# Lumi Counters

## Throughput and testing improvements

Sergio Andrés Estrada

UNIVERSIDADE DA CORUÑA
INTALENT
UDC - INDITEX
GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
Financiado por la Unión Europea
NextGenerationEU

# The issues

- Kernels working with Luminosity counters took a great portion of running time:
  - Velo + Calo + SciFi Lumi counters -> 25,2% running time

| Time (%) | Total Time (ns) | Instances | Avg (ns) | Med (ns) | Min (ns) | Max (ns) | StdDev (ns) | Name |
|---|---|---|---|---|---|---|---|---|
| 17.9 | 24,698,053,657 | 5,712 | 4,323,888.9 | 4,263,778.0 | 2,822,412 | 7,139,158 | 577,278.5 | seed_xz::seed_xz (seed_xz::Parameters) |
| 12.9 | 17,815,124,929 | 5,712 | 3,118,894.4 | 3,073,612.0 | 951,364 | 5,847,763 | 772,682.1 | calo_lumi_counters::calo_lumi_counters |
| 7.9 | 10,941,678,138 | 5,712 | 1,915,559.9 | 1,823,831.5 | 1,052,356 | 10,014,214 | 556,156.1 | velo_search_by_triplet::velo_search_by_triplet |
| 7.8 | 10,782,218,049 | 5,712 | 1,887,643.2 | 1,858,807.0 | 401,314 | 4,437,202 | 522,048.8 | scifi_lumi_counters::scifi_lumi_counters |
| 5.5 | 7,596,854,872 | 5,712 | 1,329,981.6 | 1,289,605.5 | 676,098 | 3,133,294 | 310,768.4 | seed_confirmTracks::seed_confirmTracks |
| 4.6 | 6,291,094,791 | 5,712 | 1,101,382.1 | 1,075,732.5 | 742,467 | 2,322,122 | 172,421.1 | gather_selections::run_lines |
| 4.5 | 6,205,299,468 | 5,712 | 1,086,362.0 | 1,049,860.0 | 302,305 | 2,615,402 | 335,875.2 | velo_lumi_counters::velo_lumi_gec_counters |

(hlt1_pp_matching_no_ut_1000KHz sequence and MEP_dumps_09_04_24/bu_289232* data)

  - Not detected by tests as simulation data does not include luminosity events.
- Tests for Lumi counters were unstable both in CPU and GPU:
  - Hard to test if improvents done to kernels were properly implemented.

# Fixing tests

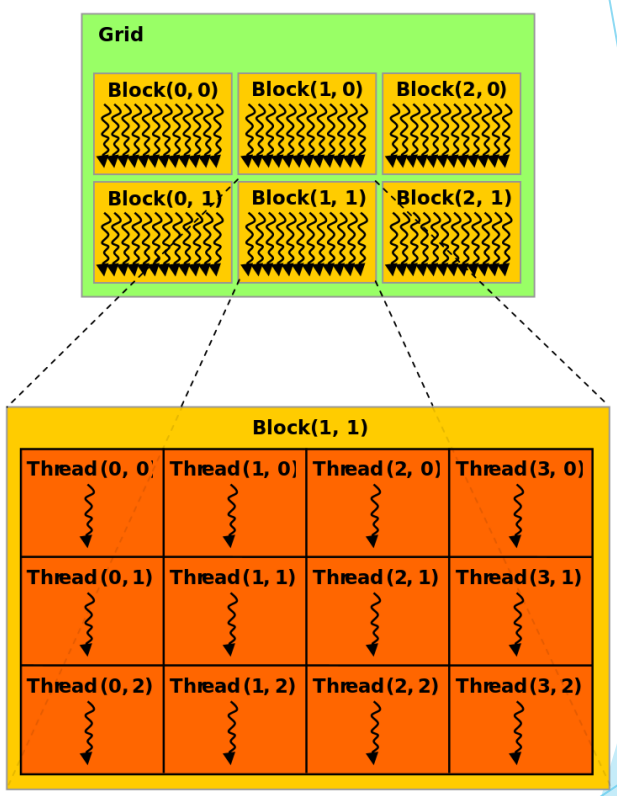
- Tests worked by running the sequence over a fixed set of events, and printing the Lumi counters for a single event every X events (usually 100).
- To check whether a test passed or not, the output of the run was compared to the reference.
- It was found that the events being printed varied across executions.
- The approach:
  - Instead of printing the Lumi counters when the event was processed, they are stored in a sorted map where key = event ID.
  - At the end, follow the original method and print the desired number of events -> printing all events would create outputs bigger than desired.
- With this change, the test were stable in CPU, but not in GPU.

