HLT performance studies

in preparation for Run 3

Rafał Bielski

on behalf of the Trigger Core Software Team

Software Coordination meeting 29/10/2020

Outline

- Overview of HLT performance features and needs
  - Partially repeated from SPOT meeting 14/09
- HLT performance measurement and monitoring tools
- Recent results

HLT vs offline reconstruction

Similarities

- Both are part of Athena(MT) – a lot of common components between them
- Both use GaudiHive Avalanche Scheduler for Run 3

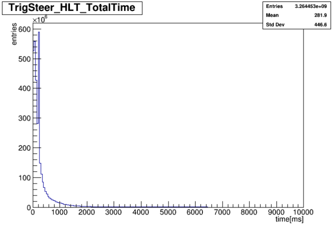
Differences

- Reco performance limited by memory needs, HLT by algorithm processing time and upstream I/O
- Run-3 HLT uses the same Scheduler but paired with complex HLT Control Flow
  - Partial event data processing (**event views**) and **early rejection** are the core of HLT design
- Very different and varied event-to-event CPU cost (see next slide)
- Main motivation for HLT athenaMT migration was **not memory saving**, but:
  - Closer integration with offline for easier maintenance and common features
  - Future-proofing – more modern code, potentially more flexible for future architectures
- HLT event processing involves Athena + HLTMPPU + DCM (online) / DcmEmulator (offline)
  - Possible to run HLT with `athena.py`, but `athenaHLT.py` (including HLTMPPU+DcmEmulator) better corresponds to online processing at P1

HLT performance needs

Purpose of the HLT: process events with very high input rate and select a small fraction of interesting ones

- 80–99% events are **rejected** (depending on streams and prescales) – these are never seen offline but constitute the majority of events in HLT and the performance needs to be optimised for them
- Most events take **<0.5 second** to process, some may take many seconds
- Performance needs constantly changing (also during a run) from:
  - Luminosity and pile-up
  - L1 rate and detector performance
  - Menu and prescales, streaming strategy



HLT event processing time online histogram, run 360026

(the peak at 250-300 ms is from a "time burner" running in standby before collisions)

HLT performance needs

The most important HLT performance metric is **total event throughput**

- HLT farm needs to sustain L1 rate up to 100 kHz
- Throughput inversely proportional to average event processing time
- I/O generally fast at P1 (using only small fragments of full event), but need to watch the data request rates to avoid hitting ROS/network limitations
- Menu/prescale changes allow to increase throughput but at the cost of physics
- Higher throughput for a given menu = more space for lower-threshold / more complex selections = more interesting events for physics analysis
- No throughput gain from MT over MP by definition, but potentially lower memory needs allow to buy more CPUs*

Run-3 HLT is **MP+MT hybrid** opening a large parameter space for optimisation

- Need to find the N forks / N threads / N slots combination allowing the highest event throughput
- For two similarly-performing configurations, the one with lower memory usage would be preferred
- Need to take system stability and error-handling consequences into account

*Not necessarily possible with IT tender, will need to see next year

HLT performance measurement and monitoring

1) Trigger cost monitoring

- Collects execution time data for every algorithm in a fraction of events
- Allows to estimate CPU cost of each algorithm and predict CPU cost of the full menu with given prescales – essential for menu design and prescale choices
- Extensive offline analysis and visualisation tools (in progress for Run-3 framework)
- Results presented at atlas-trig-cost.cern.ch (example below)

Display Algorithm Summary

Home → Directory: Technical-and-M-weeks → Run: T13-PhysicsP1-1Forks-12Threads-12Slots 383763 → Range: LumiBlock 00000 → Summary: HLT Algorithm

Trigger Expert: [New Processing Request] [View P1 Logs] [View Request Logs] [View Installed SW]

Table parsed from CSV file: Technical-and-M-weeks/costMonitoring_T13-PhysicsP1-1Forks-12Threads-12Slots_383763/csv/Table_Algorithm_HLT_LumiBlock_00000.csv

