed it up are being developed, we expect at least a further factor 8-10 improvement.
- RICH PID **Forward reconstruction** is also now available on IPU and we are ready to characterize its timing performance.
- RICH PID **Backward reconstrution** on IPU will be available is a few weeks.