# Obtaining stability

- There were two stability issues to be solved after tests' changes:
  - PV Lumi counters were non-deterministic on GPU:
    - The kernel that selected unique PVs (pv_beamline_cleanup) obtained the index to save the PV using atomic operations, which order of execution can vary.
    - The PV Lumi counters kernel later uses one of the selected PVs pseudo-randomly (actually, it uses the PV at position [eventID % nº of PVs]). This, make it difficult to test/reproduce the same case if the PVs order can vary.
    - To solve this, selected PVs are now ordered by their Z position (we are sure no two PV with the same Z position are selected) before returning from the kernel.
  - Calo Lumi counters are floats and updated from different threads using atomic operations -> again, order is not guaranteed.

# Improving throughput

- Main issue: kernels were being launched with a fixed number of blocks:
  - CaloLumiCounters -> 2
  - VeloLumiCounters and SciFicLumiCounters -> 4
- Parallelism was only exploited at event level and not inside them. Each event was processed by a single thread.
- First changes:
  - Number of blocks launched = number of events being processed at run time.
  - Parallelize event processing across block threads.
  - Use shared memory within block with atomic operations and reduce as much as possible the number of writes to global memory.



https://en.wikipedia.org/wiki/Thread_block_(CUDA_programming)

# Solution

- Velo and SciFi Lumi counters were unsigned -> with the previous changes, this kernel already performed better and in a determinic way.
- Calo counters were floating point values -> again, atomic operations lead to non-deterministic behaviour.
- The approach:
  - Use unsigned long long shared memory counters as a middle step.
  - In order to keep a significant amount of precisión -> multiply values by 1e8 before adding them as unsigned long long (i.e. 12,34567890 would be stored as 1234567890).
  - Divide the final result by the same factor before writing results to global memory.
  - Need to split counters in two, positive and negative values, and make the subtration at the end, since GPU do not support atomic operations on signed long long values.

# Results

- All tests are stable once a newly obtained reference is used.
- Throughput improvements:
- % of running time:

| Kernel | Original | Modified | Speed-up |
|---|---|---|---|
| Calo | 12.9 | 1.9 | **6.79** |
| SciFi | 7.8 | 0.3 | **26.00** |
| Velo | 4.5 | 0.5 | **9.00** |

