On the HLT side...

K. Trabelsi for the HLT/DAQ group and Software/Detectors groups
HLT is...

- max. input from L1 (designed value) = 30 kHz
- event size = 100 kB/ev, (PXD: 1 MB/ev)
- max. output = 10 kHz
- event size = (100 + 100 (PXD)) kB/ev

⇒ main functions of HLT
- trigger rate: reduction by a factor 3 (*)
- reconstruction without PXD → RoI feedback to Pixel Detector Readout
- tag events for calibration and physics skims
- monitoring (DQM on HLT/ExpressReco)

(*) or more depending also on how loose is the L1 trigger

1 HLT unit, ~400 CPU cores/unit
each unit is completely independent
keep up with luminosity increase
HLT is a detector

Hardware (DAQ)

Recent hardware upgrades:
2020 summer: 2880 → 4012 cores
2021 winter: 4012 → 4800 cores (75% of design)
2022 winter: +15%?

10 HLT units
4800 cores
#processes ~ 2 × #cores
HLT filter preparation beginning of 2021
''it's about time... to turn on HLT filter''

Thorough work to check the health of data taking with HLT filtering:

- HLT trigger
- DQM & detectors
- Physics
- Calibration
- Computing
- DQM & RC
- HLT
- L1 trigger

### Turning on HLT filter in 2021a

- **BII-7784** HLT code modification for 2021a and new Bhabha prescales
- **BII-7786** DQM with HLT filter on
- **BII-7787** trigger studies for physics (y line in exp14)
- **BII-7996** HLT calibration skims when HLT filter on
- **B2CC-416** Make files from HLT filtering test available to subdetector experts
- **B2CC-418** Check data size per unit luminosity with and without HLT filtering in exp 16
- **BII-8013** CR shifter plots for number of events before and after HLT filter decision
- **BII-8017** Gracefully leave when events have partial SoftwareTriggerResult
- **BII-8019** L1 in exp16

[Effective Cross Section (triggers)](chart)
HLT filter on (since end of March)

⇒ reduce trigger rate by factor 6-10
⇒ direct (and welcome) impact on data storage offline

⇒ further reduction 33 nb → 30 nb by end of 2021
HLT is software

Software optimization is the key to improve the situation in addition to tightening L1 trigger menu.

Monitoring during data taking is the key to understand better the HLT operation and optimize its performances (CPU and memory).
When starting to work on HLT in Fall 2020, we noted that HLT operation could be limited by higher luminosity (may not be able to operate beyond 9 kHz).

At that point (summer 2020) we were already running at \( \sim 5 \text{ kHz} \) @ \( 2 \times 10^{34} \text{ cm}^{-2} \text{s}^{-1} \)

Plan was to reach \( 4 \times 10^{34} \text{ cm}^{-2} \text{s}^{-1} \) by end of 2020c and maybe \( 6.5 \times 10^{34} \text{ cm}^{-2} \text{s}^{-1} \) by the end of 2021a

\[ \Rightarrow \text{we needed to make sure we can HLT smoothly until LS1} \]

\[ \Rightarrow \text{tightening L1 trigger so that HLT has to process fewer events} \]
Changes in L1 menu ⇒ L1 trigger rate

To lower the trigger rate, L1 menu evolves:

**exp 16 \rightarrow exp 17:**
CDC 2D lines (f) prescaled (\rightarrow y)
\(-20\%\)

**exp 17 \rightarrow exp 18:**
Bhabha events prescaled
\((-15\%)\)

Possible updates of L1 menu are being discussed to reduce further trigger rate:
- prescale on ECL lines (lml0, lml13), modified hie and sttsecl lines
⇒ L1 rate at 12kHz @ \(10^{35}/\text{cm}^2/\text{s}\)
When starting to work on HLT in Fall 2020, we noted that HLT operation could be limited by higher luminosity (may not be able to operate beyond 9 kHz)