Algorithm Summary

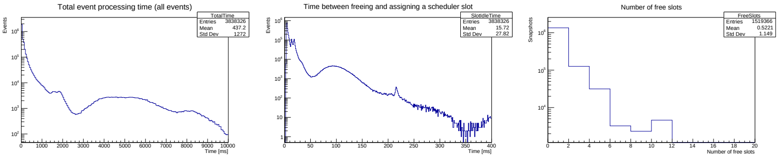
50 | Page 1 of 9 | Displaying 1 to 50 of 427 items

Name	Raw Active Events	Active Events	Calls/Event	Calls > 1000 ms	Event Rate [Hz]	Call Rate [Hz]	Alg Total Time [s]	Alg Total Time [%]	Alg Total Time/Call [ms]	Alg Total Time/Event [ms]
TrigFastTrackFinder__jet	4294	4.294e+04	1	4.294e+04	51.12	51.12	2e+05	54.51	4663	4663
HLTCaloClusterMakerFS	5398	5.398e+04	1	5.398e+04	64.26	64.26	8.205e+04	22.36	1529	1529
PFAlgorithm_ftf	2962	2.962e+04	1	0	35.26	35.26	9527	2.597	323.8	323.8
TrigMuonCombinedAlg_RoI	1318	1.318e+04	1.179	2610	15.69	18.5	5427	1.479	346.6	408.7
HLTCaloCellMakerFS	5398	5.398e+04	1	100	64.26	64.26	5394	1.47	98.89	98.89
TrigCaloClusterMakerMT_topoLC	1364	1.364e+04	1.205	2910	16.24	19.56	3973	1.083	240.2	289.3
InDetSCT_Clusterization__jet	4294	4.294e+04	1	0	51.12	51.12	3901	1.063	90.81	90.81
TrigMuPatTrackBuilder_RoI	8158	8.158e+04	1.031	0	97.12	100.1	3518	0.9589	41.49	42.78
TrigMuonCandidateAlg_RoI	8158	8.158e+04	1.031	0	97.12	100.1	3092	0.8427	42.96	44.54
InDetPixelClusterization__jet	4294	4.294e+04	1	0	51.12	51.12	3087	0.8414	72.31	72.31
PFTrackSelector_ftf	2962	2.962e+04	1	0	35.26	35.26	2972	0.8102	100.2	100.2
InDetSiTrackerSpacePointFinder__jet	4294	4.294e+04	1	0	51.12	51.12	2918	0.7954	67.7	67.7
HLTCaloClusterCalibratorLCFS	4294	4.294e+04	1	0	51.12	51.12	2915	0.7946	67.8	67.8

HLT performance measurement and monitoring

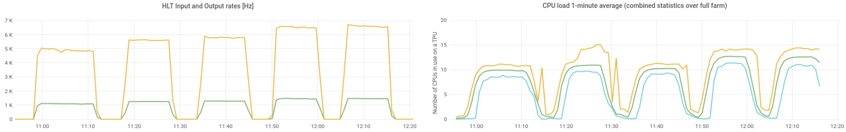
2) Online monitoring histograms

- A selection of histograms with simple metrics allow to evaluate overall HLT farm performance in real time
- Allow to narrow down performance limitations and other problems in real time
- Very useful also to analyse offline and compare configurations



3) Online monitoring data in PBEAST/Grafana

- Monitors huge amount of information at P1
- Most relevant for performance: rates and system load at different stages of TDAQ chain, mem/cpu



HLT performance measurement and monitoring

4) Dedicated offline studies (running athenaHLT* + perf tools)