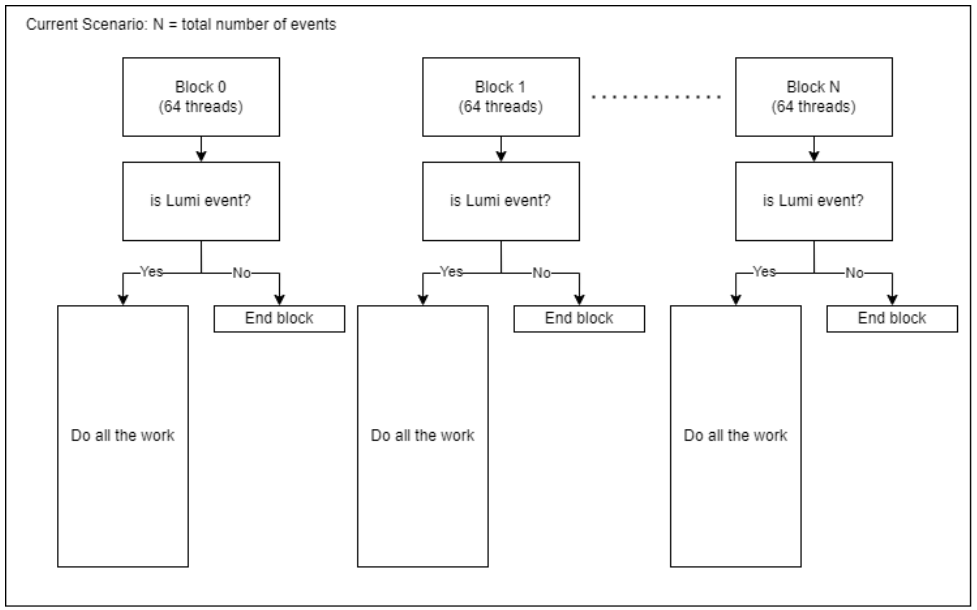
- Average single run time (in nanoseconds):

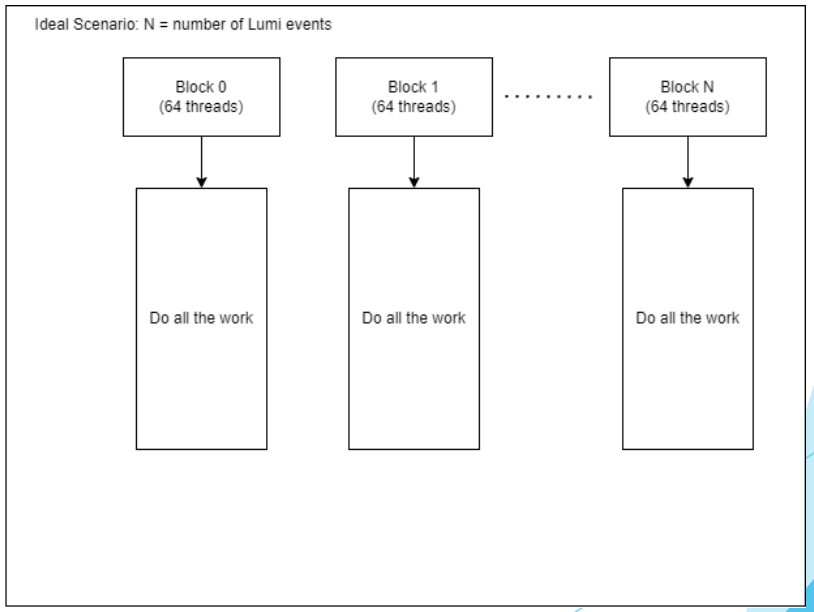
| Kernel | Original | Modified | Speed-up |
|---|---|---|---|
| Calo | 3,118,894 | 374,132 | **8.34** |
| SciFi | 1,887,643 | 64,614 | **29.21** |
| Velo | 1,086,362 | 100,209 | **10.84** |

- Overall sequence throughput gain ~14%
- Included in Allen v4r8.

# Future work

- Currently, a block is created for each event, but there are many which are not Lumi events. Those blocks just check that condition and end.
  - Implement a mechanism to know in runtime, how many events are Lumi ones, and only create and launch a block for them.



# Future work II

- This is already work in progress:
  - Allen MR 1461 “Prefer filters over selections” by Saverio Mariani (merged)
    - Creates the possibility to make the luminosity event mask

      https://gitlab.cern.ch/lhcb/Allen/-/merge_requests/1461
  - Allen MR 1628 “Lumi line under filter” by Shu Xian (WIP)
    - Will use the mask as the event list in the luminosity algorithms in order to get the desired shown behaviour and resource usage.

      https://gitlab.cern.ch/lhcb/Allen/-/merge_requests/1628