At that point (summer 2020) we were already running at ~5 kHz @ $2 \times 10^{34} \text{cm}^{-2}\text{s}^{-1}$

Plan was to reach $4 \times 10^{34} \text{cm}^{-2}\text{s}^{-1}$ by end of 2020 and maybe $6.5 \times 10^{34} \text{cm}^{-2}\text{s}^{-1}$ by the end of 2021 a

⇒ we needed to make sure we can HLT smoothly until LS1

⇒ tightening L1 trigger so that HLT has to process fewer events

But might affect physics (especially low multiplicity)
Since we are taking data in 2021-2022, **Monitor and understand:**
- CPU time evolution
- Memory usage
- Hyperthreading
- HLT unit dependence

**Tighten L1 Trigger**

But might affect physics (especially low multiplicity)

**Software optimization**

**Tracking takes up most of the time in HLT processing**

There was a Software - Tracking + HLT session in the last B2GM to discuss the possibilities and the potential

But optimizing tracking will change the results (HLT decision, DQM...). Needs meticulous validation to make sure the changes are acceptable and cannot be implemented fast enough for ongoing data taking
Software optimization

SVDSpacePointCreator was the most time consuming module!

There were DQM and unpacker modules in top-20 CPU time consumers?!

Tracking takes up most of the time in HLT processing

There was a Software - Tracking + HLT session in the last B2GM to discuss the possibilities and the potential

But optimizing tracking will change the results (HLT decision, DQM...). Needs meticulous validation to make sure the changes are acceptable and cannot be implemented fast enough for ongoing data taking

Can buy us some time to survive until LS1
**HLT limits: Optimization**

On the other hand, there are some optimization tasks that produce exactly identical results (easy to validate)

- Needs detailed profiling
- Needs effective collaboration with all detectors/software

Can buy us some time to survive until LS1

---

**Software optimization**

- **Step 1**
  - Tracking takes up most of the time in HLT processing
  - But optimizing tracking will change the results (HLT decision, DQM...). Needs meticulous validation to make sure the changes are acceptable and cannot be implemented fast enough for ongoing data taking

- **Step 2**
  - SVDSpacePointCreator was the most time consuming module!
  - There were DQM and unpacker modules in top-20 CPU time consumers?!

- **Step 3**
  - FastReco?
**SVDSpacePointCreator Optimization**

### Step 1: Identical results

- **SVDSpacePointCreator** (-97 ms/evt, ~75x)

[Giulio Dujany, PR #7796]

Returning a reference instead of a copy of an object saves ROOT from destroying a lot of TObjects.

Optimal data object management (especially ROOT) becomes a recurring theme.

This has an impact on the Data Production too:
Mixed BB event production is ~3% faster!

Using the processing time of 1000 events of Exp 18 Run 1434 (~5 kHz @ 2.6 x 10^{34} cm^{-2} s^{-1}) with release-05-01-08 as reference.
DQM Optimization

**Step 1: Identical results**

- **SVDSpacePointCreator**: (-97 ms/evt, ~75x)
- **DQM modules**
  - TRGGDLQDM: (-8 ms/evt, ~6x)
  - ECLDQMEXTENDED: (-5, ~40x)
  - TRGCDCT3DConverters: (-2, ~10x)

**TRGGDLQDM**: Remove some unused plots; replace unused TH2D with arrays.

**ECLDQMEXTENDED**: Skip certain time-consuming events without much loss of information (memory has also been optimized: 600 → 350 MB).