- Collect prmon data, online histograms, cost data from different configurations to compare
  - Useful to debug specific issues or to test many configurations on small samples
  - Faster turnaround and easier to automatise than P1 running
  - athenaHLT scaling plots shown later are made using this approach
  - Close to online, but not identical – main differences in I/O (slower offline) and available resources (1 machine vs full HLT farm)
- Can also use Intel VTune to hunt for specific problems
  - Useful in hunting for fairly obvious and large issues
  - Limited usefulness otherwise as it cannot handle longer jobs with many threads/processes and software sampling is capped at 10 ms
  - Requires a lot of expertise and thought to interpret results case by case for anything other than "top 10 hot spots"

*athenaHLT.py emulates online running by reusing/emulating parts of the TDAQ software that steer Athena execution

Results introduction

The following slides present performance results from **athenaHLT offline** studies and from **P1 measurements** in Technical Runs

- We use labels "F-T-S" where F = N forks, T = N threads per fork, S = N slots per fork
  - N-1-1 is pure MP configuration, 1-N-M is pure MT, others are hybrid
- athenaHLT studies of wider phase-space are shown first
  - Chronologically, the P1 measurements were first and the offline ones were a follow-up
- Discovered a major flaw in data extraction procedure
  - Used `TH1::GetMean` from online histograms
  - Overflow is ignored by default and may introduce large error (shift)
  - Particularly for steeply falling distributions with long tails
  - Timing distributions and thus throughput measurement very affected
- Throughput information shown today extracted with alternative methods not using histograms
  - Detailed data separating event processing and I/O times available only from histograms
  - Plan to use extensible-axis histograms in the future
- We have lots of detailed in-job monitoring not shown today to help us understand the behaviour
  - For example data from SchedulerMonSvc presented in SPOT meeting 14/09

Scaling with athenaHLT

Release: Athena, 22.0.18

Trigger menu: LS2_v1 aka Dev_pp_run3_v1 without streamers (240 chains)

Data: 11.2k EnhancedBias stream events, run 360026 (late 2018), with original L1 prescales, decompressed

Machine: pc-tbed-pub-29, Xeon E5-2620 v4 2.10 GHz, 16 physical cores (2 sockets x 8 cores), 32 logical cores (2 threads per physical core), 32 GB RAM

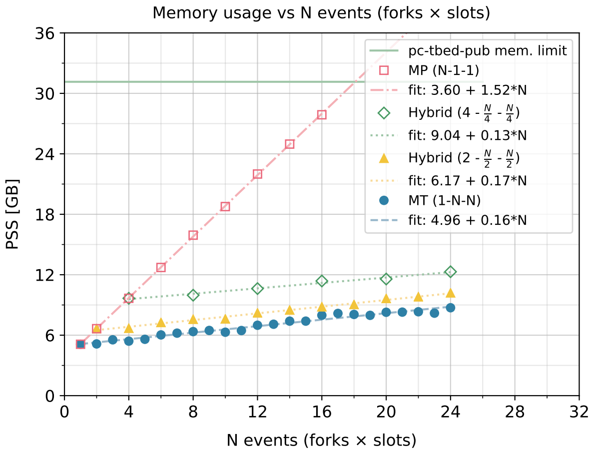
Configurations: threads=slots with 1, 2, 4 forks and pure MP, up to 24 total threads

Command:

```
prmon -i 3 -- athenaHLT.py \
--nprocs=${nF} --threads=${nT} --concurrent-events=${nT} \
-c "doStreamingSlice=False;doMonitorSlice=False;doBeamspotSlice=False;enableSchedulerMon=True;" \
-R 360026 -f /scratch/rbielski/large.decompressed.data._0001.data \
TriggerJobOpts/runHLT_standalone.py
```

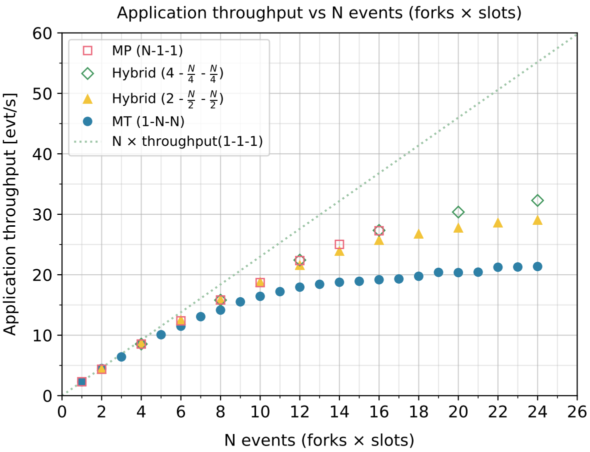
Disclaimer: we observe some MT errors in 22.0.18, mainly ATR-22141, but these are rare and shouldn't affect the measurements

Scaling with athenaHLT – memory



- Expected behaviour: linear scaling for all configs with MT slope much smaller
- Need 5.1 GB for first fork, then 100–200 MB per slot and 1.5 GB per fork
- At 24 events factor 4.5 difference between pure MP and pure MT

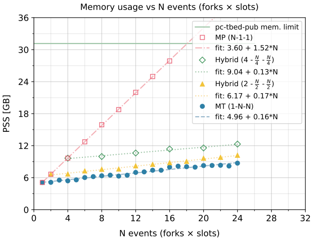
Scaling with athenaHLT – throughput

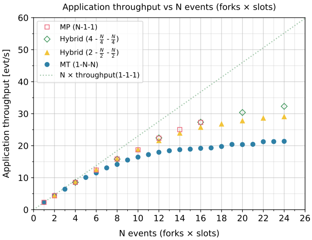


- Fairly good scaling until 8 events, then processing starts to slow down considerably
- Pure MT configuration much slower than pure MP at high thread/process counts
- Both trends partially due to hardware / low-level effects, but software effects are under investigation

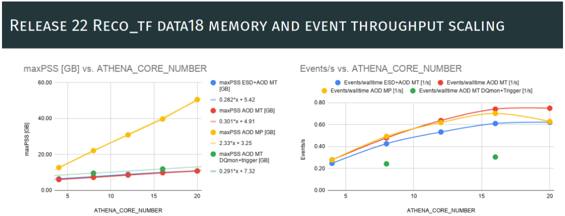
Comparison to offline reconstruction

HLT performance as measured with athenaHLT on pc-tbed-pub:



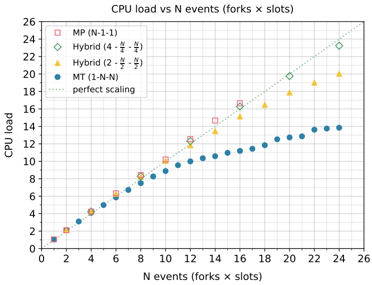


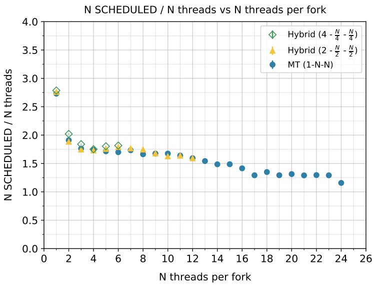
Offline reco performance on data as shown last week by Johannes:

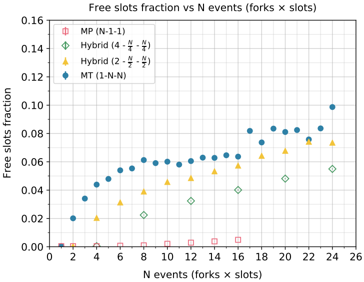


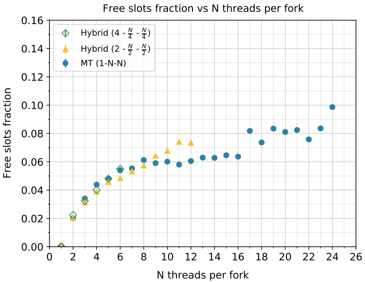
Very tempting to do this comparison, though the jobs are very different and probably mechanisms of saturation are also different to some extent