**TRGCDCT3DConverters**: Use `boost::multi_arrays` instead of nested vectors.

Target for sum of DQM modules: < 1 ms/evt.
HLT limits: Optimization (step 1)

Unpackers Optimization

<table>
<thead>
<tr>
<th>Step 1: Identical results</th>
</tr>
</thead>
<tbody>
<tr>
<td>● SVDSpacePointCreator (-97 ms/evt, ~75x)</td>
</tr>
<tr>
<td>● DQM modules (-15, ~7.5x)</td>
</tr>
<tr>
<td>● Unpacker modules</td>
</tr>
<tr>
<td>○ SVDUnpacker (-9 ms/evt, ~2.5x)</td>
</tr>
<tr>
<td>○ CDCUnpacker (-1, ~2x)</td>
</tr>
<tr>
<td>○ ECLUnpacker (-2, ~3x)</td>
</tr>
<tr>
<td>○ TOPUnpackers (-2, ~5x)</td>
</tr>
</tbody>
</table>

Framework’s addRelation() is expensive (but easy to use).

SVD and CDC unpackers create relations which are not used later, removed them.

Mikhail implemented a quicker local RelationArray for ECLUnpacker.

CDCTriggerUnpacker change was not noticed earlier.

And SVD group is working to improve further!

Target for sum of Unpacker modules: < 3 ms/evt.
Optimizations implemented (new releases) during 2021

Software optimization reduces processing time on HLT.

Included only optimizations which don't affect the performances, which could be included at minor/patched releases.
Optimizations implemented (new releases) during 2021

33% improvement during data taking (2021)

∼ 9 kHz → ∼ 13 kHz
Should new be able to operate HLT smoothly till LS1!

Further improvements for release-06 (including efforts from TOP/SVD): ∼−15%
**HLT limits: Optimization (step 2)**

Need to prepare HLT for even higher luminosity

Tracking takes ~70% of total HLT processing time now

**Track refitting with different mass assumptions (K, p)**

- (TrackCreator)
  - Simplify geometry/faster fitter
  - Delay the refitting after HLT filter calculation
  - Limit momentum ranges for K, p TrackFitting

**Event t0 estimator**

- (FullGridChi2TrackTimeExtractor)
  - faster track fitter and lighter geometry

**K_S, Λ reconstruction**

- (V0Finder)
  - speed up code

**Pattern recognition**

- (TFCDC_AxialTrackFinderLegendre)
  - avoid checking all hits in each node
  - efficient caching

**Extrapolation**

- (Ext)
  - Limit #particle hypothesis?

**Muon Identification**

- (Muid)
  - move to post filter?

⇒ aim for an improvement of factor 2
Summary

- Patient and fruitful collaboration of the HLT/DAQ group with software and detector groups allowed to save \( \sim 30\% \) CPU with no impact on results (step 1)
  ⇒ should allow us to run HLT smoothly until LS 1 (keep monitoring to preserve the performance obtained)
  (improvement also for memory performance, see backup slides)

- Will focus from now on on the reconstruction (mostly tracking) with possible modifications in results so careful benchmarking needed (step 2 target: release-07, next summer)
  ⇒ aiming to reduce CPU/evt by a factor 2
  (while hardware + 25% to reach design)

- L 1 menu will keep changing to reduce input rate while HLT could evolve for faster decision (step 3 ?)
A special run (Exp 18 Run 2344) was taken on 16 June 2021, with only 3 units active, forcing the system into hyperthreading (CPU load ~85%).

We noticed processing time increased by 30% compared to the previous run with same L1 trigger rate and Luminosity.

\[ \therefore \sim 85\% \text{ processes are active} \Rightarrow \text{Processing time increases by } \sim 30\%. \]
HLT limits extrapolation

Considering only Exp 18 Runs 1080-1563 with following conditions:

<table>
<thead>
<tr>
<th>Essential</th>
</tr>
</thead>
<tbody>
<tr>
<td>○ Latest L1 menu</td>
</tr>
<tr>
<td>○ Same release (05-02-06)</td>
</tr>
<tr>
<td>○ With weakest unit (HLT07)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Realistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>○ Nominal background</td>
</tr>
<tr>
<td>○ 9 out of 10 units active</td>
</tr>
<tr>
<td>○ L1 trigger rate &gt; 2 kHz</td>
</tr>
<tr>
<td>○ #processes doesn’t drop*</td>
</tr>
<tr>
<td>○ We can go up to 85% processes active</td>
</tr>
</tbody>
</table>