Teaser: other interesting plots









P1 performance results

Release: Athena, 22.0.18

Trigger menu: "Dev" (240 chains) and "PhysicsP1" (54 chains)

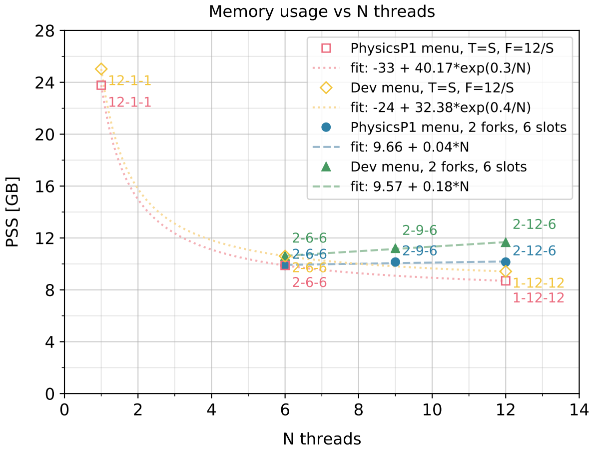
Data: 2.5k EnhancedBias stream events, run 360026 (late 2018), with original L1 prescales, preloaded into ROS at P1 and looped over many times (~3M events per run)

Machines: 5 racks, 39 TPUs each (~10% of the farm), each TPU with Xeon E5-2660 v4 2.0 GHz, 28 physical cores (2 sockets x 14 cores), 56 logical cores (2 threads per physical core), 65 GB RAM

Configurations: Tested **5 ways of processing 12 events in parallel** on each node (F-T-S = forks-threads-slots) – here in order from the least to the most MT:

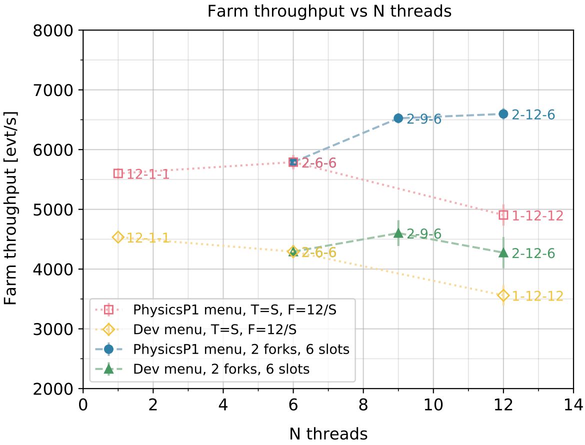
- **12-1-1** - pure MP, no MT, like in Run 2
- **2-6-6** - hybrid with threads = slots (12 threads in total)
- **2-9-6** - hybrid with threads = 1.5 × slots (18 threads in total)
- **2-12-6** - hybrid with threads = 2 × slots (24 threads in total)
- **1-12-12** - pure MT, no MP, threads = slots

Scaling at P1 – memory



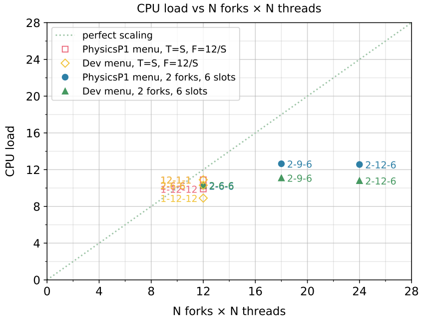
- Expected: MT uses much less memory than MP
- Expected: Dev menu (more chains) needs slightly more memory than PhysicsP1
- Interesting but not a problem: Adding threads per event requires a little bit more memory?