Realistically, maximum limit of $13 \text{ kHz} @ \ 7 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ (or) average of $11 \text{ kHz} (+2) @ \ 6 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$

(+20% contingency in the L1 rate vs Luminosity)

*needs to be fixed
Memory usage: Dirty events

On April 2 (Exp 17 Runs 206-227), due to high HER background:

Rapid: startup phase (expected memory usage)

Slow: continuous operation, possible memory leaks and memory increase whenever a event requires more memory than any previous event

Avoiding this extra memory usage is essential for smooth operation of HLT and for surviving higher luminosity
SVD group @ [BII-8283] identified 70k SPs as the ideal limit, above which SpacePoints will not be created, and this is the overall reduction in the total memory usage after it is implemented:
Memory usage: Dirty events

The break down of improvement in individual modules:
addRelation() is expensive

From the perf profile of the HLT processing script on L1 passthrough events of Exp 17 Run 244, we saw that addRelation() takes up a lot of time, considerably from unpackers:

| 46%  | 1,583 | – | Belle2:SVD:SVDUnpacker:Module:Module::event |
| 11%  | 370   | – | Belle2:ECLUnpacker:Module:Module::event |
| 9.3% | 323   | – | Belle2:ECLDigitCalibrator:Module:Module::event |
| 5.1% | 177   | – | Belle2:TGFUnpacker:Module:Module::event |
| 4.2% | 145   | – | Belle2:TOPReconstructor:Module:Module::event |
| 2.0% | 99    | – | Belle2:SVDSpacePoint:Director:Module:Module::event |
| 1.9% | 64    | – | Belle2:ECLSplitter:Module:Module::event |
| 1.7% | 60    | – | Belle2:RelatedTracksCombiner:Module:Module::event |
| 1.5% | 53    | – | Belle2: ostat:Tracker:Module:Module::event |
| 1.5% | 52    | – | Belle2:TrackFinder:Module:Module::event,Belle2:TrackFinder:Module:Module::event,Belle2:TrackFinder:Module:Module::event |
| 1.2% | 43    | – | Belle2:ECLSplitter:Module:Module::event |
| 0.9% | 30    | – | Belle2:ECLTriggerFinder:Module:Module::event |
| 0.6% | 21    | – | Belle2:KAlp:Unpacker:Module:Module::event |
| 0.6% | 20    | – | Belle2:ECLTrigger:Module:Module::event |
| 0.4% | 13    | – | Belle2:PruneRecHit:Module:Module::event |
| 0.3% | 11    | – | Belle2:KMLDgeReconstruction:Module:Module::event |
| 0.3% | 9     | – | Belle2:Particle:Loader:Module:Module::event |
| 0.2% | 8     | – | Belle2:CKToSVD:Filter:apply |
| 0.2% | 8     | – | Belle2:ECLLocalMaximumFinder:Module:Module::event |
| 0.2% | 7     | – | Belle2:ClusterMatcher:Module:Module::event |
| 0.2% | 6     | – | Belle2:VISFinder:Module:Module::event |
| 0.1% | 4     | – | Belle2:VIDEx:PID:Module:Module::event |
| 0.1% | 4     | – | Belle2:KMLCA:Reconstruction:Module:Module::event |
| 0.1% | 4     | – | Belle2:Data:PID:Module:Module::event |
| 0.1% | 3     | – | Belle2:ECLWaveformFilter:Module:Module::event |

addRelation() is expensive because it resolves objects to indices and updates the bidirectional lookup index and more (but easy to use).

To be avoided when possible. If not, Martin suggested the possibility of creating a local* quicker RelationArray (by manually managing the indices) like he did in PXDClusterizer.

*Further care could be taken in filling the relations for optimal memory consumption.