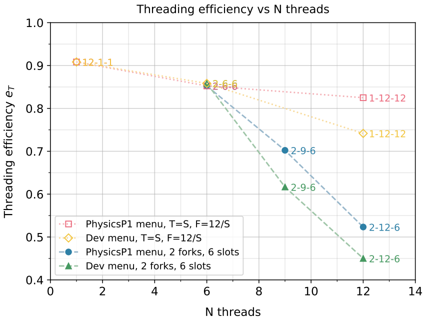
Scaling at P1 – throughput



- At threads=slots, MT configurations are slower than MP (also seen offline)
- Adding a few more threads (1.5 per event on average) speeds up the processing
- The speed-up saturates quickly, no improvement with 2 threads per event

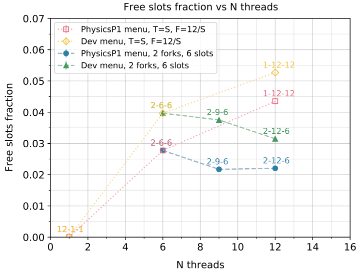
Scaling at P1 – CPU load

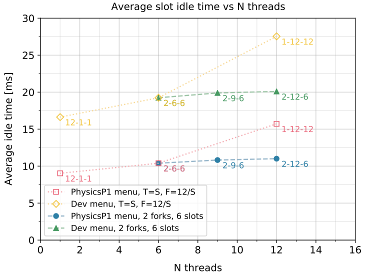


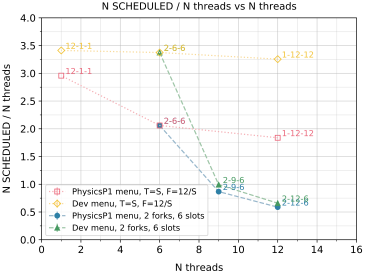


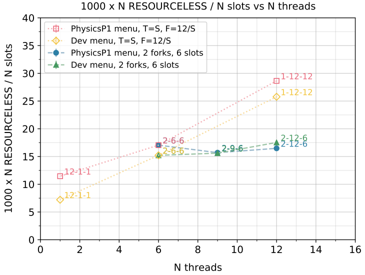
- Extra threads beyond 1.5 thread per event remain idle – no gain from adding more
- Mechanism is understood – mostly sequential Control Flow required for early rejection combined with a few long critical algorithms (see also the SPOT meeting slides)
- The optimal threads-to-slots ratio depends on menu, but is always slightly above 1
- We can monitor scheduler under-allocation in real time online thanks to SchedulerMonSvc
- The inefficiency of MT at threads=slots with respect to MP is under investigation (ATR-22112)

Teaser: other interesting plots









Summary

- Presented HLT performance needs, monitoring tools and results of P1 and offline scaling studies
- Extensive inventory of tools allows us to extract detailed information both in real time and post-run
- Memory scaling as expected, a lot can be saved in MT
  - Not so crucial for HLT, but useful input for Run-3 hardware purchase
- Pure MT at threads=slots is slower than pure MP
  - To some extent natural due to hardware/low-level effects
  - Partially caused by I/O and in-event effects (locks, unclonable algs)
  - Investigating whether there is still space for software improvements
- Some throughput improvement possible with larger thread pool
  - The optimal configuration for HLT somewhere between 1 and 1.5 threads per event
- To define the best Run-3 configuration, need to take into account throughput, memory usage, error handling, stability
  - Most likely will run with hybrid mode with a few slots per fork and 1.2–1.5 threads per slot
- Timeline:
  - Implement all chains by the end of 2020
  - Validation / optimisation / CPU cost estimates in winter/spring 2021
  - Run-3 farm hardware choice and order in summer 2021

Links

Past events:

- May 2020 HLT Hackathon
- 19/06/2020 Trigger Core Software meeting
- Technical Run 10 Jira ticket and performance plots
- Technical Run 12 Jira ticket and performance plots
- 14/09/2020 SPOT meeting
- September 2020 HLT Hackathon
- Technical Run 13 Jira ticket and 09/10/2020 Trigger Core Software meeting
- 23/10/2020 Trigger Core Software meeting

Future events:

- Trigger Workshop, 9–13/11/2020
- Technical Run 14, 16–20/11